Institut für Parallele und
Verteilte Systeme

Abteilung Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 2724

Konzeption und Realisierung eines zeitbezogenen Contextcast

Ronald Schertle

Studiengang:	Informatik
Prüfer:	Prof. Kurt Rothermel
Betreuer:	Dipl.-Inf. Lars Geiger
begonnen am:	7. Januar 2008
beendet am:	8. Juli 2008
CR-Klassifikation:	C.2.1, C.2.2, C.2.4, F.4.1

Kurzfassung

Verteilte kontextbezogene Systeme sind Middleware-Systeme. Ihnen wird in der Forschung, speziell im Bereich der Vernetzung mobiler Endgeräte, zunehmende Aufmerksamkeit gewidmet. Teilnehmer eines kontextbezogenen Systems stellen dem System in Form von Kontexten Informationen über ihre Situationseigenschaften bereit. Diese Kontexte enthalten Attribute, die beispielsweise den Aufenthaltsort oder die Berufstätigkeit einer Person wiedergeben. Bei der kontextbezogenen Kommunikation werden diese Informationen für die Adressierung der Empfänger einer Nachricht verwendet.

Das Contextcast ist ein System, welches eine effiziente kontextbezogene Kommunikation ermöglicht. Eine Einschränkung des Systems ergibt sich bezüglich der Semantik der Nachrichtenübermittlung. Nachrichten können prinzipiell nur an Kontexte adressiert werden, die zum Zeitpunkt der Nachrichtenankunft bei einem Teilnehmer noch gültig sind. Dieser Einschränkung nimmt sich das zeitbezogene Contextcast an, das eine Nachrichtenadressierung an vergangene und zukünftige Kontexte eines Teilnehmers ermöglicht. Die Betrachtung der mit dem Kontextattribut „Zeit“ verbundenen Erweiterungen am bestehenden Contextcast stehen demnach im Mittelpunkt dieser Arbeit.

Die Performanz der Verfahren für das Routing von Nachrichten eines zeitbezogenen Contextcast wird anhand von ausgewählten Metriken durch Simulationen evaluiert. Dafür wurde eigens ein Prototyp des zeitbezogenen Contextcast implementiert. Abschließend folgt ein Ausblick in weitere Forschungsthemen und mögliche Optimierungen des zeitbezogenen Contextcast.

Abstract

Distributed context-aware systems are middleware systems, that are gaining increased attention, specially in the research of mobile device interconnection. Participants of a context-aware system provide information about their individual situation to the system in the form of contexts. These contexts contain, for instance, attributes describing the location or profession of a person. In terms of context-aware communication this information is used to address the receivers of a message.

Contextcast is a system that enables context-aware communication in an efficient way. A constraint of the system arises from the fact that the semantics of message delivery only allows messages to address contexts which are valid when the message arrives at a participant. The temporal Contextcast eliminates this constraint and therefore allows messages to address historic and future contexts. The extensions of the contexcast related to the attribute "time" of a context are the focus of this paper.

The performance of the developed methods for a routing of messages in a temporal Contextcast is determined by simulation with chosen metrics. For this purpose, a prototypical implementation of the temporal Contextcast has been used. The conclusion of this paper indicates further research topics and optimizations concerning a temporal Contextcast.

Inhaltsverzeichnis

Abbildungs- und Tabellenverzeichnis v

Abkürzungsverzeichnis vii

1 Einführung 1
- 1.1 Einleitung und Motivation . . . 1
- 1.2 Aufbau dieser Arbeit . . . 2

2 Grundlagen 3
- 2.1 Overlay . . . 3
- 2.2 Xcast und Xcast+ . . . 5
- 2.3 Publish-Subscribe . . . 7
  - 2.3.1 Event-Notification-Services . . . 7
  - 2.3.2 Siena . . . 10
  - 2.3.3 Historische Notifications . . . 17
- 2.4 Contextcast . . . 19

3 Problemstellung 29
- 3.1 Beispielszenarien . . . 30
  - 3.1.1 Zeugenfahndung . . . 30
  - 3.1.2 Messehinweis . . . 31
- 3.2 Anforderungen . . . 32

4 Aspekte eines zeitbezogenen Contextcast 35
- 4.1 Historisierung der Kontexte . . . 39
- 4.2 Historisierung der Nachrichten . . . 43
- 4.3 Zeitbezogene Nachrichten . . . 47
- 4.4 Datentyp für zeitbezogene Angaben . . . 49
- 4.5 Routing und Matching . . . 52
- 4.6 Zusammenfassung . . . 63

5 Exploration der Routingverfahren 67
- 5.1 Duplikatvermeidung . . . 68
  - 5.1.1 Nachrichtenduplikate mit gleichem Empfänger . . . 68
  - 5.1.2 Gebündeltes Contextcast historischer Nachrichten . . . 72
- 5.2 Verteilte Auswertung futuristischer Nachrichten . . . 80
- 5.3 Kontextdatenbankverteilung . . . 87

6 Evaluierung 91
- 6.1 Simulator . . . 91
- 6.2 Komponenten . . . 92
- 6.3 Simulationsaufbau und -ablauf . . . 94
- 6.4 Resultat . . . 96
  - 6.4.1 Backtracking historischer Nachrichten . . . 96
  - 6.4.2 Verteilte Auswertung futuristischer Nachrichten . . . 100
- 6.5 Fazit . . . 104

7 Zusammenfassung und Ausblick 109

7.1 Zusammenfassung 109
7.2 Ausblick 111

A Simulationsergebnisse **115**

Literaturverzeichnis **A**

Abbildungsverzeichnis

2.1 Routing in einem Overlay-Netz 4
2.2 Publish/Subscribe API 9
2.3 Beispiel einer Notification 11
2.4 Beispiel eines Filters 12
2.5 Siena API 14
2.6 Contextcast API 20
2.7 Komponenten des Contextcast 21

4.1 Komponenten eines zeitbezogenen Contextcast 37
4.2 Konzepte des zeitbezogenes Contextcast 39
4.3 Kontextdatenbank-API 41
4.4 Mailbox-API 46
4.5 Nachrichtenkategorien eines zeitbezogenen Contextcast 48
4.6 Routing und Matching historischer Nachrichten 56
4.7 Struktur einer zeitbezogenen Nachricht 63
4.8 Struktur eines Kontextes 65
4.9 Benutzer-API eines zeitbezogenen Contextcast bei den ContextNodes 66

5.1 Duplikaterkennung 70
5.2 Backtracking Varianten 73
5.3 Backtracking Beispiel 74
5.4 TTL-Backtracking Beispiel 77
5.5 Beispiel für eine verteilte Auswertung futuristischer Nachrichten 83

6.1 ContextNode-Empfänger der historischen Nachrichten 97
6.2 Maximale Backtracking-Tiefe 98
6.3 Nachrichtenaufwand des Backtrackings 99
6.4 Nachrichtenbestätigungen durch Trusted Nodes 99
6.5 Nachrichtenaufwand für die Zustellung der historischen Nachrichten 100
6.6 Bestätigte ContextNode-Empfänger der futuristischen Nachrichten 101
6.7 ContextNode-Empfänger im Vergleich 102
6.8 Nachrichtenaufwand für das Routing futuristischer Nachrichten 103
6.9 Nachrichtenaufwand im Vergleich 103
6.10 Zwischengespeicherte futuristische Nachrichten 4D 104
6.11 Zwischengespeicherte futuristische Nachrichten 3D 105

Tabellenverzeichnis

4.1 Funktionalität der Komponenten eines zeitbezogenen Contextcast 38
4.2 Operatoren und Auswertungsregeln für Zeitintervalle 50
4.3 Kategorisierung der zeitbezogenen Nachrichten 54
4.4 Lebensdauer zeitbezogener Nachrichten 58
4.5 Matching im zeitlichen Verlauf 64

6.1 PeerSim-Schnittstellen der zyklenbasierten Einheit 92

A.1 Originäre Zustellung historischer Nachrichten 115
A.2 Vollständiges Backtracking der historischen Nachrichten 116
A.3 Tiefenbeschränktes Backtracking der historischen Nachrichten 116
A.4 Originäre Zustellung futuristischer Nachrichten 117
A.5 Verteilte Auswertung futuristischer Nachrichten 117
A.6 Zwischengespeicherte futuristische Nachrichten 118

Abkürzungsverzeichnis

ABNF	Augmented Backus-Naur-Form for Specifications
API	Application Programming Interface
CN	Context Node
CORBA	Common Object Request Broker Architecture
CR	Context Router
DR	Designated Router
GPS	Global Positioning System
ID	IDentifier
IGMP	Internet Group Management Protocol
IP	Internet Protocol
ISP	Internet Service Provider
JEDI	Java Event-Based Distributed Infrastructure
LAN	Local Area Network
LOT	LOcal Time
NTP	Network Time Protocol
PADRES	Publish/Subscribe Applied to Distributed REsource Scheduling
REBECA	Rebeca Event-Based Electronic Commerce Architecture
RTT	Round Trip Time
SCE	Short Circuit Evaluation
SIENA	Scalable Internet Event Notification Architectures
SNTP	Simple Network Time Protocol
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TIB/Rendezvous	The Information Bus/Rendezvous
TN	Trusted Node
TTL	Time To Live
UT	Universal Time
UTC	UT Coordinated
VID	Virtual IDentity
WAN	Wide Area Network
XCAST	EXplicit Multi-UniCAST
XML	Extensible Markup Language
YEAST	Yet another Event-Action Specification Tool

1 Einführung

Der erste Abschnitt diese Kapitels beginnt mit einer Einleitung 1.1 in das Themengebiet dieser Arbeit. Eine kurze Beschreibung verwandter Systeme und deren Fähigkeiten leitet in die Motivation dieser Ausarbeitung und des zu entwickelnden Systems über. Dabei werden ebenfalls kurz die Kernanforderungen an das System geschildert. Abschnitt 1.2 schließt das Kapitel mit einem Überblick über den Aufbau der Arbeit ab.

1.1 Einleitung und Motivation

Mit der zunehmenden Verwendung mobiler Endgeräte mit umfangreicher peripherer Ausstattung, wie zum Beispiel GPS-Modulen und drahtlosen Netzwerkkarten, sind verstärkt Kommunikationsmöglichkeiten gefragt, die sich von den klassischen Kommunikationsparadigmen des *Unicast*[1] und *Multicast*[2] unterscheiden. Diese Nachfrage basiert unter anderem auf der Dynamik der sich möglicherweise ständig in Bewegung befindlichen mobilen Geräte. Unter diesen Umständen gestaltet sich die Nachrichtenübermittlung durch Übertragungsprotokolle wie dem IP-Multicast als sehr nachrichtenaufwendig. Darüber hinaus können keine zusätzlichen Informationen über die Teilnehmer der Kommunikation in den Nachrichtenversand aufgenommen werden, da die Adressierung der Empfänger einer Nachricht auf statischen Adressen beruht.

„Neuere“ Systeme wie das Publish/Subscribe ermöglichen eine schwächere Bindung der Kommunikationspartner. Die im Interesse der Teilnehmer stehende Nachrichten, beispielsweise über Ereignisse in der näheren Umgebung eines Senders, können basierend auf dem Inhalt der Nachricht durch die Empfänger des Systems ausgewählt werden. Dabei können die unterschiedlichsten Attribute einer Nachricht für die Filterung gemäß den Interessen eines Empfängers verwendet werden. Eine explizite Anmeldung der Empfänger an Gruppen, die ein ähnliches Interesse an bestimmten Nachrichten teilen, wie es beim Multicast der Fall ist, ist nicht notwendig. Diese Art der Kommunikation fällt unter den Begriff des „Inhalt-basierten Routing“. Andere Systeme wie das Geocast erlauben zwar keine beliebige Verwendung von Attributen zur Nachrichtenadressierung, beziehen im Gegensatz zum Publish/Subscribe jedoch explizit den physischen Standort von Geräten in die Kommunikation mit ein. Dies ermöglicht ein gezieltes Routing von Nachrichten an Empfänger, die sich in einem vorgegebenen geographischen Bereich befinden. Attribute, die die Lokation eines Gerätes beschreiben, nehmen eine besondere Position ein, da sie in ungeordneten Netzen einen beträchtlichen Nachrichtenaufwand erzeugen können.

Das im Jahre 2007 entwickelte Contextcast ermöglicht die effiziente Nutzung beliebiger Attribute in einem System, wobei verstärkt ein Augenmerk auf die Unterstützung von Lokationsattributen gelegt wurde. Das Kommunikationsparadigma des Contextcast unterscheidet sich dahingehend von dem eines Publish/Subscribe-Systems, dass der Nachrichtensender die Empfänger einer Nachricht bestimmt, wie man es von klassischen Kommunikationsparadigmen her kennt. Die Eigenschaften eines Empfängers werden im Contextcast in Form von Attributen repräsentiert, die den Teilnehmern des Systems durch Kontexte bekannt gemacht werden. Die Anforderungen an das System fallen in die Bereiche der Effizienz, damit verbunden der hohen Skalierbarkeit und der gezielten Unterstützung mobiler Endgeräte. Die Konzeption des Systems ist Teil des Sonderforschungsbereichs Nexus des Instituts für Parallele und Verteilte Systeme an der Universität Stuttgart.

[1] Punkt-zu-Punkt-Nachrichtenübertragung
[2] Nachrichtenübertragung an eine Menge von Teilnehmern

Ein weiteres spezielles Attribut, das neben dem Standort eines Benutzers einer gesonderten Betrachtung bedarf, ist der Faktor „Zeit“. Im bisher bestehenden Contextcast ist nur eine Adressierung von gegenwärtig am System mit einem Kontext angemeldeten Empfängern möglich. Sind diese für einen längeren Zeitraum nicht erreichbar oder mit dem System verbunden, können an diese Empfänger adressierte Nachrichten nicht ausgeliefert werden. Ein weiteres Problem entsteht in dem Fall, dass eine Nachricht basierend auf in der Vergangenheit gültigen Eigenschaftswerten von Teilnehmern an die Empfänger übertragen werden soll. Wie sich die Eigenschaftswerte eines Teilnehmers im Laufe der Zeit ändern, so ändert sich auch sein Kontext. Vergangene Kontexte, die aus diesem Grund nicht mehr den aktuellen Zustand eines Teilnehmers beschreiben, sind in dem System nicht mehr verfügbar. Daher können diese auch nicht mehr in das Routing der Nachrichten mit einbezogen werden. Die Integration der speziellen Zeitattribute von Kontexten in das System erfordert deshalb Änderungen und Erweiterungen am System selbst.

Inhalt dieser Arbeit ist ein zeitbezogenes Contextcast-System, das auf dem regulären Contextcast basiert und dieses um die zeitbezogene Adressierung von Empfängern einer Nachricht erweitert. Die Anforderungen an das zeitbezogene Contextcast, wie z.B. die Effizienz des Routings, sind unter anderem die gleichen wie an das reguläre Contextcast. Zusätzlich müssen effiziente Routing-Algorithmen für die Übermittlung zeitbezogener Nachrichten entwickelt werden. Dazu gehören unter anderem Verfahren zur Speicherung der benötigten Informationsobjekte und die Formalisierung der Attribute und Filter. Die Erweiterung zu einem zeitbezogenen Contextcast sollt dabei möglichst so generisch gehalten werden, dass das Hinzufügen oder Ändern von Konzepten des regulären Contextcast nicht zu Konflikten führt. Die Semantik der Nachrichtenzustellung ändert sich im Gegensatz zum ursprünglichen Contextcast gänzlich. Als Haupteinsatzgebiet wird weiterhin von großen Internet-ähnlichen Netzen ausgegangen. Dies fordert eine hohe Skalierbarkeit des Systems. Die entwickelten Verfahren laufen allesamt auf einem Overlay ab, welches bis auf die Möglichkeit des Nachrichtenaustauschs zwischen verbundenen Knoten keine Anforderungen an das darunter liegende Netzwerk stellt.

1.2 Aufbau dieser Arbeit

Kapitel 2 dieser Arbeit erläutert die Grundlagen, die mit einem zeitbezogenen Contextcast in Verbindung stehen. Den Inhalt bilden grundlegende Kommunikationsarten, verwandte Systeme wie das Publish/Subscribe und das Contextcast als Grundsystem für ein zeitbezogenes Contextcast. Ebenfalls wird ein Einblick in verwandte Arbeiten gegeben. Die Problemstellung und Anforderungen, mit denen ein zeitbezogenes Contextcast konfrontiert wird, werden in Kapitel 3 aufgezeigt. In Kapitel 4 wird auf die Konzeption des zeitbezogenen Contextcast entsprechend den Anforderungen des vorherigen Kapitels eingegangen. Dazu gehört die Eruierung der notwendigen Komponenten eines zeitbezogenen Contextcast, die Bereitstellung der notwendigen historischen Informationen, die Formalisierung der verwendeten Attribute und Filter, sowie das Routing der verschiedenen Nachrichten. Als nächstes werden die möglichen Routingverfahren der verschiedenen Nachrichten in einem gesonderten Kapitel 5 erörtert und in Lösungsräumen anhand unterschiedlicher Gesichtspunkte gegenübergestellt. Kapitel 6 beginnt mit einer Beschreibung des Prototypen eines zeitbezogenen Contextcast, der für die Simulation der entwickelten Verfahren für das Routing von Nachrichten verwendet wurde. Anhand der resultierenden Simulationswerte werden die Verfahren anschließend hinsichtlich verschiedener Metriken analysiert und bewertet. Kapitel 7 fasst den Inhalt dieser Arbeit zusammen und weist auf weitere interessante Forschungsgebiete im Zusammenhang mit dem zeitbezogenen Contextcast hin. Eine genaue Auflistung der Simulationsergebnisse, die aus Kapitel 6 resultieren, bildet den Abschluss dieser Arbeit.

2 Grundlagen

Das folgende Kapitel vermittelt die Grundlagen für das in dieser Arbeit beschriebene System. Zu aller erst wird in Abschnitt 2.1 eine Einführung in Overlay-Netze gegeben. Der folgende Abschnitt 2.2 beschäftigt sich mit einem Multicast-ähnlichen Übermittlungsverfahren, das im Schwerpunkt dieser Arbeit Anwendung findet. Publish/Subscribe-Systeme, in die Abschnitt 2.3 einen Einblick bietet, weisen starke Ähnlichkeit zu dem im Hauptteil verwendeten Contextcast-System auf. Nachdem die unterschiedlichen Kategorien von Event-Notification-Services, von denen das Publish/Subscribe nur eine mögliche Kategorie darstellt, besprochen wurden, wird speziell auf eine konkrete Implementierung eines Publish/Subscribe-Systems eingegangen. Anschließend folgt ein kurzer Einblick in im weiteren Sinne verwandte Arbeiten. Contextcast ist in Abschnitt 2.4 beschrieben und bildet die Basis für die Erweiterung zu einem zeitbezogenen Contextcast.

2.1 Overlay

Definition 2.1.1 *"Overlay networks have a network semantics layer above the basic transport protocol level that organizes the network topology . . . "* (aus [DO03])

Wie der Name *Overlay* schon sagt, muss es sich bei Overlay-Netzen – oder kurz Overlays – um Netzwerke handeln, die über etwas anderem liegen. Aus der Definition 2.1.1 geht hervor, dass Overlays eine zusätzliche Netzwerkschicht über bestehende Transportprotokolle legen, um eine eigene *Netzwerktopologie*[1] mit eigenen Routingverfahren aufbauen zu können. Für eine genauere Erläuterung der Schichtenarchitektur von Netzen wird auf [Zim80] verwiesen. Ein Overlay befindet sich in Hinsicht auf das in [Tan03] verwendete fünfschichtige Referenzmodell auf der Anwendungsebene. Die Overlay-Schicht kann eine eigene Semantik bezüglich der Kommunikation zwischen den Komponenten des Overlays festlegen, die sich gänzlich von denen des darunter liegenden *Underlays*[2] unterscheidet. Der Nachrichtenaustausch zwischen Komponenten des Overlays wird über das Underlay vollzogen. D.h. die physischen Komponenten eines Overlays sind auch gleichzeitig Teil des Underlays. Es muss jedoch nicht jede Komponente des Underlays im Overlay vorhanden sein. Einfache logische Verbindungen zwischen Komponenten des Overlays können aus einem Pfad mit mehreren Komponenten des Underlays bestehen.
In Abbildung 2.1 ist die Struktur eines Overlays und das Routing der Nachrichten grafisch dargestellt. Die in diesem Beispiel übertragene Nachricht durchläuft innerhalb der Knoten die einzelnen Schichten des Netzprotokolls. Die Knoten des Overlays besitzen alle fünf Schichten des Protokolls und führen daher auch das Overlay-Routing auf Anwendungsebene durch. Die Router des Underlays leiten die Nachricht nach den Vorgaben des Vermittlungsprotokolls weiter. Die Empfänger der Nachrichten werden durch die Overlay-Knoten auf dem Nachrichtenpfad festgelegt.

Die Anordnung der Komponenten eines Overlays kann auf unterschiedlichste Art und Weise erfolgen. *Peer-to-Peer-Systeme*[3] stellen ein prominentes Beispiel für ein Overlay dar. In diesem Kontext wird zwischen strukturierten und unstrukturierten Peer-to-Peer-Systemen unterschieden. Während bei unstrukturierten Peer-to-Peer-Systemen die Verbindung und

[1] Beschreibt die Struktur der Vernetzung mehrerer Geräte
[2] Das unter dem Overlay befindliche Netz
[3] Verteilte Systeme, die aus gleichberechtigten Komponenten bestehen und sich in einem Netz selbst organisieren

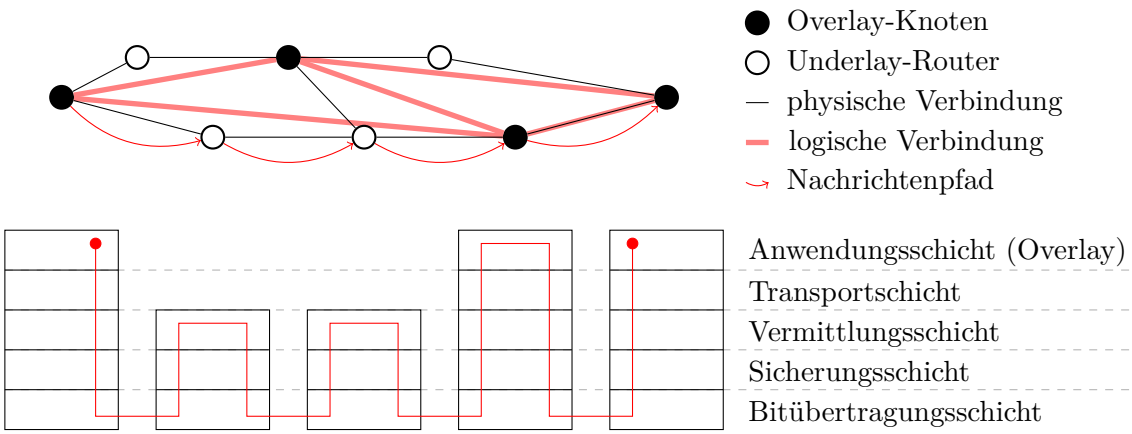


Abbildung 2.1: Routing in einem Overlay-Netz (in Anlehnung an [Tan03] Abbildung 3.2)

Anordnung der Komponenten willkürlich gewählt werden kann, werden diese bei strukturierten Peer-to-Peer-Systemen nach ihrem gespeicherten Inhalt angeordnet. Aus dieser Anordnung ergibt sich die Topologie des Netzes, die bei strukturierten Peer-to-Peer-Systemen von Ringstrukturen [SMK+01] über Schmetterlingsnetzen [MNR02] bis hin zu mehrdimensionalen kartesischen Koordinatensystemen [Rat02] reicht. [LCP+05] bietet eine ausführlichere Übersicht über die verschiedenen Topologien bekannter Peer-to-Peer-Systeme. Eine genauere Auseinandersetzung mit den unterschiedlichen Typen von Peer-to-Peer-Systemen kann in [Sch07] nachgeschlagen werden. Die verwendete Begrifflichkeit für Peer-to-Peer-Systeme kann so auch auf Overlay-Netze übertragen werden, die nicht der Peer-to-Peer-Definition folgen. In der Literatur ist eine unterschiedliche Verwendung der Begriffe anzufinden. In [DO03] ist zum Beispiel von Flooding-basierten Netzen und Overlays die Rede, was der hier verwendeten Begrifflichkeit der unstrukturierten und strukturierten Overlays entspricht.

Einige Vorteile der Overlays sind:

- Overlays lassen sich zusammen mit andere Systemen auf dem gleichen Transportprotokoll nutzen.
- Um den Transport von Nachrichten braucht sich das Overlay nicht zu kümmern, da dies Teil der Vermittlungs- und Transportschicht ist.
- Für Overlays muss kein eigenes Netz zur Verfügung stehen. Bestehende Netze wie das Internet können verwendet werden.
- Vorhandene Transportprotokolle müssen nicht auf das Overlay angepasst werden.

Unstrukturierte Overlays verwenden meist eine Flooding-basierte Verteilung von Nachrichten, da entsprechend der nicht strukturierten Anordnung der Komponenten keine Information über den Aufenthaltsort spezifischer Daten besteht. Im Schwerpunkt dieser Arbeit wird ein unstrukturiertes Overlay-Netz verwendet, da strukturierte Overlays ihre Datenobjekte gezielt speichern und keine Schlüsselwort-basierte Suche oder eine Suche basierend auf Bedingungen bezüglich des Inhalts eines Datenobjekts ermöglichen. Die logischen Verbindungen der Komponenten eines unstrukturierten Overlays können beliebig gewählt werden. Die Komponenten selbst versenden periodische PING-Nachrichten, um andere Komponenten des Systems als Nachbarn auszumachen und anschließend eine Verbindung zu diesen herzustellen. Alternativ können die Verbindungen zwischen den Komponenten auch statisch festgelegt werden. Dies zieht allerdings einen hohen administrativen Aufwand nach sich. Die

Topologie entspricht daher einem ungerichteten zyklischen Graphen. Als Transportprotokoll kann ein beliebiges Protokoll verwendet werden, das die Möglichkeit zur Übermittlung von Nachrichten von einer Komponente zur anderen ermöglicht. Werden von der Applikation, welche das System nutzt, weitere Anforderungen an das System gestellt, müssen diese auf den Hosts durch Ende-zu-Ende Bestätigungen realisiert werden. Ein Beispiel hierfür ist die zuverlässige Kommunikation, falls sie nicht bereits durch das zugrunde liegende Protokoll des Underlays angeboten wird. Eine Anpassung der Overlay-Struktur durch sich ändernde Verhältnisse im Underlay ist deshalb nur bedingt notwendig.

2.2 Xcast und Xcast+

Multicast ist ein Konzept zur Reduzierung der Nachrichtenlast bei der Verteilung einer Nachricht an mehrere Empfänger im Gegensatz zum Unicast. In [Cis06] ist eine Übersicht über grundlegende Konzepte des *IP-Multicast* gegeben. Grundsätzlich existieren zwei unterschiedliche Methoden für den Aufbau eines Spannbaums, entlang welchem die Nachrichten verteilt werden: Den Shared-Tree und den Source-Trees Ansatz. Während Source-Trees, hinsichtlich einer gegebenen Metrik, immer den kürzesten Pfad für Nachrichten vom Sender zu den Empfängern wählen, erfolgt die Nachrichtenzulieferung beim Shared-Tree Ansatz immer von einem „Rendezvous-Knoten“ aus. Der Shared-Tree Ansatz führt nicht zwangsläufig zu kürzesten Strecken, weshalb ein Mehraufwand an Nachrichten in Kauf genommen werden muss. Andererseits müssen für Source-Trees eine Menge von Informationen in den Routern des Baums gehalten werden. Diese werden durch die Verwendung eines Shared-Trees reduziert. Im Allgemeinen müssen die Teilnehmer einer Multicastgruppe bei Verwendung von Source-Trees die Sender der Gruppe kennen, damit sie sich gezielt für die Nachrichten eines Sender anmelden können. Beim Shared-Tree Ansatz wird dieser Umstand durch die Verwendung eines gemeinsamen Rendezvous-Knoten für die Sender einer Gruppe umgangen.
Beide Verfahren eignen sich für eine Unterstützung mittlerer und großer Multicastgruppen, skalieren aber durch die Notwendigkeit der Speicherung zusätzlicher Routinginformationen bei den Routern schlecht für viele kleine Gruppen.

Xcast ist kein Internet-Standard und wird in [BFI+07] als ein experimentelles Protokoll beschrieben. Es entstand aus der anerkannten schlechten Skalierbarkeit traditioneller IP-Multicast-Protokolle [DC90] für eine große Anzahl kleiner Gruppen. Xcast kann allerdings nicht als Ersatz für die existierenden Multicast-Verfahren angesehen werden, da es wiederum aus im Folgenden genannten Gründen schlecht für große „Multicastgruppen“ skaliert. Für den Aufbau der Multicast-Verteilbäume wird der Source-Trees Ansatz des IP-Multicast verwendet. Vorteile von Xcast sind die Verwendung kürzester Pfade für die Nachrichtenverteilung, die Reduzierung der zusätzlichen Routinginformationen auf ein Minimum, sowie die Unterstützung sehr vieler unterschiedlicher „Multicastgruppen“. Der zuletzt genannte Vorteil entsteht dadurch, dass die Empfänger einer Nachricht explizit in den Nachrichtenheader[4] eingefügt werden. Xcast kennt keine expliziten Multicastgruppen und daher auch keine Multicastadressen. Jede Teilmenge der Menge verfügbarer Empfänger kann als Empfängerliste der Nachricht genutzt und in diesem Kontext als Multicastgruppe verstanden werden. Dazu muss der Sender natürlich die Möglichkeit haben, die Empfänger einer Nachricht auszumachen. Dies geschieht entweder über ein externes System, wie z.B. dem Web, oder beispielsweise auch über das in [Dee89] und [Fen97] beschriebene *Internet Group Management Protocol* (IGMP).

Der Vorgang beim Nachrichtenversand gestaltet sich wie folgt: Zuerst übermittelt der Sender die Nachricht an einen Router. Der Router, der eine Xcast Nachricht erhält, führt anschließend die folgenden Schritte durch:

[4] Nachrichtenkopf

1. Er ermittelt für jeden Empfänger aus dem Nachrichtenheader die Next-Hop[5] Verbindung, über die der Knoten zu erreichen ist. Diese Information kann den ursprünglichen Routingtabellen für Unicasts entnommen werden.

2. Er partitioniert die Liste der Empfänger gemäß ihren Verbindungsleitungen

3. Er schickt jeweils eine Kopie der Nachricht mit geändertem Nachrichtenheader über die Verbindungsleitungen an die Empfänger. Der geänderte Nachrichtenheader enthält nur noch die Empfänger, die über eine Verbindungsleitung erreichbar sind.

4. Enthält eine Kopie einer Nachricht nur noch einen Empfänger, so kann diese in eine entsprechende Unicast-Nachricht umgewandelt werden.

In Xcast existiert demnach kein *Single-Point-of-Failure*[6]. Nachrichten werden entsprechend dem Unicast auf den kürzesten Pfaden übermittelt, welche sich gegebenenfalls auch an Topologieänderungen anpassen. Es werden keine symmetrischen Leitungen benötigt. Insgesamt stellt sich das Verfahren als sehr einfach heraus. Nachteile, die sich durch die Verwendung von Xcast ergeben, sind die Größe und Beschränkung des Nachrichtenheaders, die aufwändigen Operationen in den Routern und die mangelhafte Skalierbarkeit unter der Voraussetzung vieler großer Multicastgruppen. Da jeder Empfänger in den Nachrichtenheader aufgenommen wird, werden die Nachrichten für eine große Anzahl an Empfängern selbst recht groß. Die Nachrichtenheader sind in ihrer Größe jedoch beschränkt. Daher werden nur kleinere Multicastgruppen unterstützt. Bei entgegen gesetzten Anforderungen muss auf das IP-Multicasting zurückgegriffen werden. Bei den Routern selbst muss für jeden Empfänger die Ausgangsleitung für den Next-Hop ermittelt werden, was in der Summe eine recht aufwändige Rechenoperation ergibt.

Das im Schwerpunkt dieser Arbeit verwendete *Contextcast*-System besitzt eine potentiell unendlich große Anzahl von Multicastgruppen. Der Aufwand für eine Gruppenverwaltung wie beim IP-Multicast kann in diesem Fall nicht gerechtfertigt werden. Daher bietet sich die Verwendung von Xcast als Übertragungsprotokoll an.

Xcast+ (Explicit Multicast Extension) [SKPK01] wird als Erweiterung von Xcast verstanden. Xcast+ soll eine, im Gegensatz zum normalen Xcast, bessere Unterstützung von großen Multicastgruppen bieten. Tatsächlich stellt Xcast+ jedoch eine Kombination des Source-Trees Ansatz von IP-Multicast und Xcast dar.
Beim Source-Trees Ansatz meldet sich jeder Empfänger über eine IGMP-Nachricht mit der Multicastadresse und der Senderadresse an einem Sender an. Xcast+ wandelt diese Nachricht beim ersten Router, der die IGMP-Nachricht erhält und im folgenden Designated Router (DR) genannt wird, in eine Xcast Nachricht um, der die Adresse des DR enthält. Diese wird dem Sender übermittelt. Der DR auf der Senderseite empfängt die Nachricht und kann somit eine Liste von DRs verwalten, die einer Multicastadresse zugeordnet werden. Sobald der Sender eine Nachricht an eine Multicastgruppe senden möchte, wird diese Nachricht von dem DR auf der Senderseite in eine Xcast-Nachricht umgewandelt, in der alle DRs der Gruppe als Empfänger im Nachrichtenheader notiert sind. Das Routing der Xcast-Nachrichten wurde in Xcast+ nicht verändert. Durch die Verwendung der DRs als Empfänger im Nachrichtenheader anstatt der eigentlichen Empfängeradressen wird die Anzahl an Adressen im Header reduziert, da meist mehrere Empfänger an einem DR angemeldet sind. Dies ist auch der Grund für eine effizientere Unterstützung von größeren Multicastgruppen als im ursprünglichen Xcast. Weiterhin verbindet Xcast+ die Schnittstelle des ursprünglichen in [DC90] erläuterten Host-Group-Model IP-Multicast mit den Vorteilen von Xcast.

[5] Bezeichnet den nächsten Schritt auf einem Pfad zum Empfänger
[6] Die zentrale Komponente eines Systems, deren Ausfall ein Komplettausfall des Systems verursacht

In dieser Arbeit wird das DR-Konzept von Xcast+ auf ContextNodes im Contextcast übertragen. Die IP-Multicast Schnittstelle kann so nicht im Contextcast verwendet werden, da so etwas wie Multicastadressen nicht existieren. Weiteres wird in den Folgekapiteln noch näher erläutert. Wenn im Folgenden in Beziehung zum Contextcast von Xcast die Rede ist, ist ein Xcast mit der DR-Optimierung von Xcast+ zu verstehen.

2.3 Publish-Subscribe

Netzwerkbasierte und speziell verteilte Systeme, wie z.B. selbstorganisierende Systeme, Peer-to-Peer-Systeme oder *Event-Notification-Services*, basieren häufig auf der Interaktion schwach gebundener (*loosely-coupled*) Komponenten. Die Komponenten des Systems kommunizieren hierbei durch asynchrone Nachrichten. Diese schwache Bindung reduziert statische Abhängigkeiten zwischen den Komponenten und ermöglicht daher ein starkes Wachstum des Systems bis auf viele Millionen von Knoten. Solche Systeme sind weiterhin durch die Dynamik ihrer Komponenten gekennzeichnet. Eine Annahme ist, dass dem System fortlaufend und spontan neue Knoten beitreten und dieses genauso spontan verlassen bzw. im extremsten Fall einfach ausfallen. Aus Gründen der Skalierbarkeit wird daher oft auf die Architektur eines verteilten Systems mit schwach gebundenen Komponenten zurückgegriffen.

2.3.1 Event-Notification-Services

Ein System mit schwach gebundenen Komponenten stellt der schon genannte Event-Notification-Service dar. *Events* werden von Komponenten des Systems erzeugt, sobald sich ein bestimmtes Ereignis ergibt, an welchem andere Komponenten interessiert sein könnten. Dies geschieht durch den Versand von asynchronen Nachrichten, den sogenannten *Notifications*. Es existieren mehrere mögliche Ansätze, um diese Notifications zu den interessierten Empfängern zu routen. Dabei ist von Bedeutung, dass dem Erzeuger der Notification die daran interessierten Komponenten, weiter *Interessenvertreter* genannt, des Systems vorab bekannt sind. Ist dies der Fall, können diese Notifications gezielt zu den Empfängern weitergeleitet werden. Dies kann zum Beispiel unter Einsatz bekannter Unicast- oder Multicast-Protokolle geschehen. Letzteres bietet den Vorteil, dass im Regelfall insgesamt weniger Nachrichten anfallen, als wenn die Nachricht jedem Empfänger einzeln übermittelt werden muss. Sind dem Sender die Empfänger der Notification vorab nicht durch explizite Adressen bekannt, sind Kommunikationsprotokolle gefragt, welche sich von den klassischen Adressierungsmethoden unterscheiden.

Nachrichtenübermittlung Eine Möglichkeit zur Verteilung der Nachrichten unter diesem Umstand ist das *Broadcasting*. Hierbei wird die Notification an alle Komponenten bzw. Teilnehmer des Systems verteilt. Broadcasting wurde in dieser Hinsicht in der Forschung bereits intensiv untersucht. Verschiedene Lösungsansätze können in [Tan03] nachgeschlagen werden.
In einem verteilten Systems stellt diese Art der Nachrichtenverteilung eine einfache, aber auch äußerst nachrichtenintensive Methode dar. Bei zunehmender Teilnehmerzahl erhöht sich die Wahrscheinlichkeit, dass das System von Nachrichten überflutet wird und daher eine Skalierbarkeit des Systems nicht mehr gewährleistet werden kann. Für den Bereich des *Local Area Network* (LAN) wurden schon mehrere Event-Notification-Services entwickelt. Diese verwenden beispielsweise eine zentrale Server-Einheit für die Verwaltung und Zulieferung von Nachrichten (UNIX cron oder YEAST [KR95]), oder bauen eine Hierarchie von Servern für die Verteilung der Nachrichten auf (JEDI [CNF01], TIB/Rendezvous von TIBCO).

Beide Lösungsansätze sind für die Anwendung in einem *Wide Area Network* (WAN) nur bedingt praxistauglich. Während eine zentrale Server-Authorität einen Single-Point-of-Failure darstellt und in einem skalierbaren verteilten System generell vermieden werden sollte, wird selbst durch eine hierarchische Anordnung von Servern eine ungleiche Verteilung der Nachrichtenlast erzeugt. Bei baumartigen Hierarchien werden meist die näher an der Wurzel des Baums liegenden Knoten stärker belastet. Auch diese kann zu schlechter Skalierbarkeit des Systems führen, falls keine weiteren Maßnahmen für den Lastausgleich getroffen werden.

Eine andere Möglichkeit der Verteilung der Nachrichten basiert auf der Idee, dass die Interessenvertreter ihr Interesse an bestimmten Events im System bekannt machen und somit nur eine bestimmte Teilmenge aller Notifications an diese Interessenvertreter als Empfänger ausgeliefert werden müssen. Der Nachrichtenverkehr wird dadurch im Gegensatz zum simplen Broadcasting erheblich reduziert.

Kategorien Gemäß [CRW00] können die populären Event-Notification-Services in drei Varianten eingeteilt werden: *Kanal-basierte* (*channel-based*), *Themen-basierte* (*subject-based*) und *Inhalt-basierte* (*content-based*) Event-Notification-Services.
Vertreter der ersten Variante sind beispielsweise der iBus [Sof08], der CORBA Event Service [Obj04] und die Java Distributed Event Specification [Sun98]. Für die Verteilung der Informationen werden Kanäle aufgebaut. Die Konsumenten und Veröffentlicher der Nachrichten können sich an verschiedenen Kanälen anmelden und mit dem Push-/Pull-Mechanismus Nachrichten senden und empfangen. Ein Kanal kann an dieser Stelle mit einer Multicast-Adresse verglichen werden. Jeder Kanal steht für eine bestimmte Art von Nachrichten. Der Interessenvertreter macht dem System seine Interessen an spezifischen Nachrichten bekannt, indem er sich an den verschiedenen Kanälen anmeldet. Bei Themen-basierten Event-Notification-Services wird unabhängig vom Inhalt ein Themen-Feld in die Nachricht eingefügt, anhand derer die Interessenvertreter Nachrichten auswählen können.

Bei Inhalt-basierten Event-Notification-Services, welche für den Gegenstand dieser Arbeit von besonderem Interesse sind, erfolgt die Auswahl der Nachrichten, an denen die Interessenvetreter interessiert sind, anhand ihres Inhalts. Bekannte Vertreter dieser Art sind der CORBA Notification Service [Obj04], Elvin [SAB+00], Gryphon [BCM+99], Keryx [WH98], Siena [CRW99], REBECA [FM00, SK03] und PADRES [FJLM05, Fid06]. Diese Systeme unterscheiden sich in der Routingfunktionalität, den verschiedenen Ansätzen für Optimierungen, den Möglichkeiten zur Abfrage historischer Notifications und dem Umfang an Nachrichteninhalt, welcher für die Auswahl von Nachrichten zur Verfügung steht. Das Verfahren zur Auswahl der Nachrichten wird *Filtering* genannt. Die Nachrichten besitzen eine für das Filtering angepasste Formatierung. Die Architektur der Systeme entspricht einer klassischen Client-Server-Architektur. Gleichzeitig können diese Systeme als Middleware für hochskalierbare verteilte Applikationen angesehen werden. Für die Anbindung des Benutzers[7] stehen spezielle Zugangsknoten des Server-Netzwerks zur Verfügung. Mit Hilfe dieser tritt ein Client in der Rolle eines *Publishers*[8] oder *Subscribers*[9] auf. Ein Publisher veröffentlicht Notifications im System. Ein Subscriber hingegen vertritt seine Interessen an bestimmten Nachrichten durch die Anmeldung von *Subscriptions*. Aus diesem Grund werden diese Art der Inhalt-basierten Event-Notification-Services auch als *Publish/Subscribe*-Systeme bezeichnet. In der Verantwortung eines Publish/Subscribe-Systems liegt die Auswahl von Notifications nach den Interessen der Subscriber und die anschließende Übermittlung jener. Im Kontext der Auswahl von Notifications durch Subscriptions fällt die Auswertung der Subscriptions auf

[7] Client
[8] Veröffentlicher
[9] Interessenvertreter

publish(notification)
subscribe(expression)

Abbildung 2.2: Publish/Subscribe API in Anlehnung an [CRW99]

die Notifications unter dem Begriff *Matching*[10]. Dies soll durch ein skalierbares, effizientes (bezüglich des Matchings) und ausdrucksfähiges System (bezüglich der Subscriptions) realisiert werden.

Publish/Subscribe Die generelle Anwendungsschnittstelle (API) eines Publish/Subscribe-System ist in Abbildung 2.2 dargestellt. Diese API wird den Clients von den Zugangsknoten des Netzes zur Verfügung gestellt. Mit der Funktion *publish(notification)* kann der Veröffentlicher Notifications an das System übergeben. Die Funktion *subscribe(expression)* dient dem Benutzer in der Rolle des Interessenvertreters dazu, dem System das Interesse an bestimmten Notifications bekannt zu machen. Der Ausdruck, der an die Funktion übergeben wird, identifiziert alle Notifications, an denen der Client interessiert ist. Dieser Ausdruck wird in diesem Kontext auch als *Filter* bezeichnet. Der Client hat die Möglichkeit gegenüber dem System gleichzeitig als Veröffentlicher von Nachrichten und als Interessenvertreter aufzutreten. Ebenso ist dem Client die Anmeldung mehrerer Subscriptions an dem System gestattet. Das System ist im Weiteren dafür zuständig alle Notifications, welche zu den Subscriptions des Benutzers passen, an den betreffenden Benutzer auszuliefern. Dies kann über die Zugangsknoten unter Einsatz beliebiger Datenübertragungsprotokolle erfolgen. Da mehrere Clients an einer bestimmten Nachricht interessiert sein können, wird die entsprechende Notification auch an mehrere Clients ausgeliefert. Es handelt sich bei Publish/Subscribe also um ein Multicast-System. Anders als bei klassischen Multicast-Protokollen, wie z.B. in IPv6 [HD98], werden beim Versand einer Notification jedoch keinerlei explizite Adressen der Empfänger oder eine Gruppenadresse benötigt. Eine indirekte Adressierung der Empfänger erfolgt über Eigenschaftswerte, die Informationen über die Art und den Inhalt der Notification geben und mit der Notification übertragen werden. Die potentiellen Empfänger wählen durch Subscriptions, die sich auf die Eigenschaftswerte beziehen, welche Notifications sie erhalten möchten. Die Auswahl bestimmter Notifications kann im Laufe der Zeit geändert werden. Ohne die Kenntnis der Eigenschaftswerte einer Notification oder den Filtern eines Interessenvertreters können die Empfänger einer Notification jedoch nicht ermittelt werden. Diese schwache Koppelung von Veröffentlicher und Interessenvertreter macht das Netz sehr flexibel. Den Veröffentlicher interessiert es im Allgemeinen nicht, ob ein Interessenvertreter seine Subscription für die Nachrichten des Veröffentlichers vom System entfernt, oder ob ein neuer Interessenvertreter hinzukommt. Er überträgt einfach bei jedem Event Notifications an das System, auch wenn zu diesem Zeitpunkt keine Subscriptions für diese Notifications angemeldet sind. Anders als bei klassischen Kommunikationsparadigmen wählt in einem Publish/Subscribe-System der Interessenvertreter als Empfänger aus, welche Nachrichten er zugeliefert bekommt. Bei der expliziten Adressierung klassischer Übertragungsverfahren erfolgt die Auswahl der Empfänger einer Nachricht durch den Sender.

Nach [CRW00] ergeben sich folgende Herausforderungen an die Entwickler eines Publish/Subscribe-Systems:

- Für Empfänger existieren keine expliziten Adressen mehr, wie z.B. beim „Internet Protocol“. Diese werden durch die Datenwerte der Notifications und die Bedingungen der Filter ersetzt.

[10] Übereinstimmung prüfen

- Die Routingtabellen der Server sind demnach Abbildung von Bedingungen auf Verbindungen zu benachbarten Servern.
- Die Effizienz der Nachrichtenzuteilung wird durch die Effizienz des Auswertungsverfahrens beim Matching von Notifications auf vorhandene Subscriptions bestimmt.

2.3.2 Siena

Im Weiteren soll das Publish/Subscribe-Paradigma anhand des Systems Siena [CRW99] beschrieben werden. Nach einer kurzen Einführung zu Siena folgt eine fundierte Erläuterung der unterschiedlichen grundlegenden Nachrichtentypen des Systems und deren Auswertung als Grundlage des Routings. Anschließend folgt die Vorstellung der Siena-API, die dem Benutzer als System-Schnittstelle zur Verfügung steht. Die dem System zugrunde liegende Topologie stellt gewisse Anforderungen an das Routing von Nachrichten. Nach einer kurzen Beschreibung der Topologie werden die möglichen Routingverfahren und deren Optimierungen sowie die Routingtabellenstruktur besprochen. Dies bildet gleichzeitig den Abschluss des Abschnitts über Siena.

Siena integriert Optimierungen wie *Advertisements*[11] und *Covering*[12], um das System noch effizienter und damit skalierbarer zu gestalten. Der Begriff Covering, welcher später noch näher erläutert wird, wird in diesem Zusammenhang auch oft anstatt dem Begriff des Matching von Filter auf Notifications verwendet. Damit im Folgenden keine Verwirrung entsteht, wird der Begriff Matching exklusiv im Zusammenhang mit dem Filtering verwendet.

Wie bereits erwähnt werden Subscriptions in Form von Filtern dazu verwendet, bestimmte Notifications auszuwählen und einem Empfänger zu übermitteln. Dieser Filter stellt bestimmte Anforderungen an die Datenwerte einer Notification. Die Ausdrucksfähigkeit dieser Filter und der Notifications ist von dem verwendeten Datenmodell und den zulässigen Operatoren abhängig. Die Komplexität der Filter bestimmt die bei einem Server notwendigen Rechenoperationen für das Matching und Routing der Notifications. Deshalb steigt im Regelfall die Komplexität des Systems mit zunehmender Ausdrucksfähigkeit des Datenmodells, wobei gleichzeitig die Skalierbarkeit sinkt. Hier besteht also ein Trade-Off zwischen den zwei Entwicklungszielen. Bei der Entwicklung von Siena wurde größtenteils von dem verwendeten Datenmodell für Notifications und deren Struktur – also dem genauen Inhalt, der Formatierung und die zulässigen Datenwerte – abstrahiert. Das gleiche gilt für die zur Konstruktion der Subscriptions verwendeten Sprache, in diesem Fall zum Beispiel der möglichen Operatoren und Wertemengen. Diese Details sind zwar für das Routing, Covering und Matching von großer Bedeutung, unter anderem aber auch abhängig von den Applikationsanforderungen. Da sich Siena jedoch als Middleware-System versteht, wird von diesen Applikationsanforderungen abstrahiert. Das Augenmerk liegt hier auf den Konzepten des Matchings, den Korrelationen zwischen Notifications und Subscriptions und dem daraus resultierenden Einfluss auf die Vorgehensweise beim Routing.

Notifications Eine Notification ist die Datenrepräsentation eines Events. Sie ist eine Art strukturierter *Record*[13] bestehend aus einzelnen typisierten *Attributen*. Die Notification selbst besitzt keinen Typ, definiert sich also ausschließlich über die Werte ihrer Attribute. Ein Attribut selbst ist ein dreielementiges Tupel $\alpha = (type_\alpha, name_\alpha, value_\alpha)$ bestehend aus dem Datentyp, dem Namen und dem Wert des Attributs. Die zur Verfügung stehenden Datentypen sind angelehnt an die bekannten primitiven Datentypen von Programmiersprachen,

[11] Ankündigungen
[12] Überdeckung
[13] Datensatz

string	objekt	= automobil/pkw/cabrio
string	hersteller	= mercedes
string	modell	= slk
string	farbe	= weiß
integer	ps	= 184
string	währung	= euro
float	preis	= 36500
year	baujahr	= 2008

Abbildung 2.3: Beispiel einer Notification

z.B. string und float. Alle Datentypen sind fest definiert und können zur Laufzeit nicht verändert werden. Jedes Attribut besitzt einen genau festgelegten Wert und bezieht sich typischerweise auf bestimmte Charakteristika des Nachrichteninhalts. Diese einfache Struktur der Notifications ermöglichen die Umsetzung eines effizienten Routingalgorithmus. Abbildung 2.3 zeigt ein Beispiel einer Notification. Diese kann beispielsweise zur Ankündigung der Veröffentlichung eines neuen Automobilmodells verwendet werden.

Subscriptions, Filter und Matching In Subscriptions werden Filter verwendet um spezifische Notifications auszuwählen, die dem Verfasser der Subscription als Empfänger folglich übermittelt werden sollen. Eine Subscription entspricht einer Ansammlung einzelner Attributbedingungen, auch *Attributfilter* genannt. Diese stellen eine Beziehung zu den einzelnen Attributen einer Notification her. Ein Attributfilter ist ein vierelementiges Tupel $\phi = (type_\phi, name_\phi, operator_\phi, value_\phi)$. Für die Menge an Operatoren stehen typische binäre Operatoren wie z.B. $>, <, =, \neq$ für alle Datentypen und $substring(*), prefix(> *)$ und $suffix(* <)$ für Strings zur Verfügung.

Definition 2.3.1 *Ein Attributfilter ϕ stimmt mit einem Attribut α überein, falls folgende Bedingung erfüllt ist:*

$$\phi = (type_\phi, name_\phi, operator_\phi, value_\phi) \sqsubset_f^n \alpha = (type_\alpha, name_\alpha, value_\alpha)$$
$$\Leftrightarrow type_\alpha = type_\phi \wedge name_\alpha = name_\phi \wedge operator_\phi(value_\alpha, value_\phi).$$

Der Definition 2.3.1 kann entnommen werde, wie einzelne Attributfilter einer Subscription und Attribute einer Notification ausgewertet werden. Während die Attribute einer Notification als Menge zusammengefasst werden, gilt bei Subscriptions eine konjunktive Verknüpfung der einzelnen Attributfilter. Der Bezeichner S steht für die Menge aller möglichen Subscriptions, N für die Menge aller möglichen Notifications. Eine Subscription kann mehrere Attributfilter mit gleichem Namen und Datentyp beinhalten. Auch für diese gilt eine implizite konjunktive Verknüpfung. Ob in einer Notification ebenfalls Attribute gleichen Namens und Typ verwendet werden können, kann der Spezifikation nicht entnommen werden. Im Folgenden wird davon ausgegangen, dass dies nicht der Fall ist.

Definition 2.3.2 *Eine Subscription s bzw. ein Filter stimmt mit einer Notification n überein, kurz: $s \sqsubset_S^N n$, falls folgende Bedingung erfüllt ist:*

$$s \sqsubset_S^N n \Leftrightarrow \forall \phi \in s : \exists \alpha \in n : \phi \sqsubset_f^n \alpha.$$

Die von einer Subscription adressierte Menge an Notifications aus N wird mit N_s bezeichnet. Aus Definition 2.3.2 wird schließlich ersichtlich, dass für jeden Attributfilter einer Subscription ein passender Attributeintrag in der Notification vorhanden sein muss. Für den speziellen

string	objekt	>*	automobil/pkw
string	hersteller	=	mercedes
string	währung	=	euro
integer	ps	>	169
float	preis	<	40000
year	baujahr	=	2008

Abbildung 2.4: Beispiel eines Filters

Operator *any* wird kein zusätzlicher Wert benötigt. Wird dieser Operator in einem Attributfilter verwendet, trifft dieser Filter auf jedes Attribut, dessen *name* und *type* mit dem des Attributfilters übereinstimmt, zu. Damit kann in einer Subscription die Existenz eines bestimmten Attributs in einer Notification vorausgesetzt werden, ohne einen bestimmten Wertebereich spezifizieren zu müssen. In der Spezifikation wird kein Lösungsweg für eine disjunktive Verknüpfung der Attributfilter innerhalb einer Subscription angegeben. Es kann jedoch davon ausgegangen werden, dass solche disjunktive Verknüpfungen durch separate Subscriptions realisiert werden.
In Abbildung 2.4 ist beispielhaft eine mögliche Subscription abgebildet. Der Sender dieser Subscription ist offensichtlich an neueren Personenkraftwagen gehobener Mittelklasse der Marke Mercedes interessiert. Kann davon ausgegangen werden, dass der Sender diese Subscription vor dem Versand der Notification aus Abbildung 2.3 angemeldet hat, wird ihm die Notification vom System zugestellt. Alle Attribute der Notification treffen gemäß Definition 2.3.1 auf die Attributfilter der Subscription zu. Wie man sieht, darf eine Notification durchaus weitere Attribute beinhalten, die nicht durch die einzelnen Attributfilter der Subscription abgedeckt werden, ohne die Bedingung für das Matching zu verletzen.

Advertisements Eine Erweiterung des klassischen Publish/Subscribe-Paradigmas stellt das Konzept der Advertisements dar. Siena behandelt auch dieses Konzept, ist darauf jedoch nicht angewiesen. In [CRW99] wird explizit darauf hingewiesen, dass der Service auch ohne Advertisements implementiert werden kann und diese nur eine Optimierung darstellen. In einem System ohne Advertisements wird einem potentiellen Empfänger eine Notification übermittelt, falls dieser eine passende Subscription angemeldet hat. Mit Advertisements wird einem Empfänger nur dann eine Notification zugestellt, falls dieser eine passende Subscription angemeldet hat und der Veröffentlicher der Notification vorab ein zur Notification passendes Advertisement gesendet hat.
Ein Veröffentlicher verwendet Advertisements, um dem System die Art von Notifications bekannt zu machen, welche er in Zukunft versenden wird. Hier wird natürlich davon ausgegangen, dass der Sender vorab eine gewisse Kenntnis über die Art der Events besitzt, welche sich in Zukunft ereignen werden und für welche er Notifications versenden wird. Ähnlich der Menge N_s für Subscriptions bezieht sich ein Advertisement auf eine Menge N_a an Notifications. Sind dem System vorab die Art der Notifications bekannt, die ein Veröffentlicher sendet, können Subscriptions gezielter zu den Quellen der passenden Notifications geroutet werden. Subscriptions werden dann nur zu Veröffentlichern weitergeleitet, die mindestens ein Advertisement angemeldet haben, dessen Schnittmenge N_a mit der Menge N_s der jeweiligen Subscription eine nicht-leere Menge ergibt. Alle anderen Pfade zu Veröffentlichern können ausgelassen werden. Ohne die Verwendung von Advertisements müssen Subscriptions potentiell zu jedem Veröffentlicher des Netzes gesendet werden, da nicht bekannt ist, welcher Veröffentlicher im System zu den Subscriptions passende Notifications erzeugen wird. Leider ist in den Literaturquellen [CRW99] und [CRW00] zu Siena keine Definition bezüglich eines

konkreten Coverings von Advertisements und Subscriptions zu finden. Ein Covering von Advertisements und Subscriptions ist ausschließlich über die nicht-leere Schnittmenge von N_a und N_s definiert worden.
Für das Beispiel aus Abbildung 2.3 könnte der Veröffentlicher der Notification vorab ein Advertisement am System absetzen um anzukündigen, dass er weitere neue Fahrzeugmodelle veröffentlichen will. Ein solches Advertisement ist optional. Entscheidet sich der Veröffentlicher für ein solches Advertisement, sollten die folgenden Notifications auch zu diesem passen. Ansonsten ist die Wahrscheinlichkeit hoch, dass bestimmte Notifications nicht an alle Empfängern ausgeliefert werden.

Ähnlich den Subscriptions sind auch die Advertisements jeweils eine Menge von Attributfiltern. Für die Menge aller möglichen Advertisements wird im Folgenden der Bezeichner A verwendet.

Definition 2.3.3 *Für ein Advertisement $a \in A$ gilt die folgend angegebene Relation $a \sqsubset_A^N n$ bezüglich der zu a passenden Notifications:*

$$a \sqsubset_A^N n \Leftrightarrow \forall \alpha \in n : \exists \phi \in a : \phi \sqsubset_f^n \alpha.$$

Die Darstellung der einzelnen Attributfilter und die Relation $\sqsubset_f^n$ sind äquivalent zu den Subscriptions. Advertisements unterscheiden sich jedoch von Subscriptions in der aus Definition 2.3.3 zu entnehmenden disjunktiven Verknüpfung der einzelnen Attributfilter. In einem Advertisement können mehrere Attributfilter gleichen Namens und Typs enthalten sein, von denen nur einer zum betreffenden Attribut der Notification passen muss. Für alle Attribute einer Notification muss jedoch jeweils ein Attributfilter im Advertisement vorhanden sein. Die Bedingung aus Definition 2.3.3 gilt für alle Elemente aus N_a eines Advertisements $a \in A$.

Unsubscriptions/Unadvertisements Dem Client des Systems wird über die allgemeine Siena API die Möglichkeit geboten, Subscriptions und Advertisements, die er zuvor am System angemeldet hat, wieder abzumelden. Der Client wird diese Funktionen verwenden, falls er an bestimmten Notifications kein Interesse mehr zeigt oder falls er bestimmte Events im System nicht mehr veröffentlichen wird. Ohne diese Funktionen würde das System sonst im Laufe der Zeit unnötigen Nachrichtenaufwand erzeugen.

Unadvertisements und Unsubscriptions werden – wie auch Advertisements und Subscriptions – durch Filter repräsentiert. Diese Filter beziehen sich jedoch im gegebenen Fall nicht mehr auf Notifications, sondern auf andere Filter. Dieser Sachverhalt wird mit den binären Covering-Relationen $a_{un} \sqsubset_A^A a$ und $s_{un} \sqsubset_S^S s$ bezeichnet.

Definition 2.3.4 *Ein Filter f_1 überdeckt oder covert einen Filter f_2, falls folgende Bedingung erfüllt ist:*

$$f_1 \sqsubset f_2 \Leftrightarrow \forall n \in N : f_1 \sqsubset n \Rightarrow f_2 \sqsubset n.$$

Für die Anwendung der Bedingung aus Definition 2.3.4 auf Unadvertisements und Unsubscriptions gelten folgende Ersetzungsregeln:

- Unadvertisement:
  - f_1 wird durch ein Unadvertisement ersetzt
  - Die vom Äquivalenzpfeil links stehende Relation wird durch $\sqsubset_A^A$ ersetzt
  - Die rechts stehenden Relationen werden durch $\sqsubset_A^N$ ersetzt
  - f_2 wird durch ein Advertisement ersetzt
- Unsubscription:

publish(notification n)

subscribe(string id, pattern expression)
unsubscribe(string id, pattern expression)

advertise(string id, filter expression)
unadvertise(string id, filter expression)

Abbildung 2.5: Siena API aus [CRW99]

- f_1 wird durch eine Unsubscription ersetzt
- Die vom Äquivalenzpfeil links stehende Relation wird durch $\sqsubset_S^S$ ersetzt
- Die rechts stehenden Relationen werden durch $\sqsubset_S^N$ ersetzt
- f_2 wird durch eine Subscription ersetzt

Ein Unadvertisement bzw. eine Unsubscription adressiert und entfernt schließlich alle Advertisements bzw. Subscriptions, auf welche die Bedingung 2.3.4 mit den genannten Ersetzungsregeln zutrifft. Aus den Formeln lässt sich weiterhin ableiten, dass die Unadvertisements bzw. Unsubscriptions zu mehr Notifications passen können, als die zugehörigen Advertisements bzw. die Subscriptions, die vom System abgemeldet werden sollen. Bei einer Unsubscription bzw. einem Unadvertisement ist demnach zu beachten, dass alle überdeckten Advertisements bzw. Subscriptions des Client gelöscht werden. Durch eine Unsubscription könnten somit durchaus mehrere angemeldete Subscriptions eines Clients entfernt werden. Für die Abmeldungen müssen natürlich noch die IDs der Unadvertisements und Advertisements bzw. Unsubscriptions und Subscriptions übereinstimmen, wie der im Folgenden beschriebenen API zu entnehmen ist.

API Die Siena API erweitert die ursprüngliche Publish/Subscribe API um die bisher genannten Konzepte, wie z.B. den Advertisements. Abbildung 2.5 zeigt die API von Siena auf. Neu ist hier die Funktion zur Anmeldung von Advertisements, die ID-Parameter der Funktionen und die Funktionen für die Abmeldung der Advertisements und Subscriptions. Die zusätzlichen IDs dienen der Identifikation des Clients, der in der Rolle des Veröffentlichers Advertisements und als Interessenvertreter Subscriptions anmeldet. Die einzige Restriktion des Systems bezüglich der gewählten IDs ist die Eindeutigkeit. Die IDs dienen dazu, dass ein Client durch Unsubscriptions und Unadvertisements nur die durch ihn veröffentlichte Advertisements und Subscriptions vom System abmelden kann. Inwieweit Sicherheitsvorkehrung gegen Angriffe wie zum Beispiel *Masquerading/Spoofing*[14] getroffen wurden, ist der Spezifikation nicht zu entnehmen.
Zu erkennen ist noch, dass für Advertisements Filter verwendet werden, für Subscriptions jedoch *Patterns*. Patterns sind generischer als Filter. Ein Pattern kann sich gleichzeitig auf ein oder mehrere Notifications beziehen, welche sich z.B. auch in ihrem zeitlichen Auftreten unterscheiden können. Ein anderer Begriff hierfür sind die *Compound Subscriptions*. Hierdurch wird die Umsetzung von Relationen zwischen den Notifications, wie z.B. n_1 *and* n_2 möglich. Wie diese Patterns im System aufgelöst, ausgewertet und verteilt werden, ist für diese Arbeit bedeutungslos. Es wird davon ausgegangen, dass die Verwendung von Filtern den generellen Anforderungen genügt. Weitere Informationen zu Patterns im Zusammenhang von Publish/Subscribe-Systemen können in [CRW99] nachgelesen werden.

[14] Verschleierung der eigenen Identität

Topologie Siena selbst stellt keine Funktionalität zum Aufbau physischer Verbindungen oder gar ganzer Netzwerke zur Verfügung. Es verwendet ein Overlay auf Basis bestehender Vermittlungs- bzw. Transportprotokolle, um ein eigenes Routing definieren zu können. In der Spezifikation zu Siena wird zwischen zwei Implementierungsansätzen unterschieden: Der eine baut auf einer hierarchischen Netzstruktur auf, der andere auf einem Netz, dessen Topologie einem ungerichteten Graphen entspricht. In beiden Fällen ist eine bidirektionale Kommunikation zwischen den Knoten möglich. Die Verfahren für den ungerichteten Graphen können auch in einem hierarchisch angeordnetem Netz oder allgemein azyklischer Topologie angewandt werden. Topologien, die einem ungerichteten Graphen entsprechen, sind häufiger anzufinden und ausfallsicherer als hierarchische Netze. Die Skalierbarkeit hierarchischer Netze ist ebenfalls potentiell schlechter, da wurzelnahe Knoten meist stark belastet werden und dadurch eine ungleiche Lastverteilung entsteht. Aus diesen Gründen wird hier nicht näher auf den hierarchischen Ansatz eingegangen. Für Siena selbst ist nur wichtig, dass eine Möglichkeit des Nachrichtenversands zwischen den Servern besteht, egal welches Protokoll für das Netz oder den Nachrichtenaustausch verwendet wird. Clients des Systems verwenden die in Abbildung 2.5 beschriebene API, um mit dem System zu kommunizieren, sind also nicht Teil der Serverstruktur und somit nicht am Routing beteiligt.

Die von der Peer-to-Peer-Architektur von Siena im Allgemeinen verwendeten zyklischen Netze bergen das Problem in sich, dass mehrere Verbindungen zwischen zwei Knoten existieren können. Dadurch entstehen Zyklen, welche durch die Routingstruktur eliminiert werden müssen. Dazu verwendet man häufig Spannbäume, die z.B. durch Anwendung des Dijkstra-Algorithmus[15] [Tan03] konstruiert werden können. Die redundanten Verbindungen sind jedoch gleichzeitig der Grund für die erhöhte Robustheit und Flexibilität des Netzes, da einzelne Verbindungsabbrüche in Kauf genommen werden können. Die Administration des Netzes hat grundsätzlich nur dafür Sorge zu tragen, dass mindestens eine logische Verbindung zwischen jedem Server des Systems besteht.

Routing Basierend auf der konkret festgelegten Serverstruktur muss ein Publish/Subscribe-System Pfade zwischen den einzelnen Servern aufbauen, damit Notifications an Empfänger weitergeleitet werden können. Ein mögliches Verfahren ist das Broadcasting aller Notifications im gesamten Netz. Dies kann mit Hilfe von Spannbäumen realisiert werden. Zum Aufbau dieser Spannbäume existieren bereits eine Menge von Algorithmen, weshalb dies in der Spezifikation zu Siena nicht weiter beschrieben wird. Beim Broadcasting erhält zwar jeder Knoten des Netzes alle Notifications und kann diese gegebenenfalls an seine Clients zustellen, der Nachrichtenaufwand ist jedoch enorm. Damit eine Notification nur an die Interessenvertreter zugestellt wird, müssen sich die Subscriptions und Notifications an einer Position im Netz treffen und gegeneinander ausgewertet werden. Die Speicherung der Subscriptions bei den Zugangsknoten der Clients bietet im ersten Moment keine Verbesserung im Gegensatz zur Broadcast-Methode. Kann jedoch davon ausgegangen werden, dass Subscriptions in ihrer Anzahl weniger häufig vorkommen wie Notifications, so stellt die Speicherung der Subscriptions bei allen Zugangsknoten des Netzes eine Verbesserung des entstehenden Nachrichtenaufwands dar. Die Subscriptions werden also an alle Quellen der Notifications verteilt.
Sobald eine Notification abgeschickt wird, kann durch den ersten Server, der die Notification erhält, sofort ein Matching mit den verfügbaren Subscriptions durchgeführt und die Notification gegebenenfalls weitergeleitet werden. Für die weitere Verteilung der Notifications werden Prinzipien des IP-Multicast verwendet. Dabei wird versucht die kürzeste Route zu einem Empfänger zu wählen, die Nachricht selbst aber so spät wie möglich zu replizieren

[15] Algorithmus zur Berechnung kürzester Wege in einem Graphen

und an die einzelnen Empfänger zu senden, um den Nachrichtenaufwand gering zu halten.

Eine zusätzliche Routingoptimierung in Siena basiert auf der Erkenntnis, dass sich Subscriptions im Allgemeinen sehr ähneln. Falls also ein Server beispielsweise zwei Subscriptions s_1 und s_2 erhält, von denen der Filter von s_1 den Filter von s_2 überdeckt, muss nur noch die Subscription s_1 an die Zugangsknoten des entsprechenden Spannbaums übermittelt werden. In den folgenden Notifications, die von s_1 getroffen und daher auch in entgegengesetzter Richtung des Spannbaums an den Server übermittelt werden, sind auch die Notifications enthalten, welche potentiell von s_2 getroffen werden, da $N_{s_2} \subset N_{s_1}$ gilt. Diese Optimierung, sowie ähnliche Verfahren der Aggregation wie zum Beispiel das *Merging*[16] [MFB02], finden im Schwerpunkt dieser Arbeit aus in den folgenden Kapiteln genannten Gründen keine intensive Betrachtung. Darum werden die Auswirkungen auf das Routing hier nicht weiter besprochen.

Fakt ist jedoch, dass diese Optimierungen einige nicht einfach zu lösende Probleme mit sich bringen. Die Verwendung des Coverings im Beispiel von Siena verletzt beispielsweise die in [CRW99] genannte Behauptung, dass eine Notification höchstens ein Mal zugestellt wird. Zu einer mehrfachen Zustellung der Notification kann es kommen, wenn ein Knoten in den Spannbäumen der überdeckten und überdeckenden Subscription über zwei unterschiedliche Pfade erreichbar ist. Eine Notification, die zu der überdeckenden Subscription passt, wird dann zum Sender der überdeckten Subscription über zwei unterschiedliche Pfade zugestellt. Natürlich können solche Duplikate durch ein Filtering beim Client erkannt und verworfen werden, erzeugen aber trotz alledem einen zusätzlichen Nachrichtenaufwand.

Die primären Operationen, die ein Server beim Routing durchführen muss sind laut [CRW00] die Berechnung von Covering- bzw. Matching-Relationen und die Verwaltung der Routingtabelle für Subscriptions und Advertisements, welche sich widerrum auf die Berechnung von Covering-Relationen zurückführen lässt. Für die Subscriptions und Advertisements werden separate Routingstrukturen verwaltet, in der eine Halbordnung auf die Filter erzeugt wird. Dadurch lassen sich Covering-Relationen sehr einfach darstellen. Es werden dann nur Subscriptions, bzw. Advertisements weitergeleitet, welche nicht von einer anderen Subscription, bzw. einem anderen Advertisement überdeckt werden. Dazu sind Auswertungen der Relationen $\sqsubset_S^S$, $\sqsubset_A^A$ und $\sqsubset_A^S$ nötig. Für zwei beliebige Filter liegt die Komplexität der Auswertung einer dieser Relationen in $O(n * m)$, wobei n und m die Anzahl der Attributfilter der einzelnen Filter bezeichnen. Jeder Attributfilter eines Filters muss mit jedem Attributfilter des anderen Filters verglichen werden. Für den Vergleich an sich wird, basierend auf den primitiven Datentypen von Siena, von einer Komplexität in $O(1)$ ausgegangen. Für die Entscheidung über eine Weiterleitung einer Notification zu einem bestimmten Nachbarn des Servers müssen die Relationen $\sqsubset_S^N$ und $\sqsubset_A^N$ ausgewertet werden. Dazu müssen die Attributfilter auf die jeweiligen Attribute der Notifications ausgewertet werden.

Bei einer Komplexität von $O(1)$ für die Auswertung eines Attributfilters ergibt sich insgesamt eine Komplexität von $O(n + m)$. Da diese Komplexität wesentlich geringer ist als die des Coverings, konzentrieren sich die Entwickler von Siena hauptsächlich auf die Reduzierung der Komplexität des Coverings. Eine spezielle Routingtabellenstruktur für die Auswertung von Matching-Relationen ist nicht gegeben. Die Komplexität der Auswertung des Matchings liegt also insgesamt in $O(R(n+m))$, wobei hier R die Kardinalität der Routingtabelle angibt. Diese kann durch die ausschließliche Betrachtung z.B. von Subscriptions, die alle anderen Subscriptions überdecken, bei der Auswertung und der Sortierung von Attributfiltern und Attributen weiter reduziert werden. Für eine intensivere Erläuterung einer optimalen Routingtabellenstruktur für das Matching wird auf die Arbeiten [CW03] und [LHJ05] verwiesen. Da das dort geschilderte Konzept für das im Schwerpunkt dieser Arbeit behandelte System nicht verwendet werden kann, wird an dieser Stelle nicht näher darauf eingegangen.

[16] Zusammenfassen

An dieser Stelle soll noch einmal kurz auf die Behandlung der bereits erläuterten Unsubscriptions und Unadvertisements eingegangen werden. Unsubscriptions müssen prinzipiell durch den gleichen Spannbaum im System verteilt werden wie die vorangegangenen Subscriptions des Clients. Das gleiche gilt für die Verteilung der Unadvertisements, welche auch eine ähnliche Struktur aufweisen. Spezielle Behandlungen werden durch die Einführung des Coverings notwendig. So müssen z.B. bei der Entfernung einer Root-Subscription alle Subscriptions veröffentlicht werden, welche durch das Covering der Root-Subscription vorab nicht verteilt wurden. Werden hingegen überdeckte Subscriptions abgemeldet, so muss die Unsubscription eventuell weiter geroutet werden, da sie den Spannbaum der Root-Subscription einschränkt. Unadvertisements vergrößern möglicherweise den Spannbaum, der durch die zugehörigen Advertisements zuvor abgeschnitten wurde. Alle Subscriptions, die zuvor durch die zugehörigen Advertisements nicht weiter verteilt wurden, da von den Veröffentlichern der Advertisements keine passenden Notifications angekündigt wurden, müssen nach einem Unadvertisement eventuell weiter im Netz propagiert werden. Unsubscriptions und Unadvertisements müssen also trotz einer ähnlichen Struktur aus semantischen Gründen unterschiedlich behandelt werden.

2.3.3 Historische Notifications

Padres ist das nach eigenen Aussagen aus [LCH+07] bisher einzige verteilte Inhalt-bezogene Publish/Subscribe-System, welches die Möglichkeit anbietet, Subscriptions für historische Notifications am System anzumelden. Es kann daher in Bezug auf das Thema dieser Arbeit als einzige verwandte Arbeit bezeichnet werden. Padres basiert grundsätzlich auf den Konzepten von Siena. Zusätzlich wurden Optimierungen wie z.B. das Merging aus dem Publish/Subscribe-System „Rebeca“ übernommen. In [Zei04] ist eine Erweiterung des auf Rebeca basierenden Subscription-Verfahrens beschrieben, welche ebenfalls die Möglichkeit anbietet, Subscriptions für vergangene Notifications anzumelden. Die Intention des Verfahrens ist allerdings eine völlig andere als bei Padres. Durch ein spezielles Puffern von Notifications wird versucht, die Bootstrap-Zeiten der mobilen Clients des Systems zu mindern, besonders beim *Roaming*[17]. Obwohl die Verfahren mit ähnlichen Problemen wie die des Schwerpunkts dieser Arbeit konfrontiert werden, wie z.B. den Nachrichtenduplikaten, ist die Zielsetzung dennoch eine völlig andere.

Unter historischen Notifications sind Notifications zu verstehen, welche schon vor der Anmeldung einer Subscription im Netz verteilt wurden. Diese können in üblichen Publish/Subscribe-Systemen im Nachhinein nicht mehr zugestellt werden, falls sie nicht an irgendeiner Stelle abgespeichert wurden. Eine Client-seitige Speicherung ist sehr unflexibel und stellt in einem verteilten System keinen guten Lösungsansatz dar. Daher wurde eine Art verteilter Datenbankansatz gewählt, welcher als zusätzliche Schicht über dem Publish/Subscribe-Overlay angesehen werden kann. Die einzelnen Datenbanken sind bei den Brokern des Systems angesiedelt und für die Speicherung von Notifications zuständig. Der Prozess der Speicherung und Wiedergewinnung von historischen Notifications soll dabei dem Publish/Subscribe-Paradigma folgen. Im Gegensatz zu verteilten Datenbanken liegen die Zugangspunkte zur Datenbank nicht nur an einer Stelle, sondern bei jedem Broker, der eine Datenbank mit den notwendigen Daten verwaltet. Die verschiedenen historischen Notifications können dann auf mehrere Datenbanken aufgeteilt werden.

Padres bietet nicht nur die Möglichkeit für historische, sondern auch zukunftbezogene und hybride Subscriptions. Zukunftbezogene Subscriptions sind die bisher bekannten Subscrip-

[17] Mit Roaming ist die Fähigkeit eines Benutzers gemeint, sich von einem Netzwerk in ein anderes bewegen zu können, ohne sich explizit an- oder abmelden zu müssen oder einem Bruch der Serviceleistungen zu unterliegen

tions. Sie gelten im Regelfall für Notifications, die nach der Verbreitung der Subscription übermittelt werden.
Eine Relation, ähnlich wie bei Siena die Verwendung von Patterns, zwischen hybriden Subscriptions kann durch *Composite Subscriptions* [LJ05] hergestellt werden. Hybride Subscriptions werden bei der weiteren Behandlung durch das Routing in zukunftsbezogene und historische Subscriptions aufgeteilt. Historische Subscriptions enthalten Zeitintervalle, gekennzeichnet durch einen Start- und Endzeitpunkt, welche sich auf den Eintrittszeitpunkt der historischen Notifications beziehen. Dieser Eintrittszeitpunkt wird durch die Zugangspunkte unter Einsatz synchronisierter Uhren vergeben. Abweichungen dieser Uhren können durch Toleranzen bei der Auswertung der Operatoren beim Zeitvergleich durch das Matching ausgeglichen werden.

Die Prozedur bei der Behandlung historischer Notifications kann grob in folgende Schritte gegliedert werden:

1. Jeder Broker meldet für seine Datenbanken jeweils eine Subscription mit eindeutigem ID am System an.

2. Der Client sendet eine Notification mit einer Subscription für aufzuzeichnende Notifications an einen Datenbank-ID.

3. Der Datenbank-Broker meldet die vom Client erhaltene Subscription am System an.

4. Der Datenbank-Broker meldet gleichzeitig noch zu der erhaltenen Subscription äquivalente Advertisements an.

5. Erhält der Datenbank-Broker aufgrund der Subscription Notifications, so werden diese in äquivalente *Structured Query Language* (SQL) Insert-Statements umgewandelt und an die Datenbank übergeben.

6. Da die Clients die Lokationen der jeweiligen historischen Notifications, die sie erhalten möchten, nicht kennen, melden sie historische Subscriptions an.

7. Erhält der Datenbank-Broker eine Subscription für in der Datenbank gespeicherte Notifications, welche mittels eines SQL-Query abgefragt werden, so werden diese über das Overlay direkt an den Client gesendet. Selbst bei einer Überlappung von historischen Notifications verschiedener Clients wird keine Multicast-Übermittlung angestrebt.

Historische Subscriptions werden durch ein zusätzliches Attribut gekennzeichnet. Dies verhindert ein Matching von gegenwärtigen Notifications und historischen Subscriptions. Auf das Splitting und Merging bei der Behandlung von Composite Subscriptions wird hier nicht näher eingegangen. Die Vorgehensweise für das Splitting von hybriden Subscriptions konnte [LCH+07] nicht entnommen werden.

Einige Problembereiche bei der Realisierung eines Publish/Subscribe-Systems mit Zugriffsmöglichkeiten auf historische Daten sind noch Themen aktueller Forschung. Die Platzierung der Datenbanken im Overlay, die Aufteilungsmöglichkeiten der Notifications auf verschiedene Datenbanken sowie die Duplikatvermeidung bei der Zustellung historischer Notifications, sind beispielsweise einige Punkte, mit denen sich das Entwicklerteam von Padres noch beschäftigt. Der aktuelle Stand von Padres bedingt noch einige Administrationstätigkeiten, die prinzipiell auch durch das Systems selbst durchgeführt werden könnten. Diese Tätigkeitsfelder umfassen beispielsweise die manuelle Zuteilung der Datenbanken zu Brokern, die Konkretisierung der Subscriptions für die Datenbanken, die Auflösung des Schnittmengenproblems für Subscriptions unterschiedlicher Datenbanken und die aus Gründen der Fehlertoleranz notwendige redundante Speicherung von historischen Notifications.

2.4 Contextcast

Dieser Abschnitt stellt das noch recht junge Contextcast vor, welches das Basissystem für die in den folgenden Kapiteln dieser Arbeit besprochenen Verfahren bildet. Zu Beginn wird eine kurze Einführung in das Entwicklungsumfeld, die Entwicklungsziele, die Anwendungsszenarien und die daraus resultierenden Anforderungen, wie auch einigen Problemstellungen des Contextcast gegeben. Anschließend folgt eine Übersicht über die Komponenten eines Contextcast und deren Funktionalität. Der darauf folgende Unterabschnitt erklärt den Aufbau eines Kontextes und zieht einen Vergleich zum Publish/Subscribe. Nachfolgend wird der Aufbau von Nachrichten, ihre Beziehung zu Kontexten bezüglich des Routings und das Matching von Nachrichten auf Kontexte verdeutlicht. Der Unterabschnitt zur Topologie und zum Routing erläutert die dem Contextcast zugrunde liegende Topologie und die Nachrichtenübermittlung im Contextcast. Die Effizienz der Nachrichtenübermittlung wird durch den Vergleich zu einfacheren naiven Verfahren deutlich gemacht. Nach einer detaillierten Beschreibung des Routingalgorithmus wird auf die Behandlung von Spezialfällen, wie Topologieänderungen und Verbindungsabbrüche zwischen Komponenten des Contextcast, eingegangen. Mobile Clients nehmen durch ihre Besonderheit bezüglich Ressourcenverfügbarkeit und Mobilität eine gesonderte Stellung im Contextcast ein. Die Schilderung der Verfahren zur Unterstützung speziell mobiler Clients und eine kurze Zusammenfassung schließt den Abschnitt über das Contextcast ab.

Einführung Das Contextcast stellt eine neue Art der Kommunikation im Bereich *content-based adressing and routing* dar. Dieses neue Kommunikationsparadigma wurde erstmals in [Gei07] erwähnt. Es wird momentan im Sonderforschungsbereich Nexus des „Instituts für Parallele und Verteilte Systeme“ der Universität Stuttgart weiterentwickelt. Dieser Forschungsbereich beschäftigt sich mit kontextbezogenen Systemen.

Mit dem Begriff *Kontext* sind Eigenschaften eines Objekts gemeint, die seinen aktuellen Zustand möglichst genau charakterisieren [DA99]. Ein Objekt kann eine Person, eine Maschine oder auch eine Anwendung sein. Eine besondere Stellung nimmt in diesem Zusammenhang die Lokation eines Objekts ein. Sie beschreibt die gegenwärtige Position des Objekts. Diese ist von besonderer Wichtigkeit, da ein eindeutiges Ziel des Contextcast-Systems die Unterstützung mobiler Endgeräte ist, welche ihre Position sehr häufig wechseln. Durch eine Adressierung von Nachrichtenpaketen an geographische Lokationen erschließen sich eine Menge neuer Anwendungsszenarien. Ein Beispiel hierzu sind Staumeldungen an Fahrzeugführer. Fahrzeuge, welche sich zum Beispiel auf Autobahnen befinden, bewegen sich relativ schnell. Weiterhin kann grundsätzlich nicht vorhergesagt werden, an welcher Position sich ein Fahrzeug im zeitlichen Verlauf befinden wird. Eine zunehmende Anzahl der Fahrzeuge sind heutzutage mit GPS-Geräten ausgestattet. Macht ein Fahrzeug seinen Standort in regelmäßigen Abständen z.B. in Form von GPS-Koordinaten [Dep97] bekannt, so ermöglicht dies das gezielte Senden von Staumeldungen an Fahrzeuge, welche vom Stau betroffen sind. Die Staumeldung in Form einer Nachricht soll im Gegensatz zum IP-Unicast keine explizite Empfängeradressierung enthalten. Vielmehr soll eine Multicast-Nachricht mit einer Lokation als Empfängeradresse verwendet werden. Da meist mehrere Fahrzeuge von einer Staumeldung betroffen sind, reduziert die Multicast-Kommunikation den Nachrichtenaufwand im Netz.

Ein weiteres Problem bei der Unterstützung mobiler Endgeräte ergibt sich durch den häufigen Wechsel der Zugangsnetze durch Positionsänderungen, den beschränkt verfügbaren Ressourcen wie Stromversorgung, GPS-Geräte oder Rechenkapazität und der Verbindung zum Netz, welche meist durch störanfällige Funknetze realisiert wird. Aus diesen Gründen

send	Ermöglicht die Übermittlung von Nachrichten an das Contextcast-System.
announceContext	Erlaubt dem Benutzer die Anmeldung eines Kontextes.
revokeContext	Wird verwendet, um einen zuvor angemeldeten Kontext wieder aus dem System zu entfernen.
notify	Stellt die empfangenen Nachrichten aus dem Contextcast-System an die Anwendung zu.

Abbildung 2.6: Contextcast API aus [Gei07]

entsteht die Forderung nach weiteren Mechanismen wie z.B. dem *Caching*[18] von Nachrichten, falls zwischen dem Empfänger und dem Netz kurzzeitig keine Verbindung besteht. Der in [Gei07] beschriebene Ansatz für ein Contextcast unterstützt mobile Endgeräte, ermöglicht die Adressierung durch Lokationsangaben ähnlich dem GeoCast [NI97] und erweitert dieses Adressierungsschema um weitere Attribute, wie sie aus dem Publish/Subscribe-Umfeld bereits bekannt sind. Die Lokation erweitert um objektspezifische Attribute ergibt dabei den Kontext eines Objekts im Contextcast.

Komponenten und API Im Bereich des Contextcast werden eigene Begriffe für die Komponenten und Informationsobjekte des Systems verwendet, um eine klare Abgrenzung zu anderen Systemen deutlich zu machen und damit einer möglichen Verwechslung vorzubeugen. Die Teile oder auch Komponenten des Systems sind in einer Overlay-Schicht angeordnet. Die Komponenten ähneln denen des GeoCast-Systems [NI97]. Es existieren ContextHosts, ContextNodes und ContextRouter. Der ContextHost ist eine Anwendung, die bei einem Teilnehmer eines Contextcast für die Kommunikation mit dem System zuständig ist. Sie übernimmt die Verwaltung der Kommunikation zwischen Anwendungen, die auf die Funktionalität des Contextcast zugreifen, und dem System selbst. Dazu wird eine API mit den in Abbildung 2.6 aufgezählten Funktionen zu Verfügung gestellt. ContextNodes stellen das Verbindungsglied zwischen ContextHost und dem Contextcast-System dar. Sie sind der Zugangsknoten für die Benutzer des Systems. Gleichzeitig fungieren sie auch als Router für Nachrichten und Kontexte. Für die weitere Verbreitung von Kontexten und Zustellung der Nachrichten sind die ContextRouter zuständig.

Die unterschiedlichen Komponenten eines Contextcast sind in einer beispielhaften Anordnung in Abbildung 2.7 dargestellt. Innerhalb des jeweiligen Zugangsbereichs, die ein ContextNode abdeckt, können sich die ContextHosts der Benutzer mit dem ContextNode und damit dem System verbinden. Diese Verbindungen sind als gestrichelte Linien dargestellt. Alle rot markierten Objekte stellen Komponenten des Overlays von Contextcast dar. Die normalen Router des Netzes befinden sich auf dem Underlay und verfügen daher nicht über den Anwendungscode des Contextcast. Sie leiten die Contextcast-Nachrichten lediglich weiter. ContextNodes können ebenso als ContextRouter fungieren und Nachrichten weiterleiten.

Für die Entwicklung eines Contextcast kann man sich durchaus von den Konzepten der Publish/Subscribe-Systeme inspirieren lassen. Einige dieser Konzepte können unverändert oder mit kleinen Änderungen übernommen werden, andere machen in einem Contextcast keinen Sinn. Der Grund liegt darin, dass ganz andere Anforderung auch bezüglich der Adressierung im Gegensatz zu Publish/Subscribe vorliegen. Deshalb soll auch hier eine klare Tren-

[18] Zwischenspeicherung

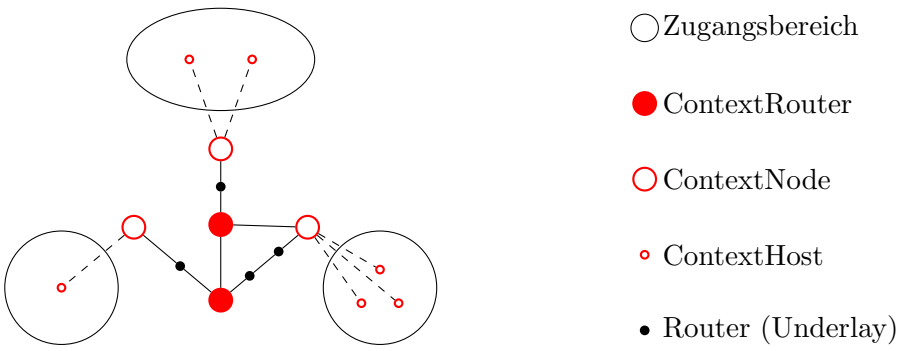


Abbildung 2.7: Komponenten des Contextcast

nung der Begrifflichkeiten erfolgen. Die informationstragenden Elemente eines Contextcast werden Nachrichten genannt, die Informationen über Benutzer des Systems Kontexte. Diese Elemente beinhalten eine gänzlich andere Semantik als Notifications und Subscriptions eines Publish/Subscribe-Systems.

Kontexte Wie schon erwähnt wurde, informiert ein Benutzer das Contextcast über seine spezifischen Eigenschaften in Form von Kontexten, die er am System anmeldet. Sie bilden die Basis für ein anschließendes gezieltes Routing von Nachrichten. Der Kontext enthält neben den verschiedensten Attributen einen global eindeutigen ID für den Kontext selbst, sowie einen ID für den ContextNode, an dem der Benutzer angemeldet und über den er erreichbar ist. Der global eindeutige ID des Kontextes dient zum Beispiel der eindeutigen Identifikation des Kontextes bei der Abmeldung. Der ID für den ContextNode gibt die aktuelle Position des Benutzers im System bekannt. Gerade kleine mobile Endgeräte verfügen nicht notwendigerweise über die Ausstattung wie z.B. GPS-Empfänger, die eine eigenständige Lokalisierung ermöglichen. Unter diesen Umständen macht man sich die Position des Zugangsnetzes zu nutzen. Bei kabelgebundenen wie auch bei funkbasierten Netzen kann der Zugangsbereich des Netzes meist abgeschätzt werden. Diese Lokation kann für den ContextNode des Zugangsnetzes verwendet werden, welche wiederum eine ungefähre Position bzw. einen Bereich für die an ihm angemeldeten Benutzer angibt. Die Verwendung des Zugangsbereichs für die Lokation der Benutzer verringert gleichzeitig die Notwendigkeit häufiger Änderungen der Kontexte aufgrund von Bewegungen der Benutzer. Diese Idee stammt von Geocast.

Eine Erweiterung dieses Systems, wie sie in [DR03] beschrieben ist, ermöglicht zudem die Verwendung von symbolischen Bezeichnern für Lokationen. Weiterhin wird erläutert, wie diese symbolischen Bezeichner für ein Lokationsmatching in Koordinaten umgewandelt werden können und vice versa. Der ID des ContextNode entspricht einem solchen symbolischen Link für dessen Zugangsbereich. Für die koordinatenbasierte Darstellung der Lokation werden meist Polygone verwendet. Um einen Empfänger der Nachricht beim Routing ausmachen zu können, muss das Polygon der Nachricht mit dem des Empfängerkontextes geschnitten werden. Wenn sich hieraus ein nicht-leerer Schnitt ergibt, kann die Nachricht an den Empfänger zugestellt werden. Wie in [NI97] erwähnt ist die Berechnung dieses Schnittes sehr rechenintensiv und muss bei jedem Router auf dem Pfad zu den Empfängern durchgeführt werden. Aus diesem Grund ist geplant auf ein System der Nexus Federation zurückzugreifen, welches aus dem Polygon der Nachricht und den Lokationsinformationen der ContextNodes auf Senderseite die ContextNode-Empfänger einer Nachricht berechnet. Dieser Schritt wird nur ein Mal pro Nachricht durchgeführt, da anschließend die Liste der ContextNodes als Empfänger der Nachricht in den Nachrichtenheader aufgenommen wird. Die Nachricht selbst wird

ähnlich dem Xcast an die Empfänger geroutet. Im Gegensatz zum herkömmlichen Xcast muss bei jedem ContextRouter und ContextNode ein Matching der Attributfilter der Nachricht auf die Attribute der Kontexte durchgeführt werden. Durch die explizite Angabe der ContextNode-Empfänger im Header der Nachricht wird das rechenintensive Matching der Lokationen von Nachrichten und Kontexten bei jedem Knoten auf dem Pfad zum Empfänger, sowie eine Prüfung der Reverse-Path-Forwarding-Bedingung vermieden. Das Matching entspricht in dieser Hinsicht der Prüfung, ob der ContextNode-ID in der Empfängerliste der Nachricht enthalten ist. Die endgültige Prüfung, ob die Lokation eines Benutzers mit der Lokation der Nachricht einen nicht leeren Schnitt ergibt bzw. ob bei einer Punktdarstellung die Position des Benutzers in dem Lokationsbereich der Nachricht enthalten ist, wird vom ContextHost auf Empfängerseite durchgeführt. Dies kann jedoch nur geschehen, wenn der Benutzer seine exakte Position kennt. Die Nachricht wird anschließend an die jeweilige Anwendung übergeben, vorausgesetzt die Prüfung liefert ein positives Resultat.

Die Kontexte der Benutzer enthalten eine Attributliste, welche mit der von Notifications eines Publish/Subscribe-Systems vergleichbar sind. Wie auch bei den Notifications besteht zwischen diesen Attributen keine logische Verbindung. Sie kann also als einfache Menge von Attributen angesehen werden. Zu beachten ist jedoch, dass hier explizit die Einmaligkeit eines Attributs in einem Kontext gefordert wird. Prinzipiell kann ein Benutzer mehrere Kontexte am System anmelden und damit das Spektrum der potentiell bei ihm eingehenden Nachricht erweitern. Der Aufbau der Attribute selbst ist der gleiche wie bei den Publish/Subscribe-Systemen, z.B. Rebeca. Der Unterschied in der Semantik von Kontexten zu Subscriptions ist der, dass in einem Contextcast der Nachrichtensender bestimmt, wer die Empfänger der Nachricht sind. Die potentiellen Empfänger, also die Benutzer, unterstützen die Entscheidung, indem sie Informationen in Form von Attributlisten über sich selbst preisgeben. Die Menge aller Kontexte wird mit $\mathbb{K}$, die Menge aller Attribute mit A bezeichnet.

Nachrichten und Matching Nachrichten stellen das Pendant zu Notifications in Publish/Subscribe-Systemen dar. Sie sind die informationstragenden Elemente des Contextcast. Im Gegensatz zu Notifications enthalten sie jedoch eine Attributfilterliste, die ähnlich denen in Subscriptions aufgebaut ist. Ebenfalls wird von einer konjunktiven Verknüpfung der Attributfilter ausgegangen, diesmal jedoch auf Nachrichtenseite. Der Nachrichtensender wählt die Empfänger mit Hilfe der Attributfilter aus. Dies entspricht dem Adressierungsschema klassischer Kommunikationsparadigmen wie Unicast in der Hinsicht, dass der Sender einer Nachricht die Empfänger angibt und nicht wie bei Publish/Subscribe-Systemen, bei denen der potentielle Empfänger auswählt, welche Nachrichten er erhalten möchte. Neben der Attributliste enthält eine Nachricht im Contextcast noch die Ziellokation der Nachricht und einen *Payload*[19]. Contextcast bezieht beim Routing nicht den kompletten Nachrichteninhalt für das Matching auf Kontexte mit ein. Es wird davon ausgegangen, dass der eigentliche Nachrichteninhalt eine beliebige Formatierung und Semantik besitzen kann und sich daher nicht für ein fest definiertes Matching eignet. Die Ziellokation dient der späteren Auswahl der Empfänger, deren Lokation sich mit der in der Nachricht schneidet. Diese müssen natürlich noch einen passenden Kontext angemeldet haben, damit sie die Nachricht auch wirklich zugestellt bekommen. Der aktuelle Stand von Contextcast verwendet Rechtecke für die Darstellung der Lokation. Eine Verwendung von Polygonen oder 2,5-dimensionalen Objekten ist in Planung.

Die Attributfilter sind wie bei Publish/Subscribe aufgebaut: Sie enthalten einen Attributnamen, einen Datentyp, einen Operator und einen Wert. Attributfilter gleichen Namens und gleichen Typs können mehrmals in der Nachricht vorkommen. Für sie gilt wie für den Rest

[19] Hier: der Nachrichteninhalt

der Attributfilter eine logische konjunktive Verknüpfung. Eine disjunkte Verknüpfung kann auch hier durch das Erzeugen weiterer Nachricht mit gleichem Nachrichteninhalt realisiert werden. Diese besitzen, bis auf die disjunktiv verknüpften Attributfilter, exakt die gleichen Attributfilter wie die ursprüngliche Nachricht besitzen. Die disjunktiv verknüpften Attributfilter werden auf die einzelnen Nachrichtenkopien aufgeteilt. Die Zuordnung der Attributfilter zu den Attributen eines Kontextes erfolgt über den Namen und den Datentyp des Filters. Die Menge der Attributfilter wird mit Φ, die Menge der Nachrichten mit $\mathbb{N}$ bezeichnet.

Definition 2.4.1 *Das Matching von Attributfiltern auf Attributen wird mit AFmatch bezeichnet. AFmatch ist eine boolesche Funktion, welche die Abbildung* $\Phi \times A \longrightarrow \{true, false\}$ *realisiert. Die Funktion* $AFmatch(\phi, \alpha)$ *mit* $\phi \in \Phi, \alpha \in A$ *wird zu true ausgewertet, falls die Bedingung aus Definition 2.3.1 auf* ϕ *und* α *zutrifft, andernfalls zu false.*

Definition 2.4.2 *Das Matching von Nachrichten auf Kontexte wird mit Match bezeichnet. Match ist ebenfalls eine boolesche Funktion, realisiert jedoch die Abbildung* $\mathbb{N} \times \mathbb{K} \longrightarrow \{true, false\}$*. Die Liste der Empfänger einer Nachricht* $n \in \mathbb{N}$ *wird mit* L_n *bezeichnet, der ID des ContextNode eines Kontextes* $k \in \mathbb{K}$ *mit* D_k*. Die Funktion Match ist wie folgt definiert:*

$$Match(n,k) = \begin{cases} true & falls\ D_k \in L_n \wedge \forall \phi \in (n \cap \Phi) \exists \alpha \in (k \cap A) : AFmatch(\phi, \alpha) \\ false & sonst. \end{cases}$$

In den Definitionen 2.4.1 und 2.4.2 sind die Funktionen für das Matching ähnlich dem Matching in Publish/Subscribe-Systemen angegeben. Die Datentypen und damit verbundenen möglichen Operatoren müssen bei der Implementierung des Systems fest definiert werden. Für die Operatoren müssen zusätzlich konkrete Auswertungsregeln festgelegt sein. Hintergrund ist, dass alle ContextRouter und ContextNodes des Systems in der Lage sein müssen ein Matching durchzuführen. Eine nachträgliche Änderung der verfügbaren Datentypen und Operatoren oder der Auswertungsregeln würde eine Änderung der Algorithmen für das Matching bei jedem ContextRouter und ContextNode des Systems bedingen.
Der Semantik der *Match*-Funktion ist zu entnehmen, dass ein Kontext für ein Matching mit einer Nachricht zumindest alle Attribute enthalten muss, für die Attributfilter in der Nachricht definiert sind. Dies muss vor allem auch dann beachtet werden, wenn ein Benutzer mehrere Kontexte anmelden oder einen bestehenden Kontext in mehrere Kontexte auftrennen möchte. Die einzelnen Kontexte sollten nach einer Trennung bezüglich ihrer Attributmenge so vollständig wie möglich sein.
Im Contextcast werden keine genauen Datentypen für eine Implementierung vorgeschrieben. Eine Empfehlung seitens des Autors von [Gei07] ist die Verwendung von Datentypen und Operatoren, wie sie auch in Publish/Subscribe-Systemen vorgegeben sind. Für die Lokationsangaben wird die Verwendung von Rechtecken vorgeschlagen. Jeder Eckpunkt dieses Rechtecks wird durch eine Koordinate in Gleitkommadarstellung beschrieben. Die Reihenfolge bei der Angabe der Eckpunkte muss festgelegt sein, damit vorab bekannt ist, welcher Eckpunkt des Rechtecks gemeint ist.

Topologie und Routing Contextcast baut, wie auch schon das Publish/Subscribe-System, auf einem Overlay auf. Der Vorteil liegt darin, dass die Protokolle auf Vermittlungs- und Transportschicht nicht geändert werden müssen. Solche Änderungen sind in WANs wie dem Internet unvorstellbar. Es müssten alle verschiedenen Protokolle abgeändert und ein Update auf jede Komponente des Netzes eingespielt werden, was unter anderem aus wirtschaftlichen Gesichtspunkten nicht zu vertreten wäre. Welchen Einfluss dies auf bestehende Applikationen hat, ist nicht abzuschätzen.

Eine Implementierung auf einer Overlay-Schicht ermöglicht die nahtlose Integration mit bereits bestehenden Applikationen, ohne die darunter liegenden Schichten ändern zu müssen. Auf welchem Protokoll das Overlay aufsetzt, spielt ebenfalls keine Rolle. Wichtig ist nur, dass ein Verbindungspfad zwischen allen Knoten des Systems besteht. Contextcast geht im Allgemeinen von einer Topologie aus, die einem ungerichteten, möglicherweise zyklischen Graphen entspricht. Knoten des Systems, die im Overlay nahe beieinander liegen, liegen physisch ebenfalls eng beieinander. Diese Lokalitätseigenschaft ist die Annahme und Grundlage vieler Konzepte des Contextcast und begründen deren Effizienz.

Publish/Subscribe-Systeme wie zum Beispiel Siena geben nicht vor, wie ein möglicher Spannbaum auf dem Netz aufgebaut wird, um mögliche Zyklen bei der Nachrichtenverteilung zu vermeiden. In Contextcast hingegen ist die Berechnung von Spannbäumen integraler Bestandteil des Systems. Dazu wird auf ein leicht abgeändertes Link-State-Protokoll [Tan03] zurückgegriffen. Die beteiligten Komponenten sind die routingfähigen Komponenten des Overlays. Es handelt sich also um ContextRouter und ContextNodes. Die Informationen über Nachbarn und den Übertragungszeiten zu diesen bilden ein *Link-State-Paket*. Die durchschnittlichen Übertragungszeiten können durch Messungen der Round-Trip-Time[20] (RTT) zu den Nachbarn ermittelt werden. Dies geschieht beispielsweise durch den Versand von ECHO-REPLY-Nachrichten. Andere Methoden und Metriken für die Abschätzung der Entfernung zweier verbundener Knoten sind ebenfalls möglich.
Die Link-State-Pakete werden nun mit einem Zeitstempel und einem Sender-ID versehen und per Flooding im Netz an alle ContextRouter und ContextNodes verteilt. Nachdem alle Komponenten Link-State-Pakete aller anderen Komponenten erhalten haben, kann unter Einsatz des Dijkstra-Algorithmus [Tan03] für jede Komponente ein Spannbaum für den Graphen des Netzes berechnet werden. Die Durchführung des Link-State-Protokolls kann periodisch oder bei Topologieänderungen, z.B. durch den Abbruch von Verbindungen oder durch dem Overlay beitretende Komponenten, durchgeführt werden. Eine periodische Durchführung kann auch dann sinnvoll sein, wenn sich die Topologie des Netzes nicht geändert hat. Durch den ungleichmäßigen Nachrichtenverkehr im Netz ändern sich die Leitungsverzögerungen der Verbindungen, die beim Link-State-Protokoll für die Gewichtung der Graphenkanten verwendet werden. Eine erneute Durchführung des Link-State-Protokolls passt die Spannbäume dem Nachrichtenverkehr an und erzielt damit eine bessere Verteilung des Nachrichtenaufwands. Der individuelle Spannbaum einer Komponente vereinfacht das Flooding von Nachrichten und kann für das Senden von Nachrichten entlang des kürzesten Pfads zu den Empfängern verwendet werden.

Ein primitiver Lösungsansatz zur Verteilung von Nachrichten in einem Contextcast[21] stellt das Flooding der Nachrichten entlang des Spannbaums des Senders dar. Dadurch könnte der ContextNode als Empfänger überprüfen, welche Kontexte der an ihm angemeldeten Benutzer zu der Nachricht passen und die Nachricht an die entsprechenden ContextHosts zustellen. Dieser naive Ansatz erzeugt jedoch ein zu hohes Verkehrsaufkommen an Nachrichten. In einem Contextcast wird angenommen, dass das Flooding der Kontexte im Gegensatz zum Flooding der Nachrichten einen weitaus geringeren Nachrichtenaufwand darstellt. Dies ist aus der Annahme begründet, dass sich die Kontexte im Regelfall seltener ändern und damit neu propagiert werden müssen, als Nachrichten am System angemeldet werden. Aus diesem Grund werden die Kontexte durch einen *Broadcast*-Algorithmus an alle ContextNodes verteilt. Wie dadurch die Verteilbäume der Nachrichten eingeschränkt werden können, wird im weiteren Verlauf dieses Abschnittes noch näher erläutert.

Der für das Broadcasting der Kontexte gewählte Algorithmus ist das *Reverse Path Forwarding* [Tan03]. Es stellt eine einfache Art des Broadcastings dar, in der die Router die

[20] Die Zeit, die ein Datenpaket vom Sender zum Empfänger und wieder zurück benötigt
[21] Das Routing von Nachrichten in einem Contextcast-System wird selbst auch als Contextcast bezeichnet

Spannbäume der anderen Knoten nicht kennen müssen und auch kein Mechanismus zur Zyklenerkennung oder zum Abbruch des Verfahrens notwendig ist. Ein Kontext wird dabei beim Empfang durch einen Knoten für ein späteres Matching gespeichert und an all seine Nachbarn weitergeleitet. Dies geschieht aber nur, falls der Kontext auf der Verbindung eingegangen ist, welche vom Empfänger des Kontextes aus auch für das Senden von Nachrichten an den Absender des Kontextes verwendet wird. Hierbei ist zu beachten, dass die ContextNode-IDs in den Kontexten mit den IDs der ContextNodes in den Link-State-Paketen übereinstimmen müssen. Durch diesen Ablauf wird garantiert, dass die Kontexte auf den kürzesten Pfaden zu den Empfängern ausgeliefert werden und dass keine Zyklen entstehen. Die Propagation der Kontexte macht diese nicht nur bei allen ContextNodes bekannt, sondern auch bei allen ContextRoutern. Alle Kontexte, die ein ContextRouter auf dem kürzesten Pfad zum Sender empfängt werden gespeichert und den Leitungen zugeordnet, über die sie am ContextRouter eingegangen sind.

Der Verteilbaum einer Nachricht ergibt sich durch die kürzesten Pfade des Senders zu den ContextNodes. Diese Pfade werden durch die Propagation der zu der Nachricht passenden Kontexte etabliert. Eine Nachricht wird auf den Verbindungsleitungen eines Knotens weitergeleitet, denen ein zu der Nachricht passender Kontext zugeordnet ist. Da die Kontexte in jedem Router in Bezug auf die Verbindung, aus der sie eingegangen sind, gespeichert werden, werden auch die passenden Nachrichten auf dieser Leitung übermittelt. Somit wandern die Nachricht auf dem gleichen Pfad zum Sender eines Kontextes, auf welcher der Kontext propagiert wurde. Alle Ausgangsverbindungen eines ContextRouters, an welchen kein passender Kontext angemeldet wurde, kürzen den Verteilbaum der Nachricht, da sie auf diesen Leitungen nicht übermittelt werden muss. Mit „erfolgreichem Matching“ ist hier auch gemeint, dass der ContextNode-ID in einem zur Nachricht, auf Basis der Attributfilter, passenden Kontext auch in der Empfängerliste der Nachricht enthalten ist (vgl. Definition 2.4.2). Ob sich ein ContextNode in Richtung einer bestimmten Leitung auf dem kürzesten Pfad zu den Empfängern befindet, könnte theoretisch auch durch den Spannbaum überprüft werden, der durch das Link-State-Protokoll aufgebaut wurde. In diesem Fall wird eine Trennung der symbolischen Lokationsinformation und des Kontextes vollzogen. In [Gei07] wurde diese Art der separaten Informationsverbreitung von Lokation und Kontext preferiert. Die Evaluation ergab jedoch, dass dadurch ein erheblicher Mehraufwand an Nachricht durch *false-positive*-Zustellung von Nachrichten entsteht. Vorteile dieser Informationsverbreitung waren z.B. das bessere Aggregationspotential der Kontexte. Man kann zusammenfassend sagen, dass durch das Entfernen von Attributen das Potential steigt. Gleichzeitig verringert sich das Potential erheblich, falls relativ individuelle Attribute, wie z.B. IDs, in den Kontext aufgenommen werden.

Aggregationsverfahren wie das Covering und Merging, sind nicht Fokus dieser Arbeit, weshalb wir von einer Propagation der Kontexte inklusive ContextNode-IDs als symbolische Lokationsangaben ausgehen. Die geographischen Lokationsangaben der Nachrichten wird nur für die Ermittlung der Empfängerliste und der eventuellen Lokationsprüfung beim ContextHost benötigt. Sie ist also nicht Teil der Attributliste des Kontextes und wird daher von den ContextRoutern nicht beachtet.

Die Nachrichten werden über ein Xcast, inklusive Matching auf Kontexte bei den Knoten des Contextcast-Systems, an die Empfänger übermittelt. Die Verwendung von Xcast+ bietet keine nennenswerte Verbesserung, da die ContextNodes als Empfänger schon so etwas wie Designated Router darstellen. Die Nachrichten werden somit über eine Art Multicast an die Empfänger weitergeleitet, weshalb Contextcast wie auch Publish/Subscribe einem Multicast-System entspricht. Eine Kopie der Nachricht wird erzeugt, wenn ein ContextRouter bei der Weiterleitung mehrere Ausgangsleitungen für die Nachricht ermittelt. Jeweils eine Kopie der Nachricht wird dann an jede dieser Verbindungen ausgestellt. Die Empfängerlisten der

einzelnen Kopien müssen wie beim regulären Xcast angepasst werden. Sie enthalten nur noch diejenigen Empfänger, die über die jeweilige Verbindungsleitung auf dem kürzesten Pfad erreichbar sind.

Der Algorithmus für die Verteilung von Kontexten und Zustellung von Nachrichten verfolgt eine *best effort* Strategie. Eine korrekte Zustellung von Nachrichten wird nur durch die Abwesenheit von Verbindungsproblemen und ausfallenden Routern garantiert. Die Robustheit des Systems ergibt sich durch eine erneute Berechnung der Spannbäume bei einer erneuten Durchführung des Link-State-Protokolls (im Falle eines Verbindungsabbruchs oder Routerausfalls). Solange das Netz zusammenhängend bleibt, also ein logischer Pfad zwischen allen Knoten des Systems existiert, kann ein korrektes Routing wiederhergestellt werden und die Nachrichten werden wieder vollständig zugestellt. Sollte es in diesem Zusammenhang zu einer Topologieänderung kommen, müssen die Kontexte nach Durchführung des Link-State-Protokolls erneut propagiert werden, da sich durch die Änderungen der Spannbäume neue kürzeste Pfade ergeben. Damit alle Knoten unabhängig neue Link-State-Pakete verschicken, muss der Knoten, der die Topologieänderung entdeckt hat, alle anderen Knoten über diese Änderung informieren. Diese Änderung kann auch durch mehrere Knoten entdeckt werden. In [Gei07] wird vorgeschlagen, dass nach einer erneuten Durchführung des Link-State-Protokolls alle Kontexte von den ContextNodes erneut propagiert werden. Die ContextRouter müssen dann bei Erhalt eines Kontextes alte Kontexte mit gleichem ID löschen, da diese eventuell falschen Verbindungsleitungen zugeordnet sind. Dadurch wird ein neuer Verteilbaum für die Nachrichten etabliert, welcher die kürzesten Routen entsprechend der neuen Topologiestruktur enthält. Dieses Verfahren birgt zwei Probleme in sich: Erstens wird das Netz bei einer Topologieänderung auf einen Schlag von einer Menge von Nachrichten überhäuft, die die neuen Kontexte per Broadcast im Netz propagieren. Zweitens werden durch das Reverse-Path-Forwarding Kontexte auch an ContextRouter verteilt, die sich nicht auf einem kürzesten Pfad zu einem ContextNode befinden. Bei einer Verteilung der Kontexte sollten nur die ContextNodes als Empfänger betrachtet werden. Ein möglicher alternativer Ansatz verwendet dazu ein Xcast oder IP-Multicast an alle ContextNodes, wobei ein ContextRouter, ähnlich dem IP-Multicasting, bei Erhalt eines Kontextes den Empfang der Nachricht auf dem kürzesten Pfad wie beim Reverse-Path-Forwarding prüfen muss. Für die Unterscheidung von reinen ContextRoutern und ContextNodes wird ein zusätzliches Flag in den Link-State-Paketen erforderlich, welches eine Identifikation eines ContextNodes ermöglicht. Allerdings wird der Nachrichtenheader einer Xcast-Nachricht sehr groß, wenn alle ContextNodes als Empfänger in der Nachricht beinhaltet sind. Weiterhin müssen durch diese Anpassung bei einer Topologieänderung alle ContextRouter auf Basis des alten und neuen Spannbaums bestimmt werden, die nach der Topologieänderung nicht mehr Teil kürzester Pfade sind. Zu diesen Routern müssen gezielt Löschnachrichten für die Kontexte des ContextNode geschickt werden. Zusätzlich müssen die Kontexte nur zu den ContextNodes neu versendet werden, zu denen durch die Topologieänderung eine neue kürzeste Route entstanden ist. Dieser Versand kann wieder durch Xcast der Kontexte durchgeführt werden. Werden diese Operationen durch zufällige Wahl eines Timers pro ContextNode verzögert, kann eine plötzliche Überflutung des Netzes verhindert werden. Weitere Themen wie eine optimale Roamingunterstützung für mobile Clients, Covering, Merging, Advertisements für Nachrichten und eine optimierte Datenstruktur für Kontexte in den ContextRoutern zur Reduzierung der Matching-Komplexität sind aktuelle Forschungstätigkeiten im Bereich Contextcast. Einige dieser Themen wurden jedoch bereits bei der Entwicklung des grundlegenden Systems beachtet, was die Eingliederung der Verfahren in das System erleichtern sollte.

Unterstützung mobiler Clients Für die Unterstützung mobiler Clients im Contextcast wurden spezielle Algorithmen entwickelt. In diesem Zusammenhang werden andere Anforderun-

gen an das System gestellt, als zum Beispiel bei der primären Unterstützung stationären Terminals. Bei stationären Terminals wird die Lokationsverwaltung dadurch vereinfacht, dass sich die Terminals nicht bewegen und daher von einer eher dauerhaften Anbindung an einen ContextNode ausgegangen werden kann. Gleichzeitig ist davon auszugehen, dass stationäre Clients in der Regel ihre Kontexte vom System abmelden, falls diese nicht mehr gültig sind. Bei mobilen Clients hingegen stellen Verbindungsabbrüche beispielsweise durch schlechten Empfang oder Übergänge in einen Stand-by Modus und fluchtartige Ortswechsel keine Seltenheit dar.

Die Kontexte der Benutzer werden aus den genannten Gründen in einem *soft-state* Ansatz bei den ContextNodes gespeichert. Ein Benutzer muss in regelmäßigen Abständen durch die Aufforderung des ContextNodes die Aktualität seiner Kontexte bestätigen. Dazu sendet der ContextNode zum Beispiel in regelmäßigen Abständen ein *Beacon*-Paket[22] aus. Erfolgt die Bestätigung nach Ablauf einer gewissen Zeit nicht, so kann der ContextNode davon ausgehen, dass der Benutzer nicht mehr präsent ist. Damit nun keine ungültigen Kontexte als Altlasten im System zurückbleiben und einen unnötigen Nachrichtenaufwand erzeugen, werden die entsprechenden Kontexte durch eine Löschnachricht der ContextNodes aus dem System entfernt. Die Beacon-Pakete können gleichzeitig von den Benutzern für das Auffinden eines ContextNodes beim *Bootstrapping*-Prozess[23] verwendet werden. Bei einem Absturz eines ContextNodes sollten die Kontexte aus einem persistenten Speichermedium ausgelesen werden können. Andernfalls müssten alle Kontexte mit dem entsprechenden ID des ContextNodes aus dem System durch eine Löschnachrichten entfernt werden. Danach werden die Benutzer vom ContextNode aufgefordert, ihre Kontexte neu zu übertragen, die dann neu im Netz propagiert werden müssen.

Ein weiteres Problem ergibt sich, wenn ein mobiler Client für eine gewisse Zeit keine Verbindung zum System hat und in dieser Zeit Nachrichten am entsprechenden ContextNode eingehen, die zu einem Kontext des Benutzers passen. Kann der Benutzer anschließend die Verbindung wieder herstellen, müssen ihm die in seiner Abwesenheit eingetroffenen Nachrichten zugestellt werden. Aus diesem Grund werden Nachrichten an Benutzer beim ContextNode gepuffert, falls jene Benutzer zum Zeitpunkt des Empfangs der Nachricht „offline“ sind. Die Nachrichten im Puffer können mit einer *Time-to-live*[24] (TTL) versehen werden, oder sie werden so lange gepuffert, bis der Kontext des Benutzers durch den soft-state Ansatz aus dem System entfernt wird. Diese Entscheidung und die Wahl des Knotens, der die TTL festlegt, kann bei der Implementierung des Systems getroffen werden. Falls dazu synchrone Uhren benötigt werden, kann auf das *Network Time Protocol* (NTP) [Mil92] oder das davon abgeleitete, einfachere *Simple Network Time Protocol* (SNTP) [Mil96] zurückgegriffen werden.

Da sich mobile Clients häufig durch eine geringe Kapazität der Energieversorgung auszeichnen, dürfen nur so wenig Nachrichten wie möglich an die Clients ausgeliefert werden. Methoden zur Duplikatvermeidung von Nachrichten spielen daher eine große Rolle. In der bisher vorgestellten Version des Contextcast werden keine Duplikate erzeugt und an die Benutzer ausgeliefert, weshalb Mechanismen zur Duplikatvermeidung nicht betrachtet wurden. Mit der Einführung der Möglichkeit eines Mergings von Kontexten als Beispiel, spielen Mechanismen zur Duplikatvermeidung jedoch sehr wohl eine Rolle.

Im Contextcast existieren noch eine Menge verschiedener Systemnachrichten, mit denen zum Beispiel Kontexte aus dem System entfernt werden können. Die Vorgehensweise dieser Algorithmen ist mehr oder weniger geradlinig, weshalb sie hier nicht weiter erläutert werden.

[22] Signalpaket
[23] Bezeichnet den gesamten Prozess der Anbindung eines neuen Benutzers an das System
[24] Zeitpunkt, nach dessen Ablauf die Nachricht verworfen wird

Interessierte Leser werden an die Literatur [Gei07] verwiesen.
Zusammenfassend bietet das Contextcast Anwendungen die Möglichkeit, Nachrichten per Multicast an eine Menge von Empfängern über ein Overlay zu übermitteln. Der Nachrichtensender muss die Empfänger der Nachricht vorab nicht kennen. Durch die Verwendung von Filtern in den Nachrichten wird durch das Inhalt-basierte Routing eine Menge von Empfänger ermittelt, deren Eigenschaften in Form von Kontexten dem Filter der Nachricht entsprechen. An dieser Stelle soll nochmals darauf hingewiesen werden, dass das Contextcast zwar viele Ähnlichkeiten zum Publish/Subscribe-Kommunikationsparadigma aufweist, diese sich aber bereits in den Anforderungen stark unterscheiden. Im Contextcast werden die Empfänger einer Nachricht vom Sender adressiert, was eine klare Abgrenzung zu Publish/Subscribe darstellt. Benutzer des Systems sind potentielle Empfänger von Nachrichten. Sie stellen dem System lediglich eine Menge nicht verknüpfter Eigenschaftswerte über sich selbst zur Verfügung, um ein gezieltes Routing der Nachrichten zu ermöglichen. In den folgenden Kapiteln dieser Arbeit wird sich noch herausstellen, dass diese Unterschiede zu Publish/Subscribe-Systemen erhebliche Auswirkungen auf die Anwendbarkeit von Publish/Subscribe-Konzepten auf das Contextcast hat.

3 Problemstellung

Nachdem im vorherigen Kapitel die Grundlagen abgehandelt wurden, ist in diesem Kapitel die Problemstellung und Ausgangsbasis für das zu entwickelnde System festgehalten. Die Beschreibung der Möglichkeiten bei der Nachrichtenadressierung in einem Contextcast weist auf Einschränkungen hin, die durch die Erweiterung zu einem zeitbezogenen Contextcast aufgehoben werden sollen. Nach einer Beschreibung des primären Ziels und einer Abgrenzung zu dem regulären Contextcast werden zwei Beispielszenarien angeführt, die den Funktionsumfang des Systems anhand von praxisnahen Beispielen umreißen sollen. Anschließend werden aus diesen Beispielszenarien heraus die Anforderungen an das System definiert, welche gleichzeitig den Abschluss dieses Kapitels bilden.

Durch das Contextcast ergibt sich ein neues Kommunikationsparadigma. Empfänger von Nachrichten werden durch selektive Filter adressiert, die auf Situationseigenschaften der Netzteilnehmer ausgewertet werden. Die gebündelten Eigenschaftswerte einer Entität wird in diesem Zusammenhang Kontext genannt. Nach [DA99] ist ein Kontext wie folgt definiert:

Definition 3.0.3 *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*

Diese Definition wurde im Zusammenhang der kontextbezogenen Applikationen erarbeitet. Diese Arbeit beschäftigt sich jedoch nicht mit kontextbezogenen Anwendungen, sondern mit kontextbezogener Adressierung und Kommunikation. Eine Entität ist demnach alles, was von einer anderen Entität adressiert werden möchte und damit einen Kommunikationspartner darstellt. Das Contextcast liegt als Middleware im Schichtenmodell noch unter der des Benutzer und der Anwendung. In [DA99] wurden die Faktoren Lokation und *Zeit* als primäre Teile eines Kontextes herausgearbeitet. Während sich das ursprüngliche Design des Contextcast intensiv mit der spezielle Behandlung der Lokation als Kontextattribut beschäftigte, soll in dieser Arbeit dem Faktor „Zeit“ mehr Aufmerksamkeit gewidmet werden.

Im Gegensatz zu anderen Kontextattributen unterliegt die Zeit ständig fortlaufenden Werteänderungen. Es macht also keinen Sinn die aktuelle Uhrzeit der Umgebung einer Entität in den Kontext zu integrieren. Interessanter erscheint vielmehr der Wechsel der Situationseigenschaften einer Entität und damit der Kontexte im zeitlichen Verlauf. Die Kontexte der Entitäten können im Gegensatz zu kontextbezogenen Applikationen nicht abgefragt oder für Berechnungen verwendet werden. Sie dienen einzig dem Routing von Nachrichten. In dem ursprünglichen Contextcast gemäß [Gei07] konnte eine fehlerfreie Übermittlung der Nachrichten bei fehlerfreier Funktionalität der Knoten und Verbindungen unter folgenden Bedingungen garantiert werden:

1. Der Empfänger befindet sich zum Zeitpunkt t_{s1} in einem zur Nachricht passenden Kontext K.

2. Zum Zeitpunkt $t_{s2} > t_{s1}$ wurde der Kontext K zumindest bis zum Senderknoten der Nachricht propagiert. Werden Konzepte wie das Covering und Matching angewandt, kann der Kontext auch durch einen ihn überdeckenden Kontext oder einen zusammengefassten Kontext repräsentiert werden.

3. Zum Zeitpunkt $t_{s2} + d_{max} = t_e$ (Empfangszeitpunkt der Nachricht) befindet sich der Empfänger nach wie vor im Kontext K und es wurde kein Wechsel oder eine Abmeldung

des Kontext im Zeitraum $[t_{s2}, t_e]$ vollzogen. d_{max} steht für die maximale Nachrichtenverzögerung des Netzes, die statistisch ermittelt wird und mit der die Übertragungszeit der Nachricht an den Empfänger nach oben hin abgeschätzt wird.

Nachrichten können nur an Benutzer zugestellt werden, die zum Zeitpunkt des Nachrichtenversands einen gültigen und zur Nachricht passenden Kontext angemeldet haben. Diese Einschränkung soll durch die in dieser Arbeit vorgestellten Verfahren aufgehoben werden. Die einzelnen Kontexte lassen sich entlang eines Zeitstrahls anordnen. Dieser Zeitstrahl gibt den Zeitraum der Gültigkeit der Kontexte an, also wann die Kontexte am System angemeldet waren und die Situationseigenschaften der Entität in diesem Zeitraum repräsentierten. In einem *zeitbezogenen Contextcast*, das die in dieser Arbeit beschriebenen Konzepte vereint, lassen sich die Nachrichten an alle Kontexte des Zeitstrahls adressieren. Gleichzeitig können Filter, wie auch beim ursprünglichen Contextcast, auf weitere Attribute eines Kontextes definiert werden. Die Ausdrucksfähigkeit der Filter für eine Nachricht wird durch die Einführung eines zeitlichen Attributfilters erweitert. Dieser schränkt den Zeitraum für die zur Nachricht passenden Kontexte durch die Angabe von Zeiten ein. Eine Nachricht mit solch einem Attributfilter wird im Folgenden *zeitbezogene Nachricht* genannt. Aus Sicht des Nachrichtensenders verliert die Nachricht durch den zusätzlichen zeitlichen Attributfilter ihren Bezug auf die Gegenwart völlig. Vergangene, gegenwärtig und zukünftig angemeldete Kontexte werden auf die gleiche Art und Weise adressiert. Alle Verfahren, Methoden und Algorithmen in dieser Arbeit bauen auf dem ursprünglichen Contextcast auf und erweitern dieses zu einem zeitbezogenen Contextcast mit den eben genannten Adressierungsmöglichkeiten.

3.1 Beispielszenarien

Im Folgenden werden zwei Beispielszenarien geschildert, für die sich eine Anwendung des zeitbezogenen Contextcast anbietet. Sie zeigen gleichzeitig die Schwächen des ursprünglichen Entwurfs des Contextcast auf, welches den Anforderungen der folgenden Szenarien nicht gerecht werden kann. Von den Anwendungen, die das zeitbezogene Contextcast-System benutzen, wird abstrahiert. In den beschriebenen Fällen geht es hauptsächlich um Mitteilungen an die Benutzer des Systems. Es wird also davon ausgegangen, dass eine Anwendung mit einer wohl bekannten Applikations-ID auf den Geräten der Benutzer läuft, die Nachrichten entgegen nimmt und diese dem Benutzer über eine Ausgabeschnittstelle mitteilt.

3.1.1 Zeugenfahndung

Am Donnerstag, den 3.4.2008 ereignete sich um exakt 8.30 Uhr ein Verkehrsunfall am Leonberger Autobahnkreuz in der Region Stuttgart. Nachdem eine Kraftfahrzeugführerin zeitgleich mit einem LKW-Fahrer die Leitspur wechseln wollte, kollidierten die zwei Fahrzeuge. Zwanzig Minuten später trifft die Polizei mit Verspätung durch den entstandenen Stau am Unfallort ein, sichert den Unfallort und nimmt die Aussagen der Betroffenen vorschriftsmäßig zu Protokoll. Zwei Wochen später kommt die Verkehrspolizei zu dem Schluss, dass der Tathergang aufgrund sich widersprechender Aussagen der Unfallbeteiligten und mangelnden Beweisen nicht exakt rekonstruierbar ist. Daher entscheidet sich die Polizei dazu, eine Nachricht über ein zeitbezogenes Contextcast-System an alle Fahrzeughalter zu senden, die sich am 3.4.2008 zwischen 8.25 Uhr und 8.32 Uhr im Umkreis von einem Kilometer um den Unfallort befanden. Die Nachricht beinhaltet die Aufforderung sich bei der nächsten Polizeistelle zu melden, falls der Empfänger der Nachricht Zeuge des Vehrkehrsunfalls gewesen ist. Mit dieser Maßnahme erhofft sich die Polizei durch weitere Zeugenvernehmungen möglichst schnell den Tathergang zu klären.

Da die Kontexte eines Contextcast grundsätzlich nur die aktuelle Situation eines Benutzers beschreiben, können Nachrichten nicht an vergangene Situationen bzw. Kontexte eines Benutzers adressiert werden. Das zeitbezogene Contextcast hingegen ermöglicht die Adressierung auf Basis von Informationen über vergangene Situationseigenschaften der Empfänger. In dem hier gewählten Szenario lässt sich die Nachricht der Polizei für ein zeitbezogenes Contextcast abstrakt wie folgt darstellen:

lokation	$\in$	[Bereich ein Kilometer um Unfallstelle]
typ	$=$	person
transportmittel	$=$	automobil
uhrzeit	$\in$	[08:25:00/03.04.2008 , 08:32:00/03.04.2008]
payload	$=$	Bitte bei nächster Polizeistelle melden

Die Nachricht bezieht sich auf Personen, die sich innerhalb des gegebenen Zeitintervalls in der Nähe des Autobahnkreuzes befanden. Zu dem Zeitpunkt des Absendens der Nachricht können sich die Empfänger an einem ganz anderen Ort aufhalten oder sind eventuell nicht einmal mit dem System verbunden. Unter all diesen Umständen sollte eine erfolgreiche Nachrichtenübermittlung möglich sein.

Solche Nachrichten sollten natürlich signiert werden, damit eine eindeutige Zuordnung der Nachricht zu der Polizei als Absender durch den Empfänger realisiert werden kann. Derartige Maßnahmen werden auf Anwendungsebene zwischen den Applikationen getroffen und beeinflussen das zugrunde liegende System und die Nachrichtenzulieferung daher nicht.
Die Verwendung eines Publish/Subscribe-Systems für das geschilderte Szenario liefert ebenfalls nicht das gewünschte Ergebnis. In der Publish-Subscribe Terminologie stellt die Nachricht der Polizei eine Notification dar. Notifications werden nur zu den Empfängern übermittelt, die vorab ein Interesse in Form einer Subscription angemeldet haben, die zur betreffenden Notification passt. In diesem Zusammenhang erscheint es recht unwahrscheinlich, dass sich Personen nach zwei Wochen noch dafür interessieren, welche Meldungen der Polizei bezüglich des Unfalls am Leonberger Autobahnkreuz verschickt werden. Sollte sich eine Person, die sich zum Zeitpunkt des Unfalls und zu dem Zeitpunkt, an dem die Meldung der Polizei verschickt wurde, am Leonberger Kreuz befinden, ist eine erfolgreiche Übermittlung der Notification an diese Person wahrscheinlicher. Dieser Fall erweist sich jedoch als recht unwahrscheinlich.

In einem zeitbezogenen Contextcast wird die Nachricht der Polizei auch bei einer vorübergehenden Abwesenheit eines Empfängers unabhängig vom aktuellen Aufenthaltsort mit einer recht hohen Wahrscheinlichkeit zugestellt. Wie auch bei dem ursprünglichen Design eines Contextcast verfolgt das zeitbezogene Contextcast den *best effort* Ansatz. Bei Verbindungsabbrüchen, Hardwarefehlern oder dem Absturz eines Knotens kann und soll auch keine Zulieferung der Nachricht garantiert werden. Es werden jedoch Maßnahmen getroffen, um das System schnellstmöglich wieder in einen konsistenten Zustand zu versetzen.

3.1.2 Messehinweis

In der neuen Stuttgarter Messe soll in wenigen Tagen eine Firmenkontaktveranstaltung für Absolventen stattfinden. Die Future Systems Entwicklung GmbH wird in Austellungshalle 7 vertreten sein. Die Zielgruppe der Firmenvertreter sind verstärkt Hochschulabsolventen des Studiengangs Informatik. Da gerade diese Zielgruppe meist mit mobilen Geräten und Netzwerkzugang ausgestattet sind, möchten die Vertreter durch Hinweise auf den Stand auf sich aufmerksam machen. Dazu setzt das Event-Management der Firma eine Nachricht an einem zeitbezogenen Contextcast-System ab, die einen Hinweis auf den Stand in Austellungshalle 7 beinhaltet. Diese Nachricht soll an den Veranstaltungstagen der Messe an die Zielgruppe

verschickt werden, die sich innerhalb des Messegeländes befindet. Die Nachricht sieht in etwa wie folgt aus:

lokation	$\in$	[Bereich des Messegeländes]
typ	$=$	student
studiengang	$=$	informatik
fachsemester	$\in$	[9,15]
uhrzeit	$\in$	[Öffnungszeiten der Veranstaltung]
payload	$=$	[Informationen über den Messestand]

Das zeitbezogene Contextcast nimmt die Nachricht schon vor der dem Veranstaltungstermin an. Das Matching auf Kontexte wird bis zum Eröffnungstermin verzögert. Dann erfolgt ein Matching auf die Kontexte aller Messebesucher, die sich innerhalb des Messegeländes aufhalten.

Unter Einsatz des ursprünglichen Contextcast ließe sich ein solcher Nachrichtenversand nur durch das periodische Versenden der Nachricht ohne Uhrzeit im Veranstaltungszeitraum realisieren. Eine verzögerte Auswertung von Nachrichten war, wie auch das Matching über einen längeren Zeitraum hinweg, noch nicht vorgesehen. Ein schwerwiegender Nachteil, der durch diese Vorgehensweise entsteht, ist der erzeugte Nachrichtenaufwand und damit die Belastung des Netzes. Bei jedem Versand der Nachricht wird diese auf alle angemeldeten Kontexte im Messebereich ausgewertet und den Benutzern mit passendem Kontext zugestellt. Benutzern mit passendem Profil, die sich längere Zeit auf dem Messegelände aufhalten, wird die Nachricht in jeder neuen Periode des Nachrichtenversand erneut zugestellt, ungeachtet der bereits erfolgreich durchgeführten Nachrichtenübermittlungen.

3.2 Anforderungen

Die zentrale Erwartung, die an ein zeitbezogenes Contextcast gestellt wird, ist die Verfügbarkeit von historischen Kontexten, an die Nachrichten adressiert werden können. In diesem Zusammenhang müssen Methoden zur Verfügung stehen, die das Speichern und Abfragen dieser Kontexte ermöglichen, damit ein Matching und Routing der zeitbezogenen Nachrichten möglich ist. Für ein Matching von zeitbezogenen Nachrichten ist zu allererst ein Datentyp zu definieren, mit dem ein Filter für die Zeiträume der Kontexte konstruiert werden kann. In Verbindung zu diesem Datentyp stehen die Operatoren auf Zeiträume, die ebenfalls Teil des zeitlichen Attributfilters sind. Um die Akzeptanz des Systems aus Benutzersicht zu steigern, müssen Mechanismen zum Schutz der Privatsphäre von Benutzern vorhanden sein. Lassen sich die Kontexte eindeutig auf bestimmte Personen im realen Leben abbilden, so können leicht detaillierte Profile von Benutzern angelegt werden. Darüber hinaus ist eine Verfolgung von sich bewegenden Personen möglich, in der die Position eines Benutzers über den zeitlichen Verlauf hinweg exakt dargestellt werden kann. Zu den Mechanismen zum Schutz der Privatsphäre stehen die effizienten Routingverfahren der zeitbezogenen Nachrichten in Konflikt.

Die Hauptfaktoren zur Bewertung der Effizienz sind in diesem Sinne der benötigte Nachrichtenaufwand und die Übertragungszeit von zeitbezogenen Nachrichten. Die Verteilung der Nachrichten durch Unicast-Übertragung an die einzelnen Empfänger wird an dieser Stelle als ineffizient betrachtet. Für die Entwicklung der Routingverfahren stehen gewisse Gestaltungsspielräume offen. Die verschiedenen Verfahren sollen anhand von Simulationen basierend auf unterschiedlichen Szenarien anhand von verschiedenen ausgewählten Metriken analysiert werden. Dabei sind die Vor- und Nachteile der Verfahren zu erarbeiten.

Wie auch schon bei dem ursprünglichen Design des Contextcast liegt hier die Unterstützung von mobilen Clients im Vordergrund. Da diese meist keine durchgängige Verbindung

zum System pflegen und über mehr oder weniger große Zeiträume nicht erreichbar sind, muss ebenfalls ein Verfahren zur Speicherung der Nachrichten zur Verfügung stehen. Sind die Empfänger über einen längeren Zeitraum hinweg nicht erreichbar, bietet sich zur Speicherung der Nachrichten ein Mailboxsystem an. All diese Anforderungen an ein zeitbezogenes Contextcast werden als minimalistisch angesehen.

4 Aspekte eines zeitbezogenen Contextcast

In diesem Kapitel wird ein vollständiger Entwurf eines zeitbezogenen Contextcast beschrieben, der den im vorherigen Kapitel aufgeführten Anforderungen genügt. Als Erstes folgt eine Einleitung in die Problemstellungen des zeitbezogenen Contextcast und eine Abgrenzung zu Publish/Subscribe, sowie ein kurzer Gesamtüberblick über das System. Anschließend werden in Abschnitt 4.1 die Konzepte zur Historisierung von Kontexten beschrieben. Diese beinhalten die Festlegung des zeitlichen Gültigkeitsintervalls eines Kontextes, des Speicherorts mit den daraus resultierenden Vor- und Nachteile und der Zugriff auf Kontexte der Kontextdatenbank. Abschnitt 4.2 behandelt die Archivierung von Nachrichten zusammen mit den Vorkehrungen zum Schutz der Privatsphäre eines Benutzers. Anschließend werden die zeitbezogenen Nachrichten in Abschnitt 4.3 kategorisiert und voneinander abgegrenzt. Der verwendete Datentyp für Zeitangaben, die möglichen Operatoren für zeitbezogene Attributfilter und deren Auswertungsregeln sind Themen von Abschnitt 4.4. Nach einer genaueren Betrachtung der Semantik von Zeitangaben im zeitbezogenen Contextcast folgt eine Erläuterung des Matching und Routing zeitbezogener Nachrichten in Abschnitt 4.5. Diese beinhaltet die Kategorisierung der zeitbezogenen Nachrichten bei den unterschiedlichen Knoten des Systems, das detaillierte Routing der unterschiedlichen zeitbezogenen Nachrichten, sowie die Vermeidung von Nachrichtenduplikaten, die durch Mehrfachauswertungen entstehen. Die Zusammenfassung der konkreten Strukturen einzelner Teile des Systems in Abschnitt 4.6 schließt das Kapitel ab. Die Begriffe Zustellung, Übermittlung und Routing, wie auch Client und Benutzer, werden in den folgenden Kapiteln synonym verwendet.

Aus den in Kapitel 3 genannten Anforderungen wird klar, dass das bisher bekannte Contextcast zu einem System erweitert werden muss, welches bestimmte Informationsobjekte speichert. Dadurch können diese über den zeitlichen Verlauf hinweg für Benutzer zugänglich gemacht werden. Gleichzeitig muss jedoch auch verstärkt ein Augenmerk auf die Einhaltung der Privatsphäre und des Datenschutzes eines Benutzers gelegt werden. Hier sind Sicherheitsvorkehrungen nötig, die das beliebige Ausspähen von Benutzerinformationen sowie ein *Tracking*[1] der Aufenthaltsorte des Benutzers verhindern oder zumindest stark erschweren.

Die Lösungsansätze und Konzepte zur Historisierung[2] von Datenobjekten eines Contextcast können prinzipiell in zwei Kategorien eingeteilt werden. Diese unterscheiden sich im Typ der Informationsobjekte, die gespeichert werden müssen. In der ersten Kategorie werden Nachrichten gespeichert. Wie in Abschnitt 2.4 erläutert wurde, unterstützt das Contextcast gezielt mobile Clients. Diese können mit aktuellen Kontexten am System angemeldet sein, obwohl sie derzeit nicht direkt mit dem System verbunden und damit nicht erreichbar sind. Damit eine Nachricht an solch einen Client weitergeleitet werden kann, müssen die an diesen Client gerichtete Nachrichten für eine bestimmte Zeit zwischengespeichert werden. Diese Zwischenspeicherung erfolgt im extremsten Fall bis zu dem Zeitpunkt, an dem Client wieder erreichbar ist.
Die zweite Kategorie befasst sich mit der Speicherung von Kontexten. Sobald ein Kontext vom System abgemeldet wird, löscht das ursprüngliche Contextcast alle Vorkommen dieses Kontextes im gesamten System. Danach kann auf die Benutzerinformation aus diesem Kontext nicht mehr zugegriffen werden. Das ursprüngliche Contextcast lässt also nur Nachrichten zu, die sich auf die aktuelle Situation eines Benutzers beziehen. Dies vereinfacht einerseits den Entwurf des Systems, andererseits wird die Mächtigkeit des Systems und der Umfang an möglichen Nachrichten reduziert. Oft kennt ein Benutzer nur die Situation der

[1] Rückverfolgung
[2] Speicherung von Daten aus der Vergangenheit

Empfänger einer Nachricht zu einem bestimmten vergangenen Zeitpunkt. Diese Information kann im Contextcast jedoch nicht zur Bestimmung des aktuellen Zustands eines Empfängers verwendet werden, welcher für die Zustellung der Nachricht erforderlich ist. Eine Speicherung von Kontexten ermöglicht die Nutzung vergangenheitsbezogener Informationen über Empfänger einer Nachricht zur Bestimmung des Aufenthaltsortes oder des vom Empfänger aktuell angemeldeten Kontextes, so dass die Nachricht übermittelt werden kann.

Eine derartige Unterscheidung der Historisierungskonzepte existiert in einem Publish/Subscribe-System nicht, da sie aus Benutzersicht keinen Sinn ergibt. Dies ist ein weiteres Beispiel für die grundlegenden Unterschiede zwischen Contextcast und Publish/Subscribe. Der in Abschnitt 2.3.3 erläuterte Ansatz für die Historisierung vergangener Datenobjekte speichert nur die Notifications, also die informationstragenden Elemente eines Publish/Subscribe-Systems. Dies ist im Sinne einer Historisierung in einem Publish/Subscribe-System völlig ausreichend. Eine Speicherung von vergangenen Subscriptions liefert dem Benutzer offensichtlich keinen Mehrwert. Sobald ein Benutzer eine Subscription aus dem System entfernt, bekundet er damit eine Änderung seiner Interessen an bestimmten Notifications über Events. Eine Speicherung von Subscriptions macht weder für den Sender einer Nachricht, noch für die potentiellen Empfänger Sinn, da ein Empfänger seine ehemaligen Interessen selbst kennt. Diese spielen aktuell keine Rolle mehr und der Sender einer Nachricht ist nicht für die Adressierung der Empfänger zuständig. In einem Contextcast ist das informationstragende Element, die Nachricht selbst, gleichzeitig auch für die Adressierung der Empfänger zuständig. Eine konkrete Adressierung der Empfänger erfolgt jedoch nur unter der Präsenz der Kontexte. Die Speicherung der Kontexte ist damit für die Adressierung und das Routing notwendig. Da Nachrichten für die Adressierung der Empfänger zuständig sind kann kein weiterer Empfänger nach dem Versand der Nachricht, wie z.B. bei Publish/Subscribe, ein nachträgliches Interesse an dieser Nachricht anmelden. Aus diesem Grund werden die Nachrichten nur für die Empfänger gespeichert, die in der Nachricht durch die Empfängerliste und Filter angegeben und zum gegebenen Zeitpunkt nicht am System angemeldet sind.

Eine Historisierung der Nachrichten zielt also hauptsächlich auf die Unterstützung von mobilen Clients ab, die sich durch eine hohe Mobilität, geringe Ressourcenverfügbarkeit und fehlerbehafteten Verbindungen ausweisen. Die Nachrichten werden je nach Notwendigkeit so lange zwischengespeichert, bis sie an den Empfänger ausgeliefert werden können. Andere Benutzer, die nicht mit einem aktuell gültigen Kontext am System angemeldet sind, können durch Nachrichten des ursprünglichen Contextcast nicht erreicht werden.

Einige Konzepte des in Abschnitt 2.3.3 beschriebenen Verfahrens für Publish/Subscribe lassen sich prinzipiell in ähnlicher Weise auf ein Contextcast übertragen. Zusammenfassend wird in Padres ein vom Benutzer gewählter Filter an die Datenbank gesendet, die dann anschließend die zu dem Filter passenden Notifications für einen späteren Zugriff durch den Benutzer speichert. In Contextcast kann für die Speicherung der Kontexte ebenfalls ein Filter, wie er von Nachrichten benutzt wird, definiert werden. Die Kontexte stellen, wie die Notifications, eine Menge von Attributen dar. Da die Kontexte im gesamten Netz propagiert werden, kann eine beliebig im Netz positionierte Datenbank ohne weitere Interaktion mit anderen Knoten des Systems Kontexte speichern, die zu dem vom Benutzer gewählten Filter passen. Somit sind diese Kontexte im späteren Verlauf für den Benutzer verfügbar. Dieses Konzept fordert aber, dass der Benutzer im voraus weiß, welche Art von Kontexten er in der Zukunft benötigt. In dieser Arbeit wird die Meinung vertreten, dass dieses Wissen nicht vorausgesetzt werden kann und daher alle Kontexte abgespeichert werden müssen. Die Speicherung von Kontexten hat im Gegensatz zur Speicherung von Notifications bzw. Nachrichten den Vorteil, dass sie weitaus weniger speicheraufwendig ist, da sich Kontexte seltener ändern als Nachrichten im System verschickt werden.

Ein naiver erster Ansatz, um die historischen Kontexte für ein Matching und Routing

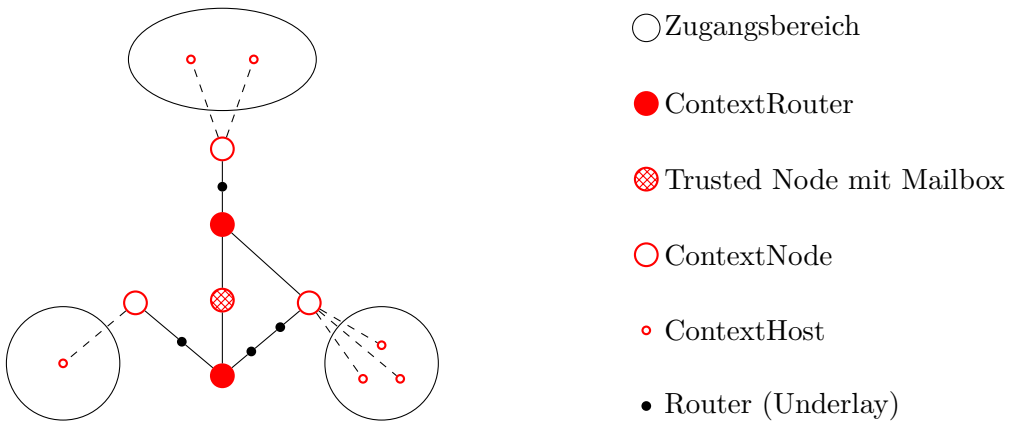


Abbildung 4.1: Komponenten eines zeitbezogenen Contextcast

von Nachrichten verfügbar zu machen, wäre die Propagation dieser historischen Kontexte nebst der regulären Kontexte. Im Gegensatz zu den Kontexten des ursprünglichen Contextcast müssen die historischen Kontexte dauerhaft zur Verfügung gestellt werden. Schnell zeigt sich, dass mit der zeitlichen Anhäufung der Kontexte und der Speicherung dieser entlang der Spannbäume der Knoten eine Speicherbelastung des Systems entsteht, die nicht getragen werden kann. Eine Aggregation der historischen Kontexte bringt keine Besserung. Damit die Speicherbelastung niedrig gehalten werden kann, müssten die Kontexte im Laufe der Zeit so stark aggregiert werden, dass der Nachrichtenaufwand durch überflüssige Nachrichtenübermittlungen enorm steigt. Schon bei der Entwicklung des Internetprotokolls „TCP/IP“ stellte der Transfer der Komplexität weg von den inneren Knoten des Netzes hin zum Benutzer ein primäres Entwicklungsziel dar. Diese Eigenschaft des Protokolls war unter anderem ausschlaggebend für die erfolgreiche Skalierbarkeit. Das gleiche Ziel soll hier im Weiteren auch für die Entwicklung eines zeitbezogenen Contextcast verfolgt werden.

Neben der Fragestellung, wo historische Kontexte abgespeichert werden, beschäftigt sich das „Wie“ mit der Struktur des historischen Kontextes. Das sind die Attribute eines Kontextes und deren Datentypen. Von zentraler Bedeutung sind die zeitbezogenen Angaben in den Kontexten, die eine Adressierung von Nachrichten auf historische Kontexte unter Einsatz von Attributfiltern ermöglichen. Dazu müssen Datentypen zur Speicherung von Zeitpunkten und Operatoren zur Konstruktion zeitbezogener Attributfilter definiert werden. Abhängig von den in einer Nachricht angegebenen Operatoren und Zeitpunkten des zeitbezogenen Attributfilters können diese dann in verschiedene Kategorien, an gegenwärtige, zukünftige oder historische Kontexte gerichtete Nachrichten, eingeteilt werden.
Für Kontexte existieren hingegen nur zwei Zustände: Gegenwärtig am System angemeldete und damit gültige Kontexte und bereits archivierte Kontexte. Für das Matching und Routing der Nachrichten unterschiedlicher Kategorie werden unterschiedliche Verfahren verwendet. In diesem Zusammenhang muss ebenfalls geklärt werden, wo, wann und wie die zeitbezogenen Attributfilter ausgewertet werden. Für die Unterstützung mobiler Clients und die Wahrung der Privatsphäre eines Benutzers müssen zusätzliche Komponenten in das Contextcast-System eingeführt werden.

In Abbildung 4.1 sind die Komponenten eines zeitbezogenen Contextcast in einem beispielhaften Szenario graphisch dargestellt. Im Gegensatz zu den Komponenten des regulären Contextcast sind lediglich die *Trusted Nodes*[3] und die Mailboxknoten hinzugekommen. Trus-

[3] Vertraute Knoten

Komponente	Aufgaben
ContextHost	· Gleiche Aufgaben und Funktionalität wie im Contextcast · Kommunikation mit Trusted Node und Mailbox
ContextRouter	· Gleiche Aufgaben und Funktionalität wie im Contextcast · Xcast zeitbezogener Nachrichten an die ContextNode-Empfänger · Kann gleichzeitig als Trusted Node und/oder Mailboxknoten fungieren · Zwischenspeicherung zeitbezogener Nachrichten · Weiterleitung und Bündelung von Nachrichten, die Listen von Trusted Nodes beinhalten · Weiterleitung von Nachrichten an Trusted Nodes und Mailboxknoten
ContextNode	· Gleiche Aufgaben und Funktionalität wie im Contextcast · Duplikaterkennung für Nachrichten · Speicherung zeitbezogener Kontexte
Trusted Node	· Bietet dem Benutzer die Möglichkeit, seine Identität zu verschleiern · Kann als einzige Komponente die verschiedenen Kontexte den Benutzern zuordnen
Mailbox	· Ermöglicht die Speicherung von Nachrichteninhalten für die anschließende Überlieferung an die Benutzer

Tabelle 4.1: Funktionalität der Komponenten eines zeitbezogenen Contextcast

ted Nodes und Mailboxknoten besitzen unter anderem die gleiche Funktionalität wie ContextRouter oder ContextNodes, da sie auf diesen aufsetzen. Wie in der Abbildung zu sehen ist, können Trusted Nodes und Mailboxen auch auf gleiche Knoten des Overlays platziert werden. Die Aufgaben der einzelnen Komponenten eines zeitbezogenen Contextcast sind in Tabelle 4.1 aufgelistet. Die Fragen, wie genau die Komponenten ihre Aufgaben erfüllen und weshalb gerade diese oder jene Komponenten für diesen Aufgabenbereich ausgewählt wurde, klären die folgenden Abschnitte dieses Kapitels.

Abbildung 4.2 gibt vorab nochmals einen zusammenfassenden Überblick über die zentralen Konzepte eines zeitbezogenen Contextcast. Die Abbildung ist sehr abstrakt gehalten, da sie lediglich einen Gesamtüberblick über das System ermöglichen soll, bevor die einzelnen Verfahren im Folgenden detailliert und intensiv besprochen werden. In der Abbildung sind die Konzepte der unterschiedlichen Hierarchieebenen je nach Ebene mit separaten Farben gekennzeichnet. Die Gruppierung der Konzepte in die Bestandteile, die Funktionsweise und den Hintergrund des Systems auf höchster Ebene sind durch gepunktete Rahmen dargestellt. Die Anmerkungen über den Kurven beziehen sich auf Inhalte und Problembereiche des jeweiligen Konzepts. Zu dem Hintergrund des zeitbezogenen Contextcast gehören die zentralen Anforderungen an das System. Darauf folgt in der Abbildung im Uhrzeigersinn die Funktionsweise des Systems, bestehend aus dem Routing der zeitbezogenen Nachrichten und der Archivierung der Informationsobjekte. Die zentralen Fragestellungen bezüglich der Archivierung der Informationsobjekte sind auf den Kurven im rechten Bereich der Abbildung zu sehen. Der Vermerk „Benutzer nicht erreichbar“ stellt eine Bedingung für die Archivierung von Nachrichten dar.
Als nächstes sind die Bestandteile des Systems zu sehen. Die verschiedenen Informationsobjekte spielen gerade für das Routing eine zentrale Rolle. Zentrale Aufgabe des zeitbezogenen

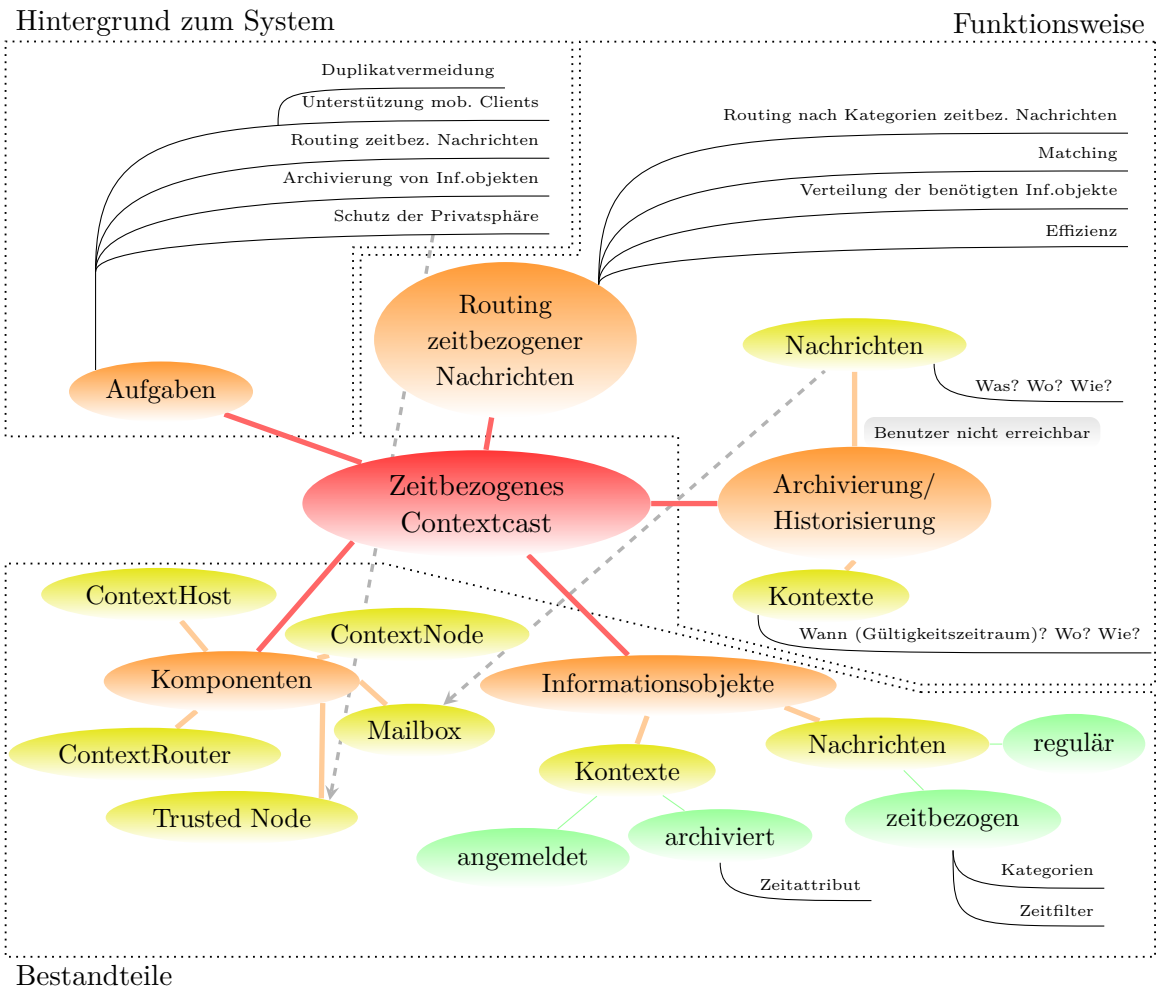


Abbildung 4.2: Konzepte des zeitbezogenes Contextcast

Contextcast bezüglich der Informationsobjekte ist die Kategorisierung dieser und die Definition ihrer Struktur. Die einzelnen Komponenten eines zeitbezogenen Contextcast vervollständigen die Übersicht. Grau gestrichelte Pfeile verdeutlichen die Ursachen für zusätzliche Komponenten im Vergleich zum ursprünglichen Contextcast.

4.1 Historisierung der Kontexte

Bevor die Verfahren zur Historisierung der Kontexte betrachtet werden können, muss zuerst einmal die zeitliche Gültigkeit von Kontexten geklärt werden. Damit ist die Klärung der Frage „In welchem Zeitintervall ist ein bestimmter Kontext gültig, so dass passende Nachrichten direkt zum Benutzer des Kontextes ausgeliefert werden können?“ zu verstehen. Ist diese Frage geklärt, lassen sich die Start- und Endzeitpunkte eines Kontextes genau festlegen. Dies ist aus verschiedenen Gründen notwendig:

- Der Übergang von einem gültigen Kontext zum anderen kann basierend auf den Gültigkeitszeitpunkten genau geklärt werden. Damit lässt sich eine zeitliche Überlappung der Gültigkeit mehrerer Kontexte des gleichen Benutzers verhindern, welche die gleichen Eigenschaften des Benutzers repräsentieren. Im Falle einer zeitlichen Überlappung würde der Benutzer möglicherweise Nachrichten erhalten, die nicht an ihn gerichtet sind

oder Duplikate bereits zugestellter Nachrichten darstellen.

- Der konkrete Endzeitpunkt der Gültigkeit eines Kontextes legt gleichzeitig den Zeitpunkt fest, ab dem ein Kontext archiviert werden kann und als historischer Kontext behandelt wird.

- Nach der Archivierung eines Kontextes wird kein Matching mehr zwischen dem archivierten Kontext und regulären Nachrichten des Systems durchgeführt. Der archivierte Kontext ist damit nicht mehr „gültig“.

Gültigkeit von Kontexten Ein Kontexte wird vom Benutzer an einem ContextNode angemeldet. Es ist daher naheliegend, dass der ContextNode bei der Anmeldung eines Kontextes zum gleichen Zeitpunkt den Startzeitpunkt dieses Kontextes vergibt. Dazu benötigt jeder ContextNode eine Uhr, die beispielsweise über das NTP synchronisiert werden kann. Die Klärung der Frage, wann ein Kontext seine Gültigkeit verliert, bedarf einer etwas umfangreicheren Betrachtung der Vorgänge bei der Abmeldung eines Kontextes. Es existieren insgesamt drei Ursachen für die Abmeldung eines Kontextes:

1. Der Kontext wird explizit durch die Aufforderung des Benutzers in Form einer Systemnachricht vom System abgemeldet.

2. Ein Kontext wird nicht durch den Benutzer aufgefrischt und die für den Kontext vorgesehene TTL läuft ab. Anschließend erfolgt eine Zwangsabmeldung des Kontextes durch den entsprechenden ContextNode.

3. Der Benutzer meldet an einem anderen ContextNode, z.B. im Rahmen eines Roaming, einen Kontext an, der den gleichen ID besitzt wie ein bereits von dem Benutzer angemeldeter Kontext, der nicht abgemeldet wurde. In diesem Fall erfolgt eine Zwangsabmeldung des alten Kontextes durch den ContextNode, an dem der Benutzer vor dem Roaming angemeldet war. Der neue Kontext kann dann wie vorgesehen im System verteilt oder durch eine Update-Nachricht auf Basis des alten Kontextes bekannt gemacht werden. Eine genauere Betrachtung der Vorgänge im Falle eines Roamings des Benutzers ist Teil der aktuellen Forschung.

Im ersten Fall wird der Endzeitpunkt des Kontextes mit dem Eintreffen der Systemnachricht zur Abmeldung des Kontextes beim ContextNode durch diesen vergeben. Der Wert des Endzeitpunkt entspricht dem Zeitpunkt des Eintreffens der Systemnachricht. Im zweiten Fall entspricht der Endzeitpunkt des Kontextes dem Zeitpunkt, an welchem die TTL des Kontextes beim ContextNode abgelaufen ist. Im dritten Fall würde eine Festlegung des Endzeitpunkts des alten Kontextes durch den ContextNode beim Eintreffen der Update-Nachricht oder des neuen Kontextes eine zeitliche Überlappung der Gültigkeit beider Kontexte ergeben. Aus diesem Grund wird der Endzeitpunkt des alten Kontextes auf den Startzeitpunkt des neuen Kontextes minus einer Zeiteinheit festgelegt. Eine Zeiteinheit muss abgezogen werden, da der Start- und Endzeitpunkt eines Kontextes Teil des Zeitintervalls seiner Gültigkeit sind und damit keine zeitliche Überschneidung der beiden Kontexte erzeugt wird. Dieser Vorgang setzt jedoch voraus, dass die Update-Nachricht bzw. der neue Kontext mit dem Startzeitpunkt propagiert wird. Dieses Attribut ist nicht Teil eines Matchings und befindet sich deshalb in der Kontextnachricht außerhalb der Attributliste.
In allen Fällen ist der ContextNode für die anschließende Archivierung des abgemeldeten Kontextes zuständig.

store(context)
get(attributeFilters)

Abbildung 4.3: Kontextdatenbank-API

Archivierung der Kontexte Ein Kontext erhält also vor der Archivierung ein präzises Zeitintervall vom zuständigen ContextNode vergeben, der den Zeitraum seiner Gültigkeit im System auszeichnet. Die API zur Kontext-Datenbank ist in Abbildung 4.3 dargestellt. Die Funktion „store(context)“ bekommt einen Kontext übergeben und speichert diesen in der Datenbank. Eine Liste von zu den Attributfiltern einer Nachricht passenden Kontexten kann mit der Funktion „get(attributeFilters)“ aus der Datenbank abgefragt werden. Die Informationen für die Attributfilter werden typischerweise aus der jeweiligen Nachricht entnommen. Für die Archivierung des Kontextes kann eine normale Datenbank verwendet werden. Der Zugriff auf diese Datenbank kann zum Beispiel unter Einsatz von SQL [KE06] erfolgen.

Eine zentrale Frage bei der Historisierung der Kontexte ist die Platzierung der Datenbank bzw. mehrerer Datenbanken. Wird von einer vollständigen Propagation der Kontexte im System ausgegangen, erhält jeder Knoten des Systems eine Kopie von jedem Kontext, der am System angemeldet wird. Werden die Löschanforderungen und Update-Nachrichten von Kontexten zusätzlich vollständig propagiert, so kann prinzipiell jeder Knoten im System von einem beliebigen ContextNode für die Einrichtung einer Kontext-Datenbank gewählt werden. Dem gewählten Knoten müsste vom ContextNode nur mitgeteilt werden, welchen ContextNode-ID er besitzt, damit die entsprechenden Kontexte des ContextNode identifiziert und archiviert werden können.

Werden Konzepte der Aggregation von Kontexten angewandt, werden unter Umständen nicht alle Kontexte im gesamten System propagiert. Dies schränkt die Menge an potentiellen Knoten für die Wahl der Positionen der Kontext-Datenbanken ein. Abschnitt 5.3 beschäftigt sich ausführlicher mit der Wahl der Positionen der Kontext-Datenbanken. Hier wird vorerst zur besseren Verständlichkeit ein einfacher Ansatz gewählt: Die Speicherung der historischen Kontexte erfolgt beim ContextNode selbst.

Nachdem ein ContextNode eine Abmeldung für einen von ihm verwalteten Kontext erhalten hat, kann nach der Propagation der Abmeldung und der Speicherung des Endzeitpunkts der Gültigkeit im Kontext eine Archivierung des Kontextes angestoßen werden. Dazu muss bei jedem ContextNode eine Datenbank vorhanden sein. Der zu archivierende Kontext wird beispielsweise in eine äquivalente SQL-Insert Anweisung umgeformt und auf die Datenbank angewendet. Die Umformung in eine SQL-Anweisung stellt keine große Herausforderung dar, da der Kontext selbst hauptsächlich aus einfachen Attribut-Wert-Paaren besteht. Das Zeitintervall des Kontextes, das nun genau festgelegt ist, wird ebenfalls abgespeichert. Mobile Clients sind unter Umständen nicht an zeitbezogenen Nachrichten oder Nachrichten, für die ihre TTL abgelaufen ist, interessiert. Für diese kann die Archivierung ihrer Kontexte entfallen.

Es wird darauf hingewiesen, dass die Archivierung eines Kontextes nicht gleichbedeutend damit ist, dass dieser Kontext älter ist als ein noch angemeldeter Kontext. Der Endzeitpunkt des Gültigkeitsintervalls für Kontexte steht in Relation mit dem Archivierungszeitpunkt. Ein noch angemeldeter Kontext kann also einen viel älteren Startzeitpunkt besitzen als ein bereits archivierter Kontext. Dieser Sachverhalt spielt für die Auswertung zeitbezogener Nachrichten eine große Rolle.

Nachdem entschieden wurde, dass historische Kontexte nicht im System propagiert, sondern direkt beim zuständigen ContextNode gespeichert werden, muss geklärt werden, wie ein Benutzer auf einen bereits archivierten Kontext zugreifen kann.

Ein Routing wie beim ursprünglichen Contextcast kann hier nicht stattfinden. Der einzige Zusammenhang zwischen dem Speicherort des archivierten Kontextes und seinem Inhalt ist in diesem Fall die Lokation. Damit ein Benutzer auf historische Kontexte zugreifen kann, muss er also zumindest eine Lokation vorgeben, für die er nach historischen Kontexten „suchen“ möchte. Basierend auf dieser Lokationsangabe kann die Nachricht dann zu den entsprechenden ContextNodes transportiert werden, deren Zugangsbereiche einen nicht-leeren Schnitt mit der vorgegebenen Lokation ergeben. Das Routing entspricht also einem Xcast ohne Matching innerhalb der ContextRouter. Für den Transport der Nachrichten werden nur die Empfängerlisten der Xcast-Nachricht betrachtet. Natürlich können auch Filter in den Nachrichten enthalten sein. Die Nachricht wird jedoch speziell markiert, so dass die Filter nicht durch die ContextRouter ausgewertet werden. Das eigentliche Matching der Filter auf historische Kontexte erfolgt erst bei den ContextNodes.
Die historischen Kontexte eines ContextNodes besitzen ausnahmslos Lokationen, die sich im Zugangsbereich des ContextNodes befinden. Da dieser Bereich nicht zwingend mit dem Schnittbereich der Lokation in der Nachricht und dem Zugangsbereich des ContextNodes übereinstimmt, muss der Attributfilter der Nachricht für die Lokation auf die Lokationen der Kontexte beim ContextNode geprüft werden.

Vorteile des Verfahrens der Speicherung der Kontexte bei den zuständigen ContextNodes betreffen zum einen den Nachrichtenaufwand. Da die Kontexte direkt beim zuständigen ContextNode gespeichert werden, entsteht kein Mehraufwand an Nachrichten zur Übermittlung des Kontextes an eine Datenbank, die sich auf einem anderen Knoten befindet. Wird die An- und Abmeldung der Kontexte mit den entsprechenden Start- und Endzeiten ihrer Gültigkeit im gesamten Netz propagiert, entsteht natürlich kein Mehraufwand, jedoch verliert dadurch der zuständige ContextNode seine eigentliche Zuständigkeit für die Archivierung des Kontextes. Bei der Anwendung von Aggregationskonzepten für Kontexten, die von vorne herein nicht ausgeschlossen werden sollen, kann eine vollständige Propagation nicht garantiert werden. Daher müssen die Kontexte entweder beim entsprechenden ContextNode gespeichert oder ein Mehraufwand an Nachrichten in Kauf genommen werden. Der Mehraufwand resultiert aus der Kommunikation des ContextNodes mit anderen Knoten bei der Auswahl einer geeigneten Position für die Kontextdatenbank.

Ein weiterer Vorteil entsteht durch die zentrale Speicherung beim ContextNode und damit dem zentralen Zugriff auf alle Kontexte eines ContextNodes. Werden die Kontexte verteilt gespeichert und ein Benutzer sendet eine „Anfrage“ an einen bestimmten ContextNode, so muss dieser im Zuge des Matchings eventuell auf all seine historisierten Kontexte zugreifen. Dies ist mit einem hohen Nachrichtenaufwand verbunden, wenn man sich vor Augen führt, dass die Anfrage zu jedem Datenbankknoten des ContextNodes übermittelt werden muss. Diese müssen anschließend noch die Ergebnisse des Matchings an den ContextNode zurücksenden.

Die verpflichtende Angabe der Lokation bei der Abfrage historischer Kontexte stellt keine Einschränkung bezüglich den spezifizierten Anforderungen dar. Es wird davon ausgegangen, dass lokationsgebundene Nachrichten im Allgemeinen die Regel darstellen und nicht lokationsgebundene zeitbezogene Nachrichten nur wenig Sinn ergeben. In diesem Zusammenhang stellt eine Speicherung der historischen Kontexte nahe beim zuständigen ContextNode bei Betrachtung des Nachrichtenaufwands sogar ein Vorteil dar. Werden in einzelnen Fällen nicht lokationsgebundene Nachrichten benötigt, so können diese durch lokationsgebundene Nachrichten simuliert werden, deren Zielgebiet ausreichend groß – im extremsten Fall den ganzen geographischen Einsatzbereich des Systems umfassend – ist.

ContextNodes, die sich bezüglich ihrer Zugangsbereiche nahe beieinander befinden, liegen auch im Overlay nahe beieinander und sind insgesamt vermehrt am Rand des Overlays angesiedelt. Eine Speicherung der Kontexte bei den ContextNodes und nicht bei den Contex-

tRoutern verlagert damit die Speicher- und Rechenbelastung des Netzes auf den Randbereich des Overlays. Dies entlastet die mittig angeordneten ContextRouter, die infolgedessen alle verfügbaren Kapazitäten für die Weiterleitung von Nachrichten verwenden können.

Natürlich stehen diesen Vorteilen gewisse Nachteile des Verfahrens gegenüber. Die zentrale Speicherung der Kontexte beim zuständigen ContextNode, welche als Vorteil bezüglich des Nachrichtenaufwands angesehen wird, stellt gleichzeitig den Hauptnachteil des Verfahrens dar. In Bezug auf verteilte Systeme weist die zentrale Speicherung von Daten meist auf einen Single-Point-of-Failure hin. Tatsächlich sind die historischen Kontexte eines ContextNodes im System nicht mehr verfügbar, falls der Knoten oder die Verbindung zum System durch den Abbruch der Verbindungen zu allen Nachbarknoten ausfällt. Ein alternativer Ansatz, der eine Lösung dieses Problems bereitstellt, muss zumindest Konzepte wie die Replikation der historischen Kontexte oder die Verteilung der Datenbanken auf mehrere Knoten in Betracht ziehen.

Ein weiterer Nachteil des gewählten Ansatzes liegt in der Lastverteilung. Gerade ContextNodes deren Benutzer ihre Kontexte häufig ändern oder bei denen eine hohe Fluktuation von An- und Abmeldungen von Kontexten durch Lokationswechsel der Benutzer besteht, weisen eine hohe Speicherlast auf. Die häufige Abfrage populärer historischer Kontexte erzeugt weiterhin eine ungleiche Verteilung der Rechenlast bei den ContextNodes. Ein ContextNode, der zum Beispiel die Kontexte von Benutzern in einem Stadion oder öffentlichen Gebäude speichert, ist im Regelfall einer höheren Anfragehäufigkeit ausgesetzt als ein ContextNode eines mittelständischen Unternehmens. Diese Unterschiede in der Anfragehäufigkeit erzeugen gleichzeitig eine ungleiche Lastverteilung des Rechenaufwands in den Knoten der näheren Umgebung eines ContextNodes, da alle Anfragen in Form von Nachrichten beim ContextNode münden. Die umliegenden ContextRouter müssen alle diese Nachrichten verarbeiten und weiterleiten.

Anfragen der Benutzer in Form von Nachrichten stellen natürlich keine Anfragen in dem Sinne dar, dass dem Benutzer die zu dem Filter der Nachricht passenden Kontexte übermittelt werden. Ansonsten hätte der Benutzer Zugriff auf alle Kontextinformationen des gesamten Systems. Dies wäre kritisch, da die Informationen der Kontexte im Zusammenhang mit anderen Kontexten beispielsweise Rückschlüsse auf die Entwicklung der Situation einer Personen ermöglichen. Beispiele hierfür sind das Tracking der Aufenthaltsorte einer Person oder Rückschlüsse auf seinen beruflichen Werdegang. Eine Abfragemöglichkeit von Kontexten durch Benutzer ist daher zum Schutz der Privatsphäre nicht gestattet. Der Zugriff auf historische Kontexte durch Nachrichten dient nur dem Routing der Nachricht zu den Empfängern, wie es in Abschnitt 4.5 ausführlich beschrieben wird.

Der folgende Abschnitt beschäftigt sich mit der Historisierung der Nachrichten eines zeitbezogenen Contextcast und vervollständigt damit die Konzepte zur Historisierung von Informationsobjekten.

4.2 Historisierung der Nachrichten

Eine erfolgreiche Übermittlung von Nachrichten im Contextcast setzt voraus, dass ein Empfänger einen zur Nachricht passenden Kontext am System angemeldet hat. Aus dieser Sicht heraus gilt ein Benutzer als nicht mit dem System verbunden, falls von ihm keine gültigen Kontexte im System existieren. Eine andere Art des Offline-Status ist die Abwesenheit einer physischen Verbindung zwischen dem Benutzer und dem ContextNode. Solange ein Benutzer mit mindestens einem Kontext am Contextcast-System angemeldet ist, wird von dem entsprechenden ContextNode ein Matching der eingehenden Nachrichten mit dem Kontext des Benutzers durchgeführt. Dem Benutzer können jedoch keine Nachrichten übermittelt werden, da ihm, wenn vielleicht auch nur für einen kurzen Zeitraum, eine physische Verbindung fehlt.

Dies betrifft wiederum hauptsächlich mobile Clients, da diese aus Gründen der beschränkten Ressourcenverfügbarkeit öfter vom System abgeschnitten sind als etwa stationäre Terminals. Aus diesem Grund wurde in [Gei07] eine TTL für Nachrichten eingeführt, für die zwar ein passender Kontext angemeldet, der Benutzer aber vorübergehend nicht erreichbar ist. Die TTL entspricht einem Zeitpunkt oder einem Zeitintervall, nach dessen Ablauf die Nachricht verworfen werden kann. Dieser Vorgang basierend auf der Annahme, dass der Benutzer für längere Zeit offline ist, oder sogar die Lokation gewechselt hat. Stellt der Benutzer vor Ablauf der TTL am gleichen Zugangspunkt erneut eine Verbindung mit dem System her, kann die Nachricht erfolgreich übermittelt werden.

Hier wird ein etwas erweiterter Ansatz gewählt. Die Repräsentation des Zeitpunkts in der Nachricht, ab dem die TTL abgelaufen ist, entspricht der Repräsentation der Zeitpunkte eines Zeitintervalls für die Gültigkeit eines Kontextes. Diese Repräsentation wird in Abschnitt 4.4 noch näher erläutert. Der ContextNode, der die Nachricht für einen nicht erreichbaren Benutzer zwischenspeichert, vergleicht diesen Zeitpunkt periodisch mit seiner eigenen Uhr. Hier müssen synchrone Uhren Voraussetzung sein. Nach Ablauf der TTL der Nachricht oder der TTL des entsprechenden Kontextes bzw. der entsprechenden Kontexte wird die Nachricht entweder verworfen oder archiviert. Hierüber entscheidet ein spezielles *Persistenzattribut*[4], das in jeder Nachricht vorhanden sein muss und vom Benutzer gewählt werden kann. Es ist nicht Teil des Filters und auch nur für ContextNodes interessant, die einen zur Nachricht passenden Kontext eines angemeldeten und nicht erreichbaren Benutzers besitzen. Die ContextRouter können dieses Attribut getrost ignorieren.

Falls ein Benutzer im Rahmen eines Roaming einen Lokationswechsel vor Ablauf der TTL der Nachricht durchgeführt hat, kann die Nachricht vom ContextNode bei Erhalt der Update-Nachricht an den neuen ContextNode des Benutzers weitergeleitet werden, welcher dann für die Zustellung der Nachricht verantwortlich ist. Um den Benutzer anhand seines Kontext eindeutig identifizieren zu können, muss ein eindeutiger benutzerspezifischer *Client-ID* in jedem Kontext vorhanden sein. Dieser wird im Falle der Weiterleitung für das Routing verwendet, so dass eine erneute Auswertung der Nachricht auf den neu angemeldeten Kontext umgangen wird. Dadurch wird sichergestellt, dass die Nachricht trotz einer möglichen Änderung des Kontextes durch die Update-Nachricht an den Benutzer zugestellt wird, obwohl der neue Kontext eventuell nicht mehr zu der Nachricht passt. Die TTL der Nachricht wird bei der Weiterleitung nicht verändert. Der neue ContextNode wendet für die Zustellung der Nachricht die gleiche Vorgehensweise an wie der ursprüngliche ContextNode. Die Weiterleitung der Nachricht findet nur dann statt, wenn der neue Kontext des Benutzers in regulärer, d.h. nicht aggregierter Form bei dem ursprünglichen ContextNode eintrifft. Der Grund dafür ist, dass ein aggregierter Kontext nicht genau einem Benutzer zugeordnet werden kann.

Das Persistenzattribut einer Nachricht gibt an, ob eine Nachricht auch bei einer längeren Abwesenheit eines Benutzers vom System zugestellt werden soll. Es ist ein boolesches Attribut mit dem Wert `true`, falls die Nachricht persistent gehalten werden soll, andernfalls `false`. Falls das Persistenzattribut einer Nachricht auf *true* steht und die TTL der Nachricht bei einem ContextNode abläuft, wird die Nachricht vom ContextNode nicht verworfen. Die Nachricht muss nun solange persistent gespeichert werden, bis der entsprechende Empfänger wieder im System erreichbar ist. Grundsätzlich könnte der ContextNode die Nachricht speichern, bis ein neuer Kontext mit dem ID des Benutzers im System angemeldet wird. Anschließend würde der ContextNode die Nachricht an den neuen ContextNode des Benutzers weiterleiten.
Dieses Vorgehen wäre gleichbedeutend mit der Vorgehensweise der ContextNodes bei der Behandlung von Nachrichten, deren TTL zusammen mit der TTL der Kontexte auf einen

[4] Der Begriff Persistenz bedeutet hier die persistente Speicherung der Nachricht bis zur Auslieferung an die Empfänger

unendlichen Wert gesetzt ist. Unter diesen Umständen darf sich der ID des Benutzers niemals ändern. Nach einer Änderung der ID könnten dem Benutzer zwischengespeicherte Nachrichten nicht mehr zugeordnet werden und verbleiben anschließend als Speicherleichen im Netz. Die Verwendung von konstanten, eindeutigen Client-IDs erlaubt jedoch Angriffsmöglichkeiten, die eindeutig in die Privatsphäre von Benutzern des Systems eingreifen. Ein Angreifer mit Zugriff auf die Kontextdatenbank eines oder mehrerer ContextNodes kann beispielsweise aufgrund der eindeutigen Benutzer-IDs in den Kontexten die Aufenthaltsorte, Anmeldezeiten und Veränderungen in den Kontextinformationen gezielter Benutzer rückverfolgen. Selbst wenn die Benutzer-IDs nicht direkt auf Personen schließen lassen, da sie zum Beispiel aus zufällig gewählten Zahlen zusammengesetzt sind, kann der Angreifer durch Kenntnisse verschiedener Aufenthaltsorte und die Person beschreibende Informationen die Menge der IDs stark einschränken, die möglicherweise auf den gesuchten Benutzer zutreffen. Zum Schutz der Privatsphäre der Benutzer des Contextcast wird daher für die Archivierung von Nachrichten ein spezielles Mailboxsystem verwendet. Wie die Speicherung der Nachrichten in Mailboxen und die Verwaltung der Client-IDs gehandhabt wird und damit die genannten Angriffsmöglichkeiten verhindert werden, wird im folgenden Abschnitt geklärt.

Mailboxsystem und Trusted Node Bevor die Speicherung von Nachrichten in einem Mailboxsystem erläutert wird, sollte nochmals festgehalten werden, dass diese Methode eventuell keine Zulieferung der Nachrichten an alle gewollten Empfänger garantiert, selbst wenn das System korrekt arbeitet und sich in einem konsistenten Zustand befindet. Dies liegt daran, dass die Nachrichten nur für Empfänger gespeichert werden, für die nach Ablauf der Nachrichten-TTL eine Übermittlung nicht stattgefunden hat. Sie sind vor allem aber nur für Empfänger bestimmt, die einen zur Nachricht passenden Kontext angemeldet haben. Meldet ein Benutzer kurz vor dem Eintreffen der Nachricht beim ContextNode seinen Kontext ab oder einen neuen Kontext erst kurz nach dem Eintreffen der Nachricht an, so wird er die Nachricht nicht erhalten, da zu dem Zeitpunkt an dem die Nachricht eintrifft kein passender Kontext des Benutzers angemeldet ist. Sind auch keine passenden Kontexte anderer Benutzer angemeldet, so erhält nicht einmal der ContextNode die Nachricht. Auch mobile Clients, die es verpassten innerhalb der vom ContextNode für die Kontexte festgelegten TTL ihre angemeldeten Kontexte aufzufrischen, gelten nach Zwangsabmeldung ihrer Kontexte als vom System abgemeldet und erhalten keine Nachrichten mehr.

Jeder Benutzer des Contextcast-Systems muss vor der ersten Nutzung des Systems einen Mailboxaccount beantragen. Dazu stellt das Contextcast-System ContextNodes bereit, die ein Mailboxsystem verwalten und für Benutzer anbieten. Je nach Umsetzung kann ein Mailboxsystem öffentlich oder nur für eine bestimmte Benutzergruppe angeboten werden. Es ist durchaus vorstellbar, dass die verschiedenen *Internet Service Provider* (ISPs) ihren Klienten vorab einen Mailboxaccount einrichten oder anbieten. Der Benutzer muss dem Mailboxanbieter jedenfalls vertrauen können, da dieser gleichzeitig als Trusted Node fungiert. Ein Trusted Node im Contextcast vergibt den Benutzern ihre eindeutigen Client-IDs. Da der ID im gesamten System eindeutig sein müssen, setzt sich der Client-ID eines Benutzers aus der für den Trusted Node lokal eindeutigen ID für den Benutzer – im Weiteren Benutzer-ID oder lokaler Client-ID genannt – und der ID des Trusted Node selbst zusammen. Dies entspricht einer zweistufigen Hierarchie für Client-IDs. Falls ein Benutzer beispielsweise von einem Trusted Node den ID $R171$ erhalten hat und der ID des Trusted Nodes selbst $SC57$ lautet, entspricht der Client-ID des Benutzers dem Bezeichner $R171@SC57$. Die IDs der Trusted Nodes selbst müssen ebenfalls systemweit eindeutig sein. Die Vergabe der IDs für Trusted Nodes kann ähnlich zu der Namensvergabe für das DNS-System im Internet gehandhabt werden. Diese werden durch eine zentrale Stelle vergeben. Eine andere, verteilte Möglichkeit ist die zufällige Wahl einer hinreichend langen Buchstaben/Zahlen-Folge, so dass die Wahrscheinlichkeit

store(messageContent,mailboxID)
getMsgs(mailboxID)

Abbildung 4.4: Mailbox-API

für eine Kollision in den unabhängig gewählten IDs der Trusted Nodes gering ist. Der Trusted Node muss seinen ID als Kontextinformation im System propagieren. Zusammen mit weiteren Attributen kann dieser Kontext auch durch den Benutzer bei einer erstmaligen Anmeldung zum Auffinden eines Trusted Nodes verwendet werden. Weiterhin können durch die Propagation der Kontexte Kollisionen der Trusted-Node-IDs schnell entdeckt und behoben werden.
Die endgültige Wahl der Methode zur Vergabe der Trusted-Node-IDs ist Teil der konkreten Implementierung eines zeitbezogenen Contextcast-Systems. Wie schon im vorherigen Abschnitt erläutert wurde, ist der eindeutige Client-ID eine Schwachstelle des Contextcast für die Privatsphäre des Benutzers. Aus diesem Grund hat der Benutzer die Möglichkeit neue Benutzer-IDs bei seinem Trusted Node einzuholen. In welchen zeitlichen Abständen dies geschieht, kann vom Benutzer zum Beispiel in Relation zu der Frequenz von ihm neu angemeldeter Kontexte gewählt werden. Das an dieser Stelle verwendete Konzept zur Wahrung der Anonymität eines Benutzers entspricht der Verwendung von *Virtual Identifier* (VIDs) in [WBS+05]. Der neue ID wird vom Trusted Node ausgewählt, mit einem für den Trusted Node eindeutigem REGID des Benutzers verknüpft und dem Benutzer übermittelt. Da nur der betreffende Trusted Node die genaue Verknüpfung von Benutzer-IDs und REGIDs kennt, kann nur über ihn ein Client-ID einer konkreten Person zugeordnet werden.
Die Verwendung der genannten IDs für das Routing von Nachrichten wird in den folgenden Abschnitten erläutert.

Muss eine Nachricht mit auf *true* gesetztem Persistenzattribut abgespeichert werden, so kann aus dem passenden Kontext des Empfängers nach Ablauf der TTL der Nachricht der Client-ID des Empfängers entnommen werden. Für die Übermittlung der Nachricht an das Mailboxsystem durch den ContextNode ist nur der ID des Trusted Nodes aus dem Client-ID von Bedeutung. Da jeder Trusted Node seinen ID im System durch Propagation eines Kontextes bekannt macht, kann die Nachricht durch Xcast vom ContextNode aus an den Trusted Node übermittelt werden. Dieser kann nun aus dem vollständigen Client-ID heraus den Benutzer-ID auslesen, diese dem Account des Benutzers zuordnen und die Nachricht im Mailboxsystem für den Benutzer ablegen. Abbildung 4.4 zeigt die allgemeine API für den Zugriff auf das Mailboxsystem. Mit der Funktion „store(messageContent,mailboxID)“ können Nachrichten in die lokale Mailbox des Benutzers abgelegt werden. Die ID für die Mailbox des Benutzers kann vom Trusted Node erfragt werden. Durch die Funktion „getMsgs(mailboxID)“ kann der Benutzer Nachrichten aus seiner Mailbox abrufen. Diese Funktion muss bei einer konkreten Implementierung des Systems durch Authentifizierungsverfahren vor fremdem Zugriff geschützt werden. Das Mailboxsystem selbst ist wie die Kontextdatenbank eine normale Datenbank, auf die zum Beispiel über SQL zugegriffen werden kann.

Im Unterschied zu den Kontexten müssen für Nachrichten keine Attribute oder Attributfilter abgespeichert werden, da ein Matching bereits erfolgt ist. Der Benutzer ist lediglich an den Nachrichteninhalten interessiert. Das Entfernen der für den Benutzer unnötigen Informationen kann schon bei den ContextNodes vor der Weiterleitung an die Trusted Nodes erfolgen. Sobald ein Benutzer wieder mit dem System verbunden ist, kann das Mailboxsystem die gespeicherten Nachrichten an den neuen Standort des Benutzer übermitteln. Selbst wenn der Benutzer in Zwischenzeit seinen ID gewechselt hat, ist durch die im Trusted Node gespeicherte Kette von lokalen Client-IDs des Benutzers eine Identifikation möglich. Der

neue Standort des Benutzers kann durch den Trusted Node aus dem neuen Kontext in Form eines ContextNode-IDs ausgelesen werden. Alternativ kann der Benutzer bei jeder Neuanmeldung am System direkt seine Mailbox auf Nachrichten prüfen und diese gegebenenfalls über ein Datentransferprotokoll „abholen“. Prinzipiell könnten der Trusted Node und das Mailboxsystem eines Benutzers auf unterschiedlichen Knoten angesiedelt sein. Da jedoch zu historisierende Nachrichten zuerst an den Trusted Node gesendet werden, ist die Vereinigung von Trusted Node und Mailbox auf einem Knoten bezüglich des Nachrichtenaufwands von Vorteil.

Durch die Möglichkeit des Benutzers seinen ID zu ändern, kann ein Angreifer nicht entscheiden, ob bestimmte Kontexte zu ein und derselben Identität gehören. Lediglich durch den sich nicht ändernden Trusted-Node-ID ist eine Eingrenzung möglich. Falls jedoch viele Benutzer den gleichen Trusted Node wählen, ist eine eindeutige Zuordnung der Kontexte zu Benutzern kaum zu realisieren. Dies ist vergleichbar mit der Verwendung von Proxy-Servern im Internet zur Verschleierung der eigenen Identität. Da also nur die Trusted Nodes eine Zuordnung von Client-IDs einer Nachricht zu den aktuellen Client-IDs und Mailboxaccounts der Benutzer durchführen können, müssen die zu archivierenden Nachrichten zuerst an jene weitergeleitet werden. Anschließend kann eine Archivierung in Mailboxen erfolgen.
Grundsätzlich ist auch ein Wechsel des Trusted Nodes eines Benutzers im Laufe der Zeit oder die Anmeldung des Benutzers an mehreren Trusted Nodes möglich. Eine vertiefte Auseinandersetzung mit dieser Möglichkeit bleibt der konkreten Implementierung überlassen.

Eine verlässliche Übermittlung der Nachrichten kann durch die Bestätigung der Nachrichten durch den Empfänger in Form von *Acknowledgement*-Nachrichten erreicht werden. Ist der Trusted Node, zum Beispiel eines ISPs, über eine ausreichend stabile Leitung mit dem Contextcast-System verbunden und werden die Nachrichten im Mailboxsystem persistent gespeichert, so kann auch von einer ausreichenden Persistenz der Nachrichten ausgegangen werden.

In den letzten zwei Abschnitten wurden die zentralen Konzepte zur Archivierung der Informationsobjekte eines zeitbezogenen Contextcast erläutert. Bevor das Routing der zeitbezogenen Nachrichten behandelt werden kann, müssen noch einige Details, wie beispielsweise die Repräsentation der zeitbezogenen Nachrichten, geklärt werden. Die folgenden Abschnitte befassen sich mit den unterschiedlichen Kategorien zeitbezogener Nachrichten, mit den zeitlichen Attributfiltern zur Auswahl historischer Kontexte, deren interne Repräsentation und der Semantik der Zeitstempel.

4.3 Zeitbezogene Nachrichten

Im ursprünglichen Contextcast kann mit den Nachrichten nur Bezug auf aktuell angemeldete Kontexte genommen werden. Nur wenn mindestens ein zur Nachricht passender Kontext zur Ankunftszeit der Nachricht bei den ContextNodes angemeldet ist, wird die Nachricht zu den Absendern der passenden Kontexte übermittelt. Zeitbezogene Nachrichten versuchen diese Einschränkung des Systems aufzuheben. Im Gegensatz zu den regulären Nachrichten sind zeitbezogene Nachrichten durch ein spezielles Zeitintervall gekennzeichnet. Reguläre Nachrichten bleiben neben den zeitbezogenen Nachrichten weiterhin bestehen.

Das Zeitintervall einer Nachricht gehört zu einem zusätzlicher Filter, welcher sich auf das Zeitintervall, also den Start- und Endzeitpunkt der Gültigkeitsdauer, eines Kontextes bezieht. Dadurch lassen sich Nachrichten erzeugen, die auch über einen längeren Zeitraum hinweg auf Kontexte ausgewertet werden können oder archiviert Kontexte adressieren. Unter Betrachtung des Zeitbezugs lassen sich zeitbezogene Nachrichten in vier Kategorien einteilen:

1. **Historische Nachrichten:** Die Nachricht bezieht sich über ihr Zeitintervall und den

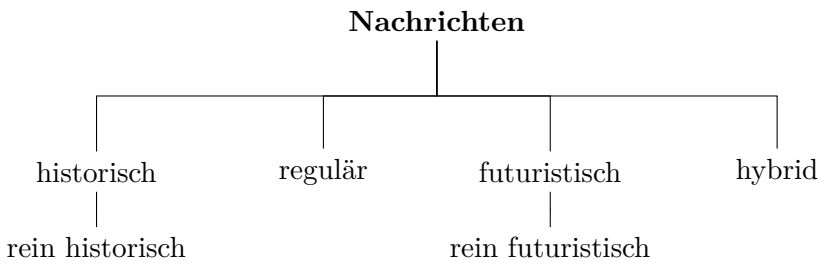


Abbildung 4.5: Nachrichtenkategorien eines zeitbezogenen Contextcast

Operator des Attributfilters ausschließlich auf historische und gegenwärtige Kontexte. Die historischen Kontexte wurden bereits in den Kontextdatenbanken archiviert. Gegenwärtige Kontexte sind daher interessant, da deren Startzeitpunkte auch in der Vergangenheit liegen.

2. **Gegenwärtige/Reguläre Nachrichten:** Nachrichten des ursprünglichen Contextcast. Sie besitzen kein Attributfilter für Zeitintervalle der Kontexte und beziehen sich nur auf aktuell angemeldete Kontexte.

3. **Futuristische Nachrichten:** Diese Nachrichten beziehen sich nur auf gegenwärtig angemeldete und zukünftige Kontexte. Ein Matching mit historischen Kontexten muss nicht durchgeführt werden. Für gegenwärtige Kontexte muss der Endzeitpunkt abgewartet werden, bevor ein Matching erfolgen kann.

4. **Hybride Nachrichten:** Hybride Nachrichten stellen eine Mischung von historischen und futuristischen Nachrichten dar. Ein Matching muss sowohl auf historische, wie auch gegenwärtige und eventuell auch zukünftige Kontexte durchgeführt werden.

Mit diesen Nachrichten wird ermöglicht, vergangene Informationen über die Situationen von Empfängern in die Nachricht einfließen zu lassen oder Nachrichten für zukünftige Kontexte zu hinterlegen. Historische und futuristische Nachrichten lassen sich weiter in rein historische und rein futuristische Nachrichten untergliedern. Abbildung 4.5 gibt einen Überblick über die verschiedenen Nachrichtenkategorien eines zeitbezogenen Contextcast.
Rein historische Nachrichten werden an Empfänger übermittelt, deren von der Nachricht adressierter Kontext in der Vergangenheit liegt und bereits archiviert wurde. Rein futuristische Nachrichten hingegen stellen das Pendant zu Subscriptions in Publish/Subscribe-Systemen dar. Sie können als eine Art Anmeldung für zukünftige Kontexte verstanden werden.
Rein futuristische Nachrichten werden an alle Empfänger übermittelt, die in der Zukunft Kontexte anmelden, die zu der Nachricht passen. Dazu müssen sie solange im Netz gespeichert werden, bis der zeitliche Attributfilter kein erfolgreiches Matching mehr mit zukünftigen Kontexten zulässt. Danach können sie verworfen werden. Bei rein historischen und rein futuristischen Nachrichten spielen also gegenwärtig angemeldete Kontexte keine Rolle. Gegenwärtige Nachrichten werden wie gehabt nach dem ursprünglichen Routing des Contextcast an die Empfänger übermittelt. Bei hybriden Nachrichten wird eine Kombination der Verfahren für historische und zukünftige Nachrichten benötigt.

Eine Einteilung der Nachrichten eines zeitbezogenen Contextcast in die verschiedenen Kategorien wird nicht explizit in den Nachrichten, z.B. durch ein spezielles FLAG[5], kenntlich

[5] Zusätzliches Wertefeld in der Nachricht

gemacht. Sie stellt lediglich eine logische Einteilung der Nachrichten dar. Anhand dieser Einteilung kann für die Nachrichten der einzelnen Kategorien ein geeignetes Verfahren zum Routing und Matching gewählt werden. Dies beinhaltet die Entscheidung, welche Kontexte, z.B. bei historischen Nachrichten, für das Matching in Betracht gezogen werden müssen. Eine Einteilung erfolgt durch die Auswertung des zeitlichen Attributfilters der Nachricht und der aktuellen Zeiten der ContextNodes, die in der Empfängerliste der Nachricht enthalten sind. Für die Auswertung ist neben dem vorgegebenen Zeitintervall des Attributfilters auch der Operator entscheidend. Für eine intensive Betrachtung der Verfahren für das Routing und speziell dem Matching der verschiedenen zeitbezogenen Nachrichten müssen weiterhin die möglichen Operatoren für Zeitintervalle festgelegt, die Bedeutung der Zeitdarstellung geklärt und der Datentyp für das Zeitintervall des Attributfilters konkretisiert werden. Dies erfolgt im folgenden Abschnitt nach einer kurzen Wiederholung der Relevanz von Zeitangaben für Kontexte und Nachrichten.

4.4 Datentyp für zeitbezogene Angaben

Für die Zeitangaben der Kontexte und Nachrichten werden Zeitintervalle verwendet. Zeitintervalle können durch ihre zeitlichen Eckpunkte definiert werden. Sie bestehen also aus einem Start- und Endzeitpunkt, welche beide Bestandteil des Intervalls sind und gemäß der *Universal Coordinated Time*[6] (UTC) angegeben werden. Dabei muss festgelegt werden, dass der Startzeitpunkt kleiner als oder gleich dem Endzeitpunkt sein muss. Es besteht weiterhin ein Unterschied zwischen der internen Repräsentation der Zeitintervalle für Kontexte und für Nachrichten. Der Start- und Endzeitpunkt der Zeitintervalle für Kontexte wird durch den ContextNode des jeweiligen Kontext exakt festgelegt. Als Startzeitpunkt gilt der Zeitpunkt, in welchem sich der Benutzer des Kontextes an dem ContextNode angemeldet hat. Der Endzeitpunkt entspricht dem Zeitpunkt, an dem der betreffende Kontext abgemeldet wurde. Die verschiedenen Ursachen für eine Abmeldung eines Kontextes wurden bereits in Abschnitt 4.1 geschildert. Jeder historische Kontext hat also ein exakt festgelegtes Zeitintervall. Dieses hält den Zeitraum fest, während dem der Kontext im System gültig war. Die Zeitangaben der Kontextnachrichten verfolgen hingegen den Zweck, eine Menge dieser Kontexte auszuwählen. Dazu wird analog zu den anderen Attributfiltern der Nachricht ein Attributfilter für den Gültigkeitszeitraum des Kontextes definiert. Der Attributfilter für den Gültigkeitszeitraum besteht aus einem, gleichnamig zum Gültigkeitszeitraum, fest gewählten Attributnamen n_{temp}, einer Operation $op \in Op_{temp}$ und einem Tripel $(a, b, z) \in C_{temp}$. Die Werte für a und b liegen in der Menge der möglichen Zeitangaben entsprechend der UTC. Die Werte für z legen einen Zeitraum fest und haben stets ein positives Vorzeichen. Die kleinste Einheit dieser Zeitspanne richtet sich nach der Basis der Zeitmessung für die UTC, der Sekunde. Werte für z werden für Ungenauigkeiten in den Zeitangaben verwendet. Die Elemente der Menge Op_{temp} definieren binäre Operatoren auf Zeitintervalle. In Tabelle 4.2 sind die verfügbaren Operatoren für Zeitintervalle des in dieser Arbeit beschriebenen zeitbezogenen Contextcast aufgelistet. Diese wurden in abgeänderter Form aus [All83] entnommen und um weitere nützliche Operatoren ergänzt. Die Ergänzung der Zeitintervalle um den Wert z ermöglicht eine Toleranz bei dem Vergleich zweier konkreter Zeitpunkte im Rahmen der Auswertung eines Operators auf zwei Zeitintervalle. Die Auseinandersetzung mit zeitlichen Intervallen in [All83] geht weit über die Anforderungen des zeitbezogenen Contextcast hinaus. Konkrete Zeitintervalle mit einem Start- und Endzeitpunkt sind an dieser Stelle völlig ausreichend. Komplexere Beziehungen zwischen Zeitintervallen werden nicht benötigt.

Auf der rechten Seite der Äquivalenzpfeile sind die Bedingungen für ein erfolgreiches Matching von *AFmatch* eines zeitbezogenen Attributfilters auf ein Zeitintervall für jede mögliche

[6] Koordinierte Weltzeit

before:	$(a,b,z) < [x,y]$	$\Leftrightarrow$	$b < x$				
after:	$(a,b,z) > [x,y]$	$\Leftrightarrow$	$y < a$				
starts-only:	$(a,b,z) <^{so} [x,y]$	$\Leftrightarrow$	$x \leq a$	$\wedge$	$a \leq y$	$\wedge$	$y \leq b$
ends-only:	$(a,b,z) >^{eo} [x,y]$	$\Leftrightarrow$	$a \leq x$	$\wedge$	$x \leq b$	$\wedge$	$b \leq y$
start-by:	$(a,b,z) <^{s} [x,y]$	$\Leftrightarrow$	$a \leq x$	$\wedge$	$x \leq b$		
end-by:	$(a,b,z) >^{e} [x,y]$	$\Leftrightarrow$	$a \leq y$	$\wedge$	$y \leq b$		
overlaps:	$(a,b,z) \cap [x,y]$	$\Leftrightarrow$	$a \leq y$	$\wedge$	$x \leq b$		
excludes:	$(a,b,z) \neq [x,y]$	$\Leftrightarrow$	$b < x$	$\vee$	$y < a$		
equals:	$(a,b,z) \approx [x,y]$	$\Leftrightarrow$	$\lvert a-x\rvert \leq z$	$\wedge$	$\lvert b-y\rvert \leq z$		
during:	$(a,b,z) \subseteq [x,y]$	$\Leftrightarrow$	$x \leq a$	$\wedge$	$b \leq y$		
contains:	$(a,b,z) \supseteq [x,y]$	$\Leftrightarrow$	$a \leq x$	$\wedge$	$y \leq b$		

Tabelle 4.2: Operatoren und Auswertungsregeln für Zeitintervalle

Operation angegeben. Diese Bedingungen können so direkt in die Implementierung des Matchings von zeitlichen Attributfiltern auf Zeitintervalle von Kontexten übernommen werden. Je nach Anforderung der Anwendung an das System können auch weitere Operatoren definiert und in die Implementierung des Systems aufgenommen werden. Die abgebildete Liste von Operatoren ist keineswegs vollständig, reicht jedoch für die grundlegenden Bedürfnisse der zeitlichen Adressierung von Kontexten aus.

Die in Tabelle 4.2 verwendeten Intervalle $[x,y]$ sind feste Zeitintervalle für Kontexte. Die Variablen a und b stehen für den Start- und Endzeitpunkt des Zeitintervalls des Attributfilters. Es gilt also ebenfalls $a \leq b$. Für die Werte von a, b und z kann in einem Attributfilter ein Platzhalter ? verwendet werden. Dies ist nur möglich, falls das entsprechende Tupel-Element keine Auswirkung auf die Auswertung des Attributfilters hat, die *boolesche Differenz* also gleich Null ist. Dieser Sachverhalt ist wie folgt definiert:

Definition 4.4.1

$$op_{x_i}(x_1,x_2,x_3,x,y) = 0, i \in (1,2,3), x_i \in X$$
$$\Leftrightarrow \forall v,u \in X, v \neq u : op(x_1,\ldots,v_i,\ldots,x_3,x,y) = op(x_1,\ldots,u_i,\ldots,x_3,x,y)$$

Hierbei bezeichnet $op_{x_i}(x_1,x_2,x_3,x,y)$ die boolesche Differenz bezüglich des Operators op und der Variable x_i. Die Variablen x_1, x_2 und x_3 stehen für die in der Definition der Operationen gewählten Variablen a, b und z. Bei Anwendung der Operation auf das Tupel (a,b,z) und das Kontextzeitintervall $[x,y]$ muss sich, damit die boolesche Differenz gleich Null ist, für beliebige Werte des gewählten Tupel-Elements für den Platzhalter immer der gleiche boolesche Wert ergeben. Äquivalent dazu ist die Wahl einer Variablen aus (a,b,z) für den Platzhalter, welche bei der Definition der zugehörigen Operation des Attributfilters *nicht* auf der rechten Seite des Äquivalenzsymbols steht.

Koordinierte Weltzeit und lokale Abweichungen Knoten eines Contextcast sind möglicherweise über die ganze Welt verstreut. Bezüglich der Zeitrepräsentation in den Kontexten und Nachrichten eines Contextcast bringt dieser Sachverhalt einige Probleme mit sich. Lokale Zeiten werden unter den verschiedensten Regeln korrigiert und damit verändert. Damit

eine Interoperabilität zwischen den verschiedenen Knoten unterschiedlicher Regionen garantiert werden kann, wird daher in dem in dieser Arbeit beschriebenen System die UTC für Zeitangaben verwendet. Ausgehend von der UTC sind die Zeiten der verschiedenen Zeitzonen der Erde durch Abweichungen von der UTC angegeben. Die UTC zusammen mit der Abweichung einer Zeitzone ergibt die *Local Time* (LOT) einer Zeitzone. Die Verwendung der UTC und ihrer Zeitzonen hat Auswirkungen auf die Semantik der Nachrichtenübermittlung des zeitbezogenen Contextcast.

Die Verwendung der LOT kommt dem intuitiven Gebrauch von Zeitangaben am nächsten. Aus diesem Grund wird für die Zeitangaben von Kontexten und Nachrichten die LOT verwendet. Bei dem Versand einer zeitbezogenen Nachricht werden die Zeitpunkte des von dem User angegebenen Zeitintervalls als in LOT angegebene Zeitpunkte interpretiert. Die Zeitangaben der Nachricht werden für die weitere Verwendung durch den ContextNode, an welchem der Absender der Nachricht angemeldet ist und von dem aus die Nachricht verschickt werden soll, in die UTC-Repräsentation umgewandelt. Eine ähnliche Vorgehensweise gilt für die Vergabe der Zeitpunkte für angemeldete Kontexte. Auch hier werden die Zeitangaben vom ContextNode in der UTC-Repräsentation vergeben. Verwendet wird dazu die LOT des betreffenden ContextNode. Dies setzt voraus, dass die UTC jedes ContextNodes des Contextcast hinreichend gut synchronisiert sind und dass jeder ContextNode weiß, in welcher Zeitzone er sich befindet und damit die Abweichung der LOT von der UTC kennt. Der Benutzer selbst muss demnach weder die momentane UTC kennen, noch die lokale Abweichung der Zeitzone, in der er sich befindet. Das Problem der synchronen Uhren bei den ContextNodes kann durch die Anwendung des NTP gelöst werden. Man erhält die UTC aus der LOT, indem man von der LOT die lokale Abweichung subtrahiert. Umgekehrt erhält man aus der UTC die LOT, indem man die Abweichung addiert. Abweichungen[7] von der UTC werden nach [KN02] durch ein Vorzeichen und vier Dezimalzahlen angegeben. Die ersten zwei Dezimalzahlen geben die Stunden vor und werden durch einen Doppelpunkt von der Angabe der Minuten getrennt. Die genaue formale Repräsentation der UTC und der LOT können in [KN02] als ABNF-Beschreibung nachgeschlagen werden.

Zeitstempel (in UTC):	YYYY-MM-DD'T'HH:MM:SS'Z'
Lokale Abweichung:	'+'|'-'HH:MM

Die Auswirkungen bei der Verwendung der UTC mit ihren Zeitzonen lässt sich am einfachsten anhand eines Beispiels veranschaulichen. Angenommen zwei Benutzer c_1 und c_2 in einem Contextcast befinden sich in unterschiedlichen Zeitzonen. Der Benutzer c_1 befindet sich in Stuttgart und der Benutzer c_2 in Chicago. Benutzer c_2 ist an einem ContextNode mit einer UTC-Abweichung von -06:00 mit einem Kontext angemeldet, der abgesehen von dem Attributfilter für das Zeitintervall von einer Nachricht des ContextNode von Benutzer c_1, mit einer UTC-Abweichung von +01:00, getroffen wird. Die aktuelle UTC sei 2008-02-22T12:00:00Z. Damit ergeben sich die lokalen Zeiten 13:00:00 für c_1 und 06:00:00 für c_2. Falls die Nachricht von c_1 an einen Benutzer gerichtet ist, dessen Kontext vor dem Zeitpunkt 07:00:00 angemeldet wurde, so wird c_2 diese Nachricht nicht erhalten, davon ausgegangen, dass sich c_2 das erste Mal an seinem ContextNode um 06:00:00 angemeldet hat. Der Grund dafür liegt nahe. Beim Versand der Nachricht durch den ContextNode von c_1 wird die UTC 2008-02-22T06:00:00Z für den zeitlichen Attributfilter der Nachricht festgelegt. Erhält der ContextNode von c_2 diese Nachricht, so wird die UTC des zeitlichen Attributfilters der Nachricht mit der UTC-Anmeldezeit des Kontextes von c_2, nämlich 2008-02-22T12:00:00Z verglichen. Der Knoten c_2 hätte sich also schon vor 00:00:00 Uhr lokaler Zeit in Chicago anmelden müssen, um die Nachricht zu erhalten.

[7] Offsets

Die Zeit, die ein sendender Benutzer für seine Nachrichten festlegt, bezieht sich also immer auf die lokale Zeit seines aktuellen ContextNode. Dieser nimmt die Nachricht entgegen und korrigiert die angegebenen Zeitpunkte des Zeitintervalls um die lokale Abweichung, woraus die UTC entsteht. Die Auswertung des zeitlichen Attributfilters erfolgt bei den ContextNode-Empfängern der Nachricht auf die Kontexte, die vorerst bis auf den zeitlichen Attributfilter zur Nachricht passen. Die Zeitangaben der Kontexte werden von den ContextNodes direkt als UTC vergeben. Beim Matching müssen somit keine lokale Abweichungen mehr betrachtet werden. Bei der Korrektur der Zeitpunkte einer Nachricht um lokale Abweichungen und bei der Vergabe des Gültigkeitszeitraums eines Kontextes wird davon ausgegangen, dass die Zeitzonen eines Benutzers und dessen ContextNode übereinstimmen. Damit wird eine gewisse Lokalität der Knoten im Contextcast vorausgesetzt.

4.5 Routing und Matching

An dem bisher beschriebenen Routingverfahren für reguläre Nachrichten muss durch die Einführung zeitbezogener Nachrichten keine Änderung vorgenommen werden. Der Benutzer selbst entscheidet darüber, ob er eine reguläre Nachricht, ohne einen auf die Gültigkeitsdauer der Kontexte bezogenen zeitlichen Attributfilter, oder eine zeitbezogene Nachricht verfassen möchte.
Der Datentyp für Zeitintervalle, die Repräsentation der Zeitstempel und deren Bedeutung, sowie die Komponenten eines auf ein Zeitintervall bezogenen Attributfilters wurden bereits in den vorherigen Abschnitten dieser Arbeit erläutert. Für die einzelnen Operatoren wurden bereits die Auswertungsregeln für ein Kontextmatching angegeben. In diesem Abschnitt ist das Routing und die spezielle Behandlung der zeitbezogenen Nachrichten beim Matching ausführlich dargestellt.

Das Matching der Nachrichten auf die Kontexte selbst wird durch die Auswertung der Funktion *Match* (nach den Regeln aus Abschnitt 2.4) durchgeführt. Die Regeln für die Auswertung der Funktion *AFmatch* auf einen zeitbezogenen Attributfilter einer Nachricht und ein Zeitintervall eines Kontextes können der Tabelle 4.2 entnommen werden.

Zeitbezogene Nachrichten werden allgemein über ein Xcast (siehe Abschnitt 2.2) an die ContextNode-Empfänger übermittelt. Es sei nur kurz erwähnt, dass durch die Verwendung von ContextNodes anstatt Benutzern – als vorläufige Empfänger der Nachricht – auch größere Empfängergruppen effizient unterstützt werden können, wie es auch bei Xcast+ der Fall ist. Der Benutzer muss bei zeitbezogenen Nachrichten eine Ziellokation der jeweiligen Nachricht angeben, aus der anschließend eine Liste von Empfängern in Form von ContextNodes ermittelt wird. Damit bei den ContextRoutern während des Routings der Nachrichten kein Kontextmatching durchgeführt wird, muss eine zeitbezogene Nachricht speziell gekennzeichnet werden. Eine Erkennung zeitbezogener Nachricht kann zum Beispiel anhand des zeitbezogenen Attributfilters erfolgen. Historische Kontexte werden nicht propagiert, weshalb ein Matching historischer Nachrichten zumindest nahe am Speicherort, in diesem Fall bei den betreffenden ContextNodes, durchgeführt werden muss. Dies gilt ebenfalls für futuristische Nachrichten. Bei diesen kann ein Kontextmatching zwar prinzipiell bei beliebigen Knoten durchgeführt werden, da zukünftige Kontexte als reguläre Kontexte in der Zukunft im Netz propagiert werden. Konzepte der Aggregation von Kontexten schränken diese Flexibilität jedoch ein. Zum besseren Verständnis der zentralen Konzepte wird das Kontextmatching im Folgenden der Einfachheit halber direkt bei den ContextNodes durchgeführt.

Kategorisierung der zeitbezogenen Nachrichten Bevor die Verfahren zum Matching und Routing der Nachrichten an die Benutzer als endgültige Empfänger der Nachrichten erörtert werden können, muss eine Einteilung der Nachrichten in die in Abschnitt 4.3 beschriebenen

Kategorien erfolgen. Für historische, gegenwärtige, futuristische und hybride Nachrichten existieren jeweils unterschiedliche Verfahren des Routings. Eine Einteilung verschiedener zeitbezogener Nachrichten in diese Kategorien kann nicht pauschal erfolgen. Gegenwärtige Nachrichten und zeitbezogene Nachrichten, welche die historischen, futuristischen und hybriden Nachrichten beinhalten, können aufgrund der An- oder Abwesenheit eines zeitbezogenen Attributfilters für die Gültigkeitsdauer der Kontexte unterschieden werden. Die zeitbezogenen Nachrichten werden anhand ihres zeitlichen Attributfilters und der aktuellen Systemzeit des ContextNodes weiter eingeteilt. Dies setzt selbstverständlich eine hinreichend synchronisierte Uhr bei den ContextNodes voraus. Ist diese nicht korrekt, so werden eventuell Nachrichten an Empfänger ausgeliefert, die durch das Zeitintervall des Attributfilters einer Nachricht eigentlich nicht vom Sender adressiert wurden. Weiterhin wird möglicherweise ein erfolgreiches Matching einer Nachricht mit einem passenden Kontext verhindert, da diesem falsche Zeitpunkte für das Zeitintervall seiner Gültigkeit vom ContextNode vergeben wurden.

Eine Einteilung der zeitbezogenen Nachrichten bei den ContextNodes wird aufgrund der in Tabelle 4.3 dargestellten Regeln durchgeführt. Dazu müssen die dargestellten Bedingungen in Abhängigkeit des Operators des Attributfilters der jeweiligen Nachricht ausgewertet werden. Das Schlüsselwort NOW bezeichnet den aktuellen Zeitpunkt des ContextNodes, der die Nachricht auswertet, bei der Ankunft der Nachricht. Für die Auswertung der Bedingungen muss der ContextNode seine LOT in die UTC umwandeln und als Wert für NOW verwenden. Alle Zeitangaben in den Nachrichten und Kontexten sind bereits in UTC angegeben und können daher direkt verglichen werden. Events, die beim ContextNode zum Zeitpunkt NOW auftreten (wie beispielsweise die An- der Abmeldung von Kontexten) gelten als gegenwärtig und historisch, nicht jedoch als futuristisch. Solche Events können sofort in eine aktuelle Auswertung der Nachricht einbezogen werden. Der Wert $NOW + 1$ steht für den direkt nach dem gegenwärtigen Zeitpunkt NOW folgende Wert der Zeitrechnung. Die in Tabelle 4.3 verwendeten Variablen a, b und z sind Teile des zeitbezogenen Attributfilters einer Nachricht, deren Bedeutung dem Abschnitt 4.4 entnommen werden kann. Die Kategorie einer Nachricht, die vorerst beim ContextNode ermittelt wird, entscheidet neben der speziellen Behandlung der Nachricht für das Matching über das finale Routing und die Übermittlung der zeitbezogenen Nachricht an den endgültigen Empfänger. Im Weiteren werden diese Verfahren getrennt nach den unterschiedlichen Nachrichtenkategorien diskutiert.

Historische Nachrichten Historische Nachrichten beziehen sich über ihren zeitbezogenen Attributfilter ausschließlich auf Kontexte, die bereits archiviert wurden oder gegenwärtig am System angemeldet sind (siehe Abschnitt 4.3). Rein historische Nachrichten müssen nur auf historische Kontexte ausgewertet werden, da per Definition ein erfolgreiches Matching auf gegenwärtig angemeldete Kontexte nicht möglich ist. Eine weitere typische Eigenschaft von historischen Nachrichten ist, dass sie sofort durch die ContextNodes ausgewertet werden können. Eine Zwischenspeicherung ist nicht erforderlich. D.h. die Nachricht kann nach dem Matching bei den ContextNodes und der anschließenden Übermittlung an die Empfänger verworfen werden. Für historische Kontexte sind beide Zeitpunkte des Zeitintervalls der Gültigkeit bereits bekannt. Für aktuell angemeldete Kontexte ist nur der Anmeldezeitpunkt und damit der Startzeitpunkt der Gültigkeit bekannt. Ohne eine Zwischenspeicherung der Nachricht ist nur dieser Zeitpunkt für die Auswertung verfügbar. Ein Matching des vollständigen Zeitintervalls aktuell angemeldeter Kontexte auf einen zeitbezogenen Attributfilter ist daher ohne Speicherung der zugehörigen Nachricht nicht möglich.

Im ursprünglichen Contextcast werden die Kontexte bei den ContextRoutern gespeichert und einer Verbindung zugeordnet. Nach einem erfolgreichen Matching wird die Nachricht auf dieser Verbindung weitergeleitet. Historische Kontexte hingegen sind Client-IDs zugeordnet.

Operator	Kategorie	Bedingung
before	rein futuristisch	$b \geq NOW$
	hybrid	$b < NOW$
after	rein historisch	$a \leq NOW + 1$
	hybrid	$a > NOW + 1$
starts-only	futuristisch	$b > NOW \wedge a > NOW$
	rein historisch	$b \leq NOW$
	hybrid	$b > NOW \wedge a \leq NOW$
ends-only	historisch	$b \leq NOW$
	rein futuristisch	$b > NOW \wedge a > NOW$
	hybrid	$b > NOW \wedge a \leq NOW$
start-by	historisch	$b \leq NOW$
	rein futuristisch	$b > NOW \wedge a > NOW$
	hybrid	$b > NOW \wedge a \leq NOW$
end-by	rein historisch	$b \leq NOW$
	futuristisch	$b > NOW \wedge a > NOW$
	hybrid	$b > NOW \wedge a \leq NOW$
overlaps	historisch	$b \leq NOW$
	futuristisch	$a > NOW$
	hybrid	$b > NOW \wedge a \leq NOW$
excludes	hybrid	$true$
equals	rein historisch	$b + z \leq NOW$
	rein futuristisch	$a - z > NOW$
	futuristisch	$b - z > NOW$
during	historisch	$b \leq NOW$
	futuristisch	$b > NOW$
contains	rein historisch	$b \leq NOW$
	rein futuristisch	$b > NOW \wedge a \geq NOW + 1$
	hybrid	$b > NOW \wedge a < NOW + 1$

Tabelle 4.3: Kategorisierung der zeitbezogenen Nachrichten

Das Matching historischer Nachrichten auf historische Kontexte erfolgt bei den ContextNodes durch den Aufruf der Funktion „get(attributeFilters)“ aus der Kontextdatenbank-API mit den Attributfiltern der Nachricht. Daraufhin erfolgt beispielsweise eine Umwandlung der Liste von Attributfiltern der Nachricht in ein SQL-Statement und eine Abfrage der Kontextdatenbank. Diese Umwandlung beinhaltet eine Umformung der Bedingungen für die Operatoren eines zeitbezogenen Contextcast, die für die verschiedenen Datentypen zur Verfügung gestellt und nicht durch die Query-Sprache der Datenbank unterstützt werden. Der Rückgabewert einer Anfrage an die Datenbank sind die Client-IDs, für die mindestens ein historischer Kontext zu der Nachricht bzw. der äquivalenten Anfrage passt. Vor der Übermittlung der Nachricht an die Benutzer der Client-IDs werden alle Filter aus der Nachricht entfernt. Filter sind allgemein nur für das Matching bedeutend. Da dieses beim ContextNode bereits durchgeführt wurde, muss folglich nur noch der eigentliche Nachrichteninhalt an die Empfänger zugestellt werden. Zu den Benutzern der Client-IDs, die durch ein Matching der Nachricht mit den gegenwärtig angemeldeten Kontexten zurückgegeben wurden, kann die Nachricht direkt übermittelt werden. Sind diese Benutzer vorübergehend nicht erreichbar, wird die Nachricht nach dem in Abschnitt 4.2 beschriebenen Verfahren weiterbehandelt.

Bevor die Nachricht an die Benutzer übermittelt wird, sollten übereinstimmende Client-IDs aus beiden Mengen des Matchings auf gegenwärtig angemeldete und historische Kontexte entfernt werden. Gleiche Client-IDs werden beim Matching zurückgeliefert, wenn mehrere Kontexte eines Benutzers zur Nachricht passen. Weiterhin können die Client-IDs aus der Menge für gegenwärtig angemeldete Kontexte aus der Menge der Client-IDs für historische Kontexte entfernt werden. Ohne diesen Schritt würde die Nachricht mehrfach zu einem Benutzer über unterschiedliche Wege übermittelt werden, falls dieser sowohl einen zur Nachricht passenden historischen als auch gegenwärtig angemeldeten Kontext besitzt. An Benutzer, die gegenwärtig an dem System mit einem nicht zur Nachricht passenden Kontext angemeldet, aber in der Menge der Client-IDs für historische Kontexte enthalten sind, kann die Nachricht unter Annahme der Erreichbarkeit des Benutzers ebenfalls direkt durch den ContextNode übermittelt werden. Die Übermittlung der Nachricht vom ContextNode zum Benutzer ist einfach und wird wie beim ursprünglichen Contextcast gehandhabt. Bei einer erfolgreichen Übermittlung wird der Client-ID des Benutzers aus der Menge für historische Kontexte gelöscht, so dass die Nachricht nicht zusätzlich an das Mailboxsystem des Benutzers weitergeleitet wird.

Hat der Benutzer in der Zeit zwischen den Gültigkeitsintervallen seines historischen und gegenwärtigen Kontextes einen Wechsel seines Client-ID vollzogen, kann eventuell eine Zuordnung seiner Kontexte durch den ContextNode nicht mehr erfolgen. Ausschließlich die Trusted Nodes können dann eine Zuordnung von Client-IDs zu Benutzern durchführen. Darum wird die Nachricht an die Trusted Nodes der Empfänger übertragen. Die Trusted Nodes der Empfänger können über die Client-IDs ermittelt werden. Die Nachricht erhält nach der Löschung der Attributfilter beim ContextNode einen neuen Attributfilter, der alle Trusted-Node-IDs der zugehörigen Client-IDs enthält, an deren Benutzer die Nachricht übermittelt werden soll. Des Weiteren werden mit diesen Trusted-Node-IDs jeweils eine Liste der lokalen Client-IDs verknüpft, so dass die Nachricht später den Empfängern und ihren Mailboxaccounts durch die Trusted Nodes zugeordnet werden kann. Die Weiterleitung der Nachricht an die Trusted Nodes erfolgt unter Anwendung von Xcast. Dies wird durch die Kontextpropagierung der Trusted Nodes mit den zugehörigen Trusted-Node-IDs möglich. Nachdem die Nachrichten bei den Trusted Nodes angekommen sind, muss jeder Trusted Node eine Abbildung der lokalen Client-IDs für die Empfänger auf die zugehörigen Mailboxaccounts durchführen. Da mehrere historische Kontexte eines Benutzers mit unterschiedlichen Client-IDs zu einem erfolgreichen Matching mit der Nachricht geführt haben könnten, muss an dieser Stelle eine erneute Duplikatfilterung durchgeführt werden. In jedem der resultieren-

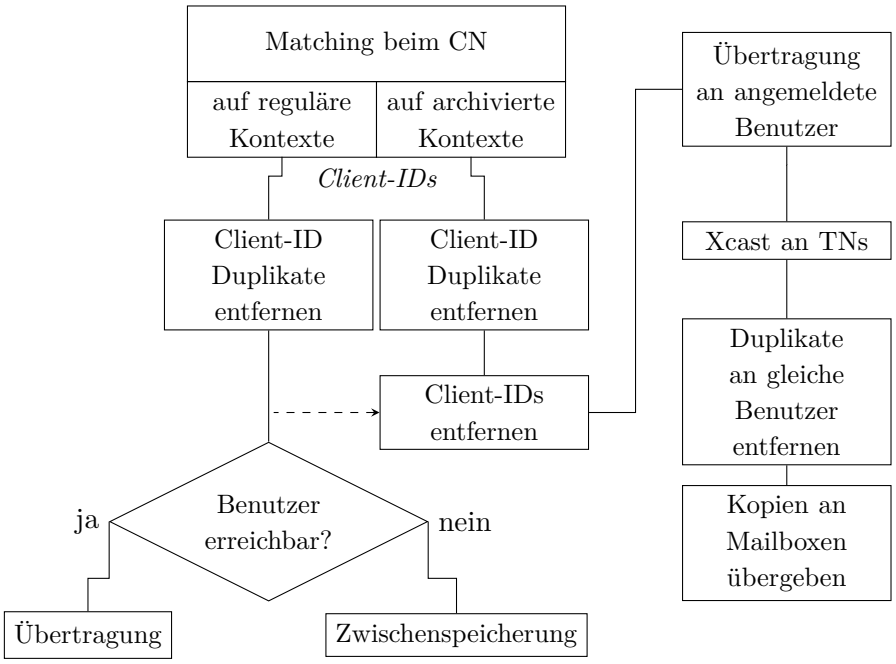


Abbildung 4.6: Routing und Matching historischer Nachrichten

den Mailboxaccounts wird daraufhin eine Kopie der Nachricht für jeden Empfänger abgelegt. Abbildung 4.6 fasst die bisher beschriebene Vorgehensweise für das Routing und Matching historischer Nachrichten in einem Flussdiagramm zusammen.

Die Übermittlung der in der Mailbox gespeicherten Nachrichten an die Benutzer kann auf mehrere Arten durchgeführt werden. Diese sind teilweise davon abhängig, wie der Benutzer sich an seinem Mailboxknoten an- und abmeldet. Hier werden nur beispielhaft drei Möglichkeiten genannt:

1. Jeder Benutzer meldet sich periodisch bei seinem Mailboxknoten, um die gespeicherten Nachrichten abzuholen („Polling“).

2. Jeder Benutzer meldet sich mit einer Adresse des Underlays, unter der er erreichbar ist, an und ab. Sobald die Mailbox des Benutzers Nachrichten enthält und der Benutzer angemeldet ist, kann der Mailboxknoten die gespeicherten Nachrichten an den Benutzer über das Underlay übertragen („Pushing over Underlay“).

3. Die Client-IDs werden als Attribute in den Kontexten im Netz propagiert. Der Mailboxknoten erkennt die Anwesenheit des Benutzers im System, sobald ein Kontext mit dem Client-ID des Benutzers bei ihm eingeht. Über den ContextNode-ID des Kontextes erfährt er gleichzeitig den aktuellen Aufenthaltsort des Benutzers. Die Nachrichten können über eine Xcast-Nachricht mit dem ContextNode-ID und aktuellen Client-ID an den Benutzer übermittelt werden, sobald dieser am System angemeldet ist („Pushing over Overlay“).

Sind im dritten Fall Empfänger bei der Ankunft einer Nachricht beim Mailboxsystem gegenwärtig am System angemeldet, so kann die Speicherung der Nachricht im Mailboxsystem entfallen und eine direkte Übermittlung an die angemeldeten Empfänger erfolgen. Wenn sich mehrere Empfänger einer Nachricht die gleiche Mailbox teilen und zum gleichen Zeitpunkt am System angemeldet sind, kann der Vorteil einer Multicast-ähnlichen Übertragung ausgenutzt werden, indem die Nachricht über ein Xcast an die Empfänger zugestellt wird. Für die

Übermittlung der Nachrichten und die Erkennung, ob ein Empfänger der Nachricht gegenwärtig am System angemeldet ist, sind die aktuell gültigen Client-IDs der Empfänger nötig. Diese können über den Trusted Node durch Übergabe der alten Client-IDs der Nachricht abgefragt werden.

Futuristische Nachrichten Bei futuristischen Nachrichten muss ein Matching auf gegenwärtig angemeldete und zukünftige Kontexte beim ContextNode durchgeführt werden. Im Gegensatz zum Matching historischer Nachrichten, ist bei einem Matching futuristischer Nachrichten auf gegenwärtig angemeldete Kontexte – zum Zeitpunkt der Nachrichtenankunft – der Endzeitpunkt des Gültigkeitsintervall relevant.

Zukünftige Kontext zeichnen sich dadurch aus, dass beide Zeitpunkte des Gültigkeitsintervalls in der Zukunft liegen. Da ein Matching erst durchgeführt werden kann, wenn die notwendigen Zeitpunkte bekannt sind, müssen futuristisch Nachrichten generell zwischengespeichert werden. Ein Matching auf einen Kontext kann erst dann erfolgen, wenn die für die Auswertung des Operators des zeitbezogenen Attributfilters der Nachricht erforderlichen Zeitpunkte des Zeitintervalls des entsprechenden Kontextes bekannt sind. Die notwendigen Zeitpunkte für die Auswertung einer Operation können aus Tabelle 4.2 abgeleitet werden. Bei einer konkreten Implementierung des Contextcast muss darauf geachtet werden, dass eine strenge Auswertung der Regeln aus Tabelle 4.2 nicht angebracht ist. Beispielsweise kann die Auswertung der Bedingung $a \leq y$ für den Operator *overlaps* entfallen, falls der Zeitpunkt a bereits in der Vergangenheit liegt und die Nachricht auf gegenwärtig angemeldete und zukünftige Kontexte ausgewertet werden soll. In diesem Fall ist der Endzeitpunkt des Gültigkeitsintervalls eines Kontextes für das Matching irrelevant, obwohl dieser in der Auswertungsregel der Tabelle 4.2 für den Operator *overlaps* aufgeführt ist. Das gleiche gilt für den *before* Operator. Nachdem der Zeitpunkt b abgelaufen ist, also in der Vergangenheit liegt, muss keine Überprüfung des zeitlichen Attributfilters mehr durchgeführt werden. Der Filter selbst kann theoretisch aus der gespeicherten Nachricht entfernt werden. Es sei hier nochmals darauf hingewiesen, dass eine Einteilung der zeitbezogenen Nachrichten in die verschiedenen Kategorien von dem Zeitpunkt NOW des ContextNodes abhängig ist, an dem die Nachricht beim ContextNode eingetroffen ist. Es ist also durchaus möglich, dass eine futuristische Nachricht im Laufe der Zeit ihren Zukunftsbezug verliert. Ein ähnlicher Effekt kann auftreten, wenn zwei bis auf den Nachrichteninhalt übereinstimmende Nachrichten von zwei unterschiedlichen Sendern verschickt werden und diese zu unterschiedlichen Zeitpunkten bei dem gleichen ContextNode eintreffen. Beim ContextNode wird die zuerst ankommende Nachricht möglicherweise als futuristische Nachricht behandelt, die letztere als historische Nachricht.

Sobald der Endzeitpunkt der Gültigkeit eines gegenwärtig angemeldeten Kontextes durch die Abmeldung bekannt wird, muss der Kontext vor der Archivierung in der Kontextdatenbank auf die gespeicherten futuristischen Nachrichten ausgewertet werden. Ein Matching neu angemeldeter Kontexte mit futuristischen Nachrichten kann nur durchgeführt werden, falls der Endzeitpunkt der Gültigkeit des Kontextes nicht für die Auswertung der Operation des zeitlichen Attributfilters in einer Nachricht benötigt wird. Andernfalls muss ein Matching mit diesem Kontext bis zu seiner Abmeldung durch die Zwischenspeicherung der Nachricht verzögert werden. Falls ein Matching mit einem angemeldeten Kontext erfolgreich ist, wird die Nachricht direkt durch den ContextNode an den Benutzer übermittelt. Nach einem erfolgreichen Matching mit einem abgemeldeten Kontext wird die Nachricht, falls der Benutzer nicht mehr erreichbar und das Persistenzattribut der Nachricht gesetzt ist, an den Mailboxknoten des Benutzers weitergeleitet. Im Gegensatz zu der Übermittlung an Benutzer, die gegenwärtig mit einem Kontext am System angemeldet sind, muss in diesem Fall die TTL der Nachricht nicht abgewartet werden. Die Übermittlung der Nachricht erfolgt

Operator	Bedingung
before	$false$
after	$NOW \geq a$
starts-only	$NOW > b \vee (NOW > a \wedge \forall m \in T(NOW) : x > a)$
ends-only	$NOW > b$
start-by	$NOW > b$
end-by	$NOW > b$
overlaps	$NOW > b$
excludes	$false$
equals	$NOW > b + z \vee (NOW > a + z \wedge \forall m \in T(NOW) : \lvert x - a \rvert > z)$
during	$NOW > a \wedge \forall m \in T(NOW) : x > a$
contains	$NOW > b$

Tabelle 4.4: Lebensdauer futuristischer und hybrider Nachrichten

entsprechend der Übermittlung historischer Nachrichten an Mailboxen.
Rein futuristische Nachrichten müssen im Gegensatz zu allgemein futuristischen Nachrichten nur auf Kontexten ausgewertet werden, deren Anmeldung nach der Ankunft der Nachricht beim ContextNode stattgefunden hat. Die bei der Ankunft der Nachricht angemeldeten Kontexte sind daher nicht relevant.

Bei einer häufigen Verwendung futuristischer Nachrichten entsteht eine hohe Speicherbelastung der ContextNodes durch die Zwischenspeicherung der Nachrichten. Für viele Operatoren kann ein Zeitpunkt angegeben werden, ab dem ein Kontextmatching mit der betreffenden zeitbezogenen Nachricht nicht mehr zum Erfolg führen kann. Bis zu diesem Zeitpunkt wurde bereits ein Matching mit all den Kontexten durchgeführt, die potentiell zu der Nachricht passen könnten. Tabelle 4.4 gibt den Zeitpunkt für alle in Abschnitt 4.4 definierten Operatoren an. Dazu werden außer den Operatoren auch die Zeitangaben der zeitbezogenen Attributfilter der Nachricht und die aktuelle Zeit des ContextNodes benötigt. Die Bedingungen der Tabelle 4.4 werden durch den ContextNode zu dem booleschen Wert $true$ ausgewertet, sobald der Zeitpunkt erreicht ist. Ab diesem Zeitpunkt können die futuristischen Nachrichten getrost verworfen und der Speicherplatz für neue futuristische Nachrichten frei gemacht werden. Die Menge $T(Z)$ ist definiert als die Menge der Kontexte, die zum Zeitpunkt Z beim ContextNode ein offenes Zeitintervall ihrer Gültigkeitsdauer besitzen. Dies ist gleichbedeutend mit der Menge der zum Zeitpunkt Z an dem ContextNode angemeldeten Kontexten.

Hybride Nachrichten Hybride Nachrichten stellen eine Mischung aus historischen und futuristischen Nachrichten dar. Die Verfahren für das Matching und Routing setzen sich demnach aus denen für historische und futuristische Nachrichten zusammen. Hybride Nachrichten müssen potentiell auf historische, gegenwärtige und zukünftige Kontexte ausgewertet werden. Für bestimmte Kombinationen aus Operator, aktuellem Zeitpunkt des ContextNodes und Zeitintervall des Attributfilters kann die Auswertung für eine dieser Mengen entfallen. Beispielsweise muss eine Nachricht mit *excludes*-Operator zum Zeitpunkt $NOW > a \wedge NOW < b$

nicht auf die gegenwärtig angemeldeten Kontexten ausgewertet werden. Durch den Ausschluss von an dem Matching beteiligten Kontexten, die nicht zu einer erfolgreichen Übereinstimmung führen, kann der Rechenaufwand beim ContextNode reduziert werden. Im Folgenden wird auf ein vollständiges Matching mit historischen, gegenwärtigen und zukünftigen Kontexten eingegangen. Falls in einzelnen Fällen eine dieser Mengen nicht betrachtet werden muss, kann der zugehörige Schritt einfach entfallen.

Sobald eine hybride Nachricht beim ContextNode eingeht, kann der historische „Teil“ der Nachricht sofort abgearbeitet werden. Nach einem Matching mit den archivierten, noch zu archivierenden und gegenwärtig angemeldeten Kontexten und dem anschließenden Versand der Nachricht an die Empfänger, bleibt noch der futuristische Teil der Nachricht für die Auswertung übrig. Ein Matching mit den gegenwärtig angemeldeten Kontexten kann natürlich nur erfolgen, wenn zu dem Zeitpunkt des Matchings für den Operator des Attributfilters und dem gegebenen Zeitintervall die Anmeldezeitpunkte der Kontexte für ein Matching ausreichen. Die Nachricht selbst bleibt während dem gesamten Matching unverändert. Im Unterschied zu der Behandlung historischer Nachrichten darf die Nachricht ab diesem Zeitpunkt noch nicht verworfen werden. Im Gegenteil: Die Nachricht wird wie bei der Behandlung der futuristischen Nachrichten gespeichert, bis die für die Auswertung der Operation erforderlichen Zeitpunkte der Gültigkeit eines Kontextes bekannt sind. Generell ist eine An- oder Abmeldung von Kontexten der Auslöser für den Anstoß eines Matchings bei der Behandlung futuristischer Nachrichten. „Für den Anstoß“ bedeutet, dass wenn zum Beispiel bei einer Anmeldung eines Kontextes für die Auswertung noch der Abmeldezeitpunkt fehlt, ein Matching nicht durchgeführt, sondern nur angestoßen wird. Diese Tatsache vereinfacht das Matching ungemein, da ansonsten periodisch eine Überprüfung auf ein durchführbares Matching mit jedem Kontext angestoßen werden müsste. Im ersten Moment klingt dies trivial, jedoch existieren individuell zu den Operatoren noch weitere Zeitpunkte, bei deren Überschreitung sich die Behandlung des Matchings ändern kann. Ein Beispiel hierzu ist der operationsabhängige Zeitpunkt, ab dem eine futuristische Nachricht gelöscht werden kann. Die Löschzeitpunkte gelten auch für hybride Nachrichten und ändern das Matching dahingehend, dass keines mehr mit der Nachricht durchgeführt werden muss.
Ein anderes interessantes Beispiel ergibt sich, wenn die aktuelle Zeit NOW bei der Auswertung des Operators *ends-only* beim ContextNode den Zeitpunkt b überschreitet. Bis zu diesem Zeitpunkt mussten für die Auswertung einer futuristischen Nachricht die Endzeitpunkte der Gültigkeitsintervalle der Kontexte abgewartet werden. Danach können alle gegenwärtig angemeldeten Kontexte sofort ausgewertet werden, da von der Bedingung des Operators ohne Einschränkung nur noch verlangt wird, dass der Endzeitpunkt der Kontexte in der Zukunft liegt. Das „Warten“ auf die Abmeldungen der Kontexte würde in diesem Fall die Weiterleitung der Nachricht und den Löschzeitpunkt nur unnötig verzögern.

Ein grundlegendes Problem bei der Behandlung futuristischer und hybrider Nachrichten ist die Mehrfachauswertung auf ein und denselben Kontext. Durch diese Mehrfachauswertungen kann es passieren, dass zwei Kopien der gleichen Nachricht durch ein mehrfach durchgeführtes erfolgreiches Matching auf einen Kontext zu einem Empfänger übertragen werden. Die geschieht beispielsweise bei einer futuristischen Nachricht, wenn bei dem Matching des *before*-Operators ein Matching für jede An- und Abmeldung eines Kontextes durchgeführt wird. Da für diesen Operator nur der Anmeldezeitpunkt für das Matching relevant ist, wird ein zur Nachricht passender Kontext bei der An- und Abmeldung mit der Nachricht übereinstimmen, was jeweils eine Übermittlung der Nachricht an den Empfänger initiiert. Dieser Aspekt muss bei der Verhinderung der Entstehung von Nachrichtenduplikaten beachtet werden. In manchen Fällen kann allerdings eine Mehrfachauswertung ohne erhöhten Aufwand nicht verhindert werden. Eine Speicherung der ausgewerteten Kontexte bei jeder gespeicherten

futuristischen und hybriden Nachricht stellt solch ein Verfahren dar, das zwar eine Mehrfachauswertung verhindern kann, aber gleichzeitig einen hohen Speicheraufwand benötigt. Im Gegensatz dazu ist es möglich, eine erneute Auswertung auf einen Kontext im Rahmen der Auswertung zu erkennen und dadurch die Entstehung von Nachrichtenduplikaten zu verhindern. Mit dieser Möglichkeit befasst sich der folgende Abschnitt.

Duplikatvermeidung Eine Vermeidung von Nachrichtenduplikaten stellt sich allgemein für zeitbezogene Nachrichten als komplexes Vorhaben heraus. Eine Betrachtung der Möglichkeiten zur Duplikatvermeidung ist Gegenstand des gesonderten Abschnitts 5.1. An dieser Stelle wird ausschließlich die Vermeidung von Nachrichtenduplikaten besprochen, die bei der Behandlung futuristischer und hybrider Nachrichten beim ContextNode durch wiederholtes Matching auf ein und denselben Kontext entstehen. Für historische Kontexte erzeugt der ContextNode keine Nachrichtenduplikate, da ein Matching auf die Kontexte ohne Verzögerung durchgeführt werden kann und redundante Client-IDs aus der Liste der Empfänger entfernt werden. Für futuristische und hybride Nachrichten entstehen ebenfalls keine Nachrichtenduplikate durch Mehrfachauswertungen von Kontexten, falls für die Auswertung der futuristischen Nachricht oder des futuristischen Nachrichtenteils der hybriden Nachricht immer auf den Endzeitpunkt eines Kontextes gewartet wird. Ein Matching erfolgt dann bei der Abmeldung eines Kontextes, also bevor dieser in der Kontextdatenbank archiviert wird. Da die Archivierung eines Kontextes eine einmalige Angelegenheit darstellt, kann es nicht zu Mehrfachauswertungen kommen. Diese Vorgehensweise beinhaltet andererseits drei wesentliche Nachteile:

1. Die Weiterleitung und damit der endgültige Empfang der Nachrichten wird eventuell unnötig lange verzögert.

2. Die Nachrichten müssen länger gespeichert werden, wodurch eine höhere Speicherbelastung der ContextNodes entsteht.

3. Nach Abmeldung des Kontextes und anschließendem Matching ist der Benutzer möglicherweise bereits nicht mehr mit dem System verbunden. In diesem Fall müsste die Nachricht zum Trusted Node des Empfängers übermittelt werden, was mit einem erhöhten Nachrichtenaufwand verbunden ist.

Diese Nachteile sollen nicht einfach in Kauf genommen werden. Da andererseits ein unnötig hoher Nachrichtenaufwand durch Nachrichtenduplikate nicht erwünscht ist, müssen bereits beim ContextNode Vorkehrungen zur Vermeidung dieser Duplikate getroffen werden. Hierfür muss zunächst geklärt werden, in welchen Ausnahmefällen des Matchings beim ContextNode Nachrichtenduplikate entstehen können.

Zu einem festen Zeitpunkt sind für das Matching einer Nachricht beim ContextNode – abhängig von der Operation des zeitbezogenen Filters – entweder nur die Anmeldezeitpunkte, die Abmeldezeitpunkte, das gesamte Zeitintervall oder gar kein Zeitpunkt von Bedeutung. Im ersten Fall werden durch die Behandlung des historischen Nachrichtenteils die archivierten und gegenwärtig angemeldeten Kontexte ausgewertet. Ein Matching auf zukünftige Kontexte kann sofort bei deren Anmeldung durchgeführt werden. Eine Überschneidung des Matchings auf Kontexte des historischen und futuristischen Nachrichtenteils besteht nicht. Der zweite und dritte Fall wird gleich behandelt, da bei der Abmeldung eines Kontextes auch gleichzeitig der Anmeldezeitpunkt bekannt ist. Für den historischen Nachrichtenteil werden die Kontexte der Kontextdatenbank heran gezogen. Der futuristische Teil wird durch ein Matching auf die gegenwärtig angemeldeten und zukünftigen Kontexte abgehandelt. Das Matching auf einzelne Kontexte erfolgt dann immer zu deren Abmeldezeitpunkt, da dieser

für die Auswertung relevant ist. Der letzte Fall tritt ein, wenn für einen neuen, bzw. gerade abgemeldeten Kontext ein erfolgreiches Matching ausgeschlossen wird oder von einem erfolgreichen Matching ohne die Auswertung des zeitlichen Attributfilters ausgegangen werden kann. Dies ist beispielsweise bei der Auswertung des *before*-Operators nach Ablauf des Zeitpunkts b der Fall. Kann, wie bei dem Beispiel mit dem *before*-Operator, generell von einem erfolgreichen Matching ausgegangen werden, wird die Nachricht sofort bei der Anmeldung des Kontextes an den Empfänger weitergeleitet. In allen Fällen kann es nicht zu einer Mehrfachauswertung einzelner Kontexte kommen.

Es existiert jedoch noch ein Spezialfall für das Matching, bei dem der gesamte Zeitintervall eines Kontextes von Bedeutung ist. Kann zu einem gewissen Zeitpunkt der Endzeitpunkt *oder* der Startzeitpunkt zu einem erfolgreichen Matching führen, kommt es möglicherweise in zwei Fällen zu Mehrfachauswertungen:

1. Beim Matching des historischen Nachrichtenteils für eine hybride Nachricht wurde die Übereinstimmung eines gegenwärtig angemeldeten Kontextes aufgrund seines Anmeldezeitpunkts erfolgreich geprüft. Anschließend wird der Kontext bei der Behandlung des futuristischen Nachrichtenteils aufgrund seines Abmeldezeitpunkts wieder erfolgreich ausgewertet.

2. Das Matching auf einen zukünftigen Kontext wird zum Zeitpunkt der An- und Abmeldung durchgeführt.

Der *overlaps*-Operator ist ein Beispiel, bei dessen Auswertung es zu den oben genannten Fällen kommen kann. Diese können nur während dem Matching von hybriden und futuristischen Nachrichten auftreten, da historische Nachrichten nicht über einen längeren Zeitraum hinweg gespeichert werden. Ebenfalls problematisch sind die möglichen operationsbedingten zeitlichen Wechsel der Signifikanz der Zeitpunkte des Gültigkeitsintervalls von Kontexten für das Matching. Es existieren genau zwei Übergänge dieser Art:

1. Ein Wechsel der Relevanz bei der Auswertung des zeitlichen Attributfilters vom Abmeldezeitpunkt zum Anmeldezeitpunkt.

2. Ein Wechsel der Relevanz vom Anmeldezeitpunkt zum Abmeldezeitpunkt.

Bei der ersten Möglichkeit kommt es nicht zu Mehrfachauswertungen. Der historische Nachrichtenteil wird komplett über die Kontextdatenbank abgewickelt. Das Matching der übrigen Kontexte erfolgt erst bei deren Abmeldung. Nach dem Wechsel erfolgt das Matching bei der Anmeldung der Kontexte. Alle bisher betrachteten Kontexte wurden bereits archiviert und werden daher nicht nochmals ausgewertet.
Bei der zweiten Möglichkeit werden für den historischen Nachrichtenteil zusätzlich die aktuell angemeldeten Kontexte ausgewertet, die ebenfalls bei der Abmeldung in die Auswertung des futuristischen Nachrichtenteils fallen. Hier kann also für hybride Nachrichten eine Mehrfachauswertung entstehen. Für hybride und futuristische Nachrichten erfolgt diese Mehrfachauswertung, wenn zukünftige Kontexte bei der Anmeldung und nach dem Wechsel nochmals bei der Abmeldung ausgewertet werden.

Für die Lösung dieses Problems und der damit verbundenen Vermeidung von unnötigen Nachrichtenduplikaten kann für jede Nachricht eine Liste der bereits ausgewerteten Kontexte gespeichert werden. Dieser Vorgang erzeugt jedoch eine hohe Speicherbelastung, da hybride und futuristische Nachrichten häufig für eine längere Zeit gespeichert werden müssen und daher eine Auswertung auf viele Kontexte erfolgt. Ein alternativer Lösungsansatz führt für das Matching mit einem Kontext vor der eigentlichen Auswertung einen weiteren Prüfschritt ein. Dieser Prüfschritt führt eine Art *Short Circuit Evaluation*[8] (SCE) auf den

[8] Kurzschlussauswertung

booleschen Auswertungsausdruck des Operators aus. Vor dem Matching bei der Abmeldung eines Kontextes wird geprüft, ob der Auswertungsausdruck bereits durch die Kenntnis des Anmeldezeitpunkts zu dem booleschen Wert *true* ausgewertet werden kann. Falls das der Fall ist, wurde der Kontext bereits bei seiner Anmeldung ausgewertet, wobei eine erfolgreiche Übereinstimmung festgestellt wurde und der Kontext daher nicht nochmals ausgewertet werden muss. Andernfalls kann der Kontext ausgewertet werden, ohne dass es zu einer Mehrfachauswertung und Nachrichtenduplikaten kommt. Dieses Verfahren stellt auch eine Lösung für das Problem der gleichzeitigen Auswertung bei An- und Abmeldung eines Kontextes dar, wie beispielsweise bei dem *overlap*-Operator im Zeitintervall von a bis b. Angenommen ein Kontext wird in diesem Zeitintervall abgemeldet. Wenn sich nun herausstellt, dass der Anmeldezeitpunkt ebenfalls in diesem Zeitintervall liegt, kann die Auswertung aufgrund der folgenden Prüfung entfallen:

$$a \leq x \leq b \wedge x \leq y \Rightarrow x \leq b \wedge a \leq y.$$

Operatoren mit einer oder-Verknüpfung der Bedingungen für den An- und Abmeldezeitpunkt im Auswertungsausdruck sind unter der Bedingung, dass $x \leq y$ und $a \leq b$ gilt, besonders anfällig für Mehrfachauswertungen. Rein theoretisch können Nachrichten mit solchen Operatoren in den zeitbezogenen Attributfiltern beim Verfasser in zwei Nachrichten aufgetrennt werden. Diese beiden Nachrichten stellen bis auf den zeitbezogenen Attributfilter Kopien dar. Der zeitbezogene Attributfilter der einen Nachricht enthält den Operator, der nur die Bedingung für den Anmeldezeitpunkt prüft. Die andere Nachricht enthält den Operator für die Prüfung des Abmeldezeitpunkts. Die Auswertungsausdrücke dieser Operatoren ergeben disjunktiv verknüpft gerade den Auswertungsausdruck des ursprünglichen Operators. Die Übermittlung zweier Nachrichten mit dem gleichen Inhalt erzeugt jedoch einen weitaus höheren Nachrichten- und Rechenaufwand als nur eine Nachricht. Aus diesem Grund sollte darauf geachtet werden, dass die verfügbare Liste an Operatoren bezüglich einer disjunktiven Verknüpfung der Auswertungsausdrücke der einzelnen Operatoren möglichst abgeschlossen ist. Operatoren, die aus der disjunktiven Verknüpfung entstehen und aus Anwendungssicht keinen Sinn machen, sollten jedoch nicht aufgenommen werden. Eine konjunktive Verknüpfung von zeitlichen Attributfiltern, also speziell auch der Operatoren, kann generell durch mehrere Attributfilter in einer Nachricht realisiert werden, da diese beim Matching konjunktiv verknüpft werden. Ein Problem entsteht in diesem Zusammenhang bei der Kategorisierung der zeitbezogenen Nachrichten. Mit Zunahme der Anzahl an Attributfiltern wird eine Einteilung immer komplexer. Aus diesen Gründen erlaubt das in dieser Arbeit beschriebene zeitbezogene Contextcast lediglich einen zeitbezogenen Attributfilter pro Nachricht.

In Tabelle 4.5 ist die Behandlung des Matchings im zeitlichen Verlauf für jeden Operator aufgelistet. Für die einzelnen Zeitintervalle ist angegeben, welche Kontexte für ein Matching betrachtet werden müssen. Historische Nachrichten und der historische Nachrichtenteil hybrider Nachrichten verwenden immer die Kontextdatenbank und die gegenwärtig angemeldeten Kontexte für ein Matching. Das Matching selbst kann sofort durchgeführt und abgeschlossen werden, weshalb keine Verzögerungen entstehen und daher eine zeitliche Betrachtung hier keinen Sinn macht. Das Matching gegenwärtig angemeldeter Kontexte für den historischen Nachrichtenteil wird durchgeführt, falls zum aktuellen Zeitpunkt des ContextNodes abhängig vom Operator nur der Anmeldezeitpunkt relevant ist (siehe Tabelle 4.5). Demnach ist der dargestellte Verlauf des Matchings nur für futuristische Nachrichten und den futuristischen Nachrichtenteil hybrider Nachrichten interessant. Mit *NOW* ist wieder die aktuelle Zeit eines ContextNodes in der UTC-Repräsentation bezeichnet. Liegt diese außerhalb des dargestellten Zeitintervalls, so wird die Nachricht durch den ContextNode als historische Nachricht interpretiert und kann nach der Auswertung und Weiterleitung verworfen werden.

Zeitbezogene Nachricht	
Anwendungs-ID	Wie beim regulären Contextcast
Nachrichten-ID	Wie beim regulären Contextcast
Lokation	Ziellokation der Nachricht in geographischer Darstellung
Empfänger*	ContextNode-IDs der durch den ersten ContextNode ermittelten ContextNode-Empfänger
Attributfilter*	Attributfilterliste inklusive zeitbezogenem Attributfilter
Zeitbez. Attributfilter	
Persistenzattribut	Boolesches Attribut für die Persistenz der Nachricht
Zeitbezogen?	Boolesches Attribut zur Unterscheidung von regulären und zeitbezogenen Nachrichten
Futuristisch/Historisch	Kategorie der zeitbezogenen Nachricht
Splitpoint-Knoten?	Boolesches Attribut
TREELEVEL?	TTL-Zähler für das tiefenbeschränkte Backtracking
Vert. Auswertung?	Boolesches Attribut
TTL	TTL in UTC-Repräsentation für die Zwischenspeicherung der Nachricht bei den ContextNodes der Empfänger
Nachrichteninhalt	Wie beim regulären Contextcast

Abbildung 4.7: Struktur einer zeitbezogenen Nachricht

4.6 Zusammenfassung

In diesem Abschnitt werden nochmals die Strukturen von Teilen des Systems zusammengefasst. Im Einzelnen sind dies die Kontexte, zeitbezogenen Nachrichten und die API des Systems für die Benutzer bei den ContextNodes. Die genaue formale Darstellung der einzelnen Elemente, die Trennzeichen und reservierten Bezeichner bleiben der Implementierung überlassen. Die gezeigte Darstellung der Strukturen verfolgt lediglich den Zweck, eine Übersicht über die notwendigen Elemente der jeweiligen Einheit zu geben. Eine detailliertere Festlegung schränkt nur unnötig den Gestaltungsspielraum der Implementierung ein.

Abbildung 4.7 zeigt die grobe Struktur einer zeitbezogenen Nachricht auf. Der Anwendungs-ID und der Nachrichten-ID sind allgemeine Bezeichner und werden wie bei Nachrichten des regulären Contextcast gehalten. Die Darstellung der geographischen Ziellokation der Nachricht hängt von der Implementierung des Systems ab. Je nach gewünschter Genauigkeit und Komplexität der Auswertung beim Matching reicht die Darstellung von einem einfachen Punkt oder Rechteck bis hin zum allgemeinen Polygon. Die Empfängerliste der Nachricht beinhaltet die ContextNode-IDs der Nachrichtenempfänger, die durch den ContextNode des Senders ermittelt und in die Nachricht eingetragen werden. Im weiteren Verlauf des Routings der Nachricht kann diese Liste auch die Trusted-Node-IDs beinhalten, an die die Nachricht für eine anschließende Archivierung in Mailboxen zugesandt wird. Zeitbezogene Nachrichten und reguläre Nachrichten besitzen die gleiche Struktur der allgemeinen Attributfilter. Der zeitbezogene Attributfilter ist für zeitbezogene Nachrichten Pflicht und Teil der gesamten Attributfilterliste. Das Persistenzattribut entscheidet über die mögliche Archivierung der Nachrichten. Gerade für historische und hybride Nachrichten sollte dieses Attribut auf den Wert „true“ gesetzt werden, da sonst keine Auslieferung der Nachrichten an Benutzer histo-

Operator	Zeitintervalle			
before:	$NOW \leq b$ W	$NOW > b$ X		
after:	$NOW < a$ Y			
starts-only:	$NOW < a$ W	$NOW = a$ Z_2	$NOW \geq a$ - $NOW \leq b$ Y	
ends-only:	$NOW \geq b$ Z_1			
start-by:	$NOW < a$ W	$NOW \geq a$ - $NOW \leq b$ X		
end-by:	$NOW < a$ W	$NOW \geq a$ - $NOW \leq b$ Y		
overlaps:	$NOW \geq a$ - $NOW \leq b$ X & Y			
excludes:	$NOW < a$ Y	$NOW \geq a$ - $NOW \leq b$ W	$NOW > b$ X	
equals:	$NOW \leq a + z$ W	$NOW = a + z$ Z_2	$NOW > a + z$ - $NOW < b - z$ W	
	$NOW \geq b - z$ - $NOW \leq b + z$ Y			
during:	$NOW < a$ W	$NOW = a$ Z_2	$NOW > a$ - $NOW < b$ W	$NOW \geq b$ Y
contains:	$NOW < a$ W	$NOW \geq a$ - $NOW \leq b$ Y		

Tabelle 4.5: Matching im zeitlichen Verlauf

W: Keine Kontexte
X: Im Zeitintervall angemeldete Kontexte
Y: Im Zeitintervall abgemeldete Kontexte
Z_1: Matching aller gegenwärtig angemeldeten Kontexte
Z_2: Überprüfung der Löschbedingung der Nachricht in Abhängigkeit der gegenwärtig angemeldeten Kontexte

Kontext	
Context-ID	Eindeutiger Bezeichner des Kontextes
Client-ID	Trusted-Node-ID und lokaler Client-ID des Benutzers
Lokation	Die Lokation des Benutzers oder der Zugangsbereich seines ContextNodes
ContextNode-ID	Der ID des ContextNodes, bei dem der Kontext angemeldet ist
Attribut*	Attributliste des Kontextes
Startzeitpunkt	Anmeldezeitpunkt des Kontextes beim ContextNode in UTC
Endzeitpunkt	Abmeldezeitpunkt des Kontexte beim ContextNode in UTC
Aggregiert/Originär	Boolesches Attribut

Abbildung 4.8: Struktur eines Kontextes

rischer Kontexte erfolgt. Je nach Implementierung des zeitbezogenen Contextcast kann ein weiteres Attribut oder der zeitbezogene Attributfilter für die Identifikation einer zeitbezogenen Nachricht verwendet werden. Die booleschen Attribute für die Einstufung der Nachricht als futuristisch oder historisch, für den Fund des Splitpoint-Knotens und der verteilten Auswertung futuristischer Nachrichten werden, wie auch das Attribut für den *TREELEVEL*, für die Verfahren im folgenden Kapitel benötigt. Die TTL der Nachricht für eine eventuelle Zwischenspeicherung kann vom Benutzer vorgegeben werden, wobei eine Festlegung der Obergrenze für diese TTL durch die ContextNodes ebenfalls denkbar ist. Die Formatierung des Nachrichteninhalts der zeitbezogenen Nachricht ist anwendungsabhängig und unterliegt keinen konkreten Vorgaben.

Boolesches Attribut:	'true'|'false'
Attributfilter:	Name':'Operator':'Wert
Zeitbezogener Attributfilter:	'ZBAF:'ZOperator':'Zeitstempel','Zeitstempel','Zeitstempel
Futuristisch/Historisch:	'Z'|'true'|'false'

Der Name des zeitbezogenen Attributfilters wurde beispielhaft auf den Wert ZBAF festgelegt. ZOperator ist die Menge der definierten Operatoren für den zeitbezogenen Attributfilter. Die Menge der verfügbaren Operatoren für reguläre Attributfilter hängt vom Typ des Attributfilters ab. Der Wert Z für die Kategorie der Nachricht bedeutet, dass die Kategorie noch nicht festgestellt wurde.

Die Struktur eines Kontextes im zeitbezogenen Contextcast ist in Abbildung 4.8 dargestellt. Für die eindeutige Identifikation, welche beispielsweise für die Abmeldung eines Kontextes eine wichtige Rolle spielt, erhält jeder Kontext eine Context-ID. Der systemweit eindeutige Client-ID setzt sich aus dem Trusted-Node-ID und dem lokalen Client-ID des Benutzers beim Trusted Node zusammen. Die Lokation des Benutzers im Kontext wird entweder vom Benutzer selbst festgelegt, falls dieser über die nötigen Geräte zur Ermittlung seiner Position verfügt, oder vom ContextNode des Benutzers auf dessen Zugangsbereich. Die Attributliste entspricht der von regulären Kontexten des ursprünglichen Contextcast. Start- und Endzeitpunkt des Kontextes sind nicht Teil der Attributliste. Auf diese kann aus Benutzersicht nur mittels des zeitbezogenen Attributfilters einer zeitbezogenen Nachricht zugegriffen werden. Der Endzeitpunkt des Kontextes ist erst nach der Archivierung verfügbar. Der Zustand eines Kontextes, also ob er ein Aggregat mehrerer Kontexte darstellt, wird für die Verfahren des folgenden Kapitels benötigt. Die TTL des Kontextes und ein Attribut, das den Benutzer des Kontextes als mobilen Client identifiziert, werden von den ContextNodes verwaltet und dem Kontext zugeordnet. Sie sind nicht Teil des Kontextes und werden

getListOfTrustedNodes()	Liefert eine Liste verfügbarer öffentlicher Trusted Nodes
login(clientID)	Anmeldung bei einem ContextNode
logout(clientID)	Reguläre Abmeldung des Benutzers vom System
publish(context)	Veröffentlichung eines Kontextes
remove(contextID)	Abmeldung eines Kontextes, der zuvor mit der gegebenen Context-ID angemeldet wurde
renew(contextID)	Auffrischung eines angemeldeten Kontextes
send(message)	Setzt eine zeitbezogene Nachricht im System ab

Abbildung 4.9: Benutzer-API eines zeitbezogenen Contextcast bei den ContextNodes

demnach auch nicht propagiert.

Client-ID:	LocClientID'@'TrustedNodeID
Attribut:	Name':'Wert

Die in Abbildung 4.9 dargestellte API wird dem Benutzer bzw. der Applikation eines zeitbezogenen Contextcast von den ContextNodes als Zugangsknoten zur Verfügung gestellt. Dies ist die primäre und einzige Schnittstelle des Benutzers zum System. Sollte ein Benutzer noch keinen globale Client-ID besitzen, der ihm beispielsweise von seinem ISP zur Verfügung gestellt wurde, kann er durch Aufruf der Funktion „getListOfTrustedNodes()“ eine Liste frei verfügbarer Trusted Nodes des Systems ermitteln und bei diesen einen Account für eine Mailbox und einen globalen ID beantragen. Die Vertrauenswürdigkeit der einzelnen Trusted Nodes kann beispielsweise durch Zertifikate sichergestellt werden. Die genaue Umsetzung bleibt der Implementierung des Systems überlassen.
Der Client-ID wird unter anderem für die An- und Abmeldung am ContextNode verwendet. Nach einer Abmeldung des Benutzers werden alle Kontexte abgemeldet, die mit der Client-ID des Benutzers beim ContextNode in Verbindung gebracht werden können. Für die Veröffentlichung von Kontexten wird die Funktion „publish(context)“ angeboten. Falls der Benutzer mehrere Kontexte anmeldet, kann durch den eindeutigen Context-ID Bezug auf einen spezifischen Kontext genommen werden. Die Funktion „remove(contextID)“ veranlasst die Abmeldung eines Kontextes mit dem gegebenen ID. Dagegen muss „renew(contextID)“ periodisch aufgerufen werden, um die weitere Gültigkeit eines Kontextes gegenüber dem ContextNode zu bestätigen. Über den Aufruf von „send(message)“ werden Nachrichten im System abgesetzt.

In diesem Kapitel wurde die Erweiterung des ursprünglichen Contextcast zu einem zeitbezogenen Contextcast ausführlich beschrieben. Das zeitbezogene Contextcast ermöglicht, über die Funktionalität des ursprünglichen Contextcast hinaus, eine Nachrichtenadressierung an historische und zukünftige Kontexte. Damit können Informationen über die vergangene und zukünftige Situation von Objekten direkt in den Versand der Nachrichten einfließen. Eine weitere zentrale Neuerung ist die Persistierung von Nachrichten. Diese können nun bei Abwesenheit eines Benutzers in einem Mailboxsystem hinterlegt werden, bis der Benutzer wieder erreichbar ist und eine Übertragung gewährleistet werden kann. Die Realisierung dieser Konzepte erfolgte unter strenger Einhaltung des Privatsphärenschutzes eines Benutzers. Alle bisher erläuterten Verfahren bilden zusammen einen vollständigen Entwurf eines zeitbezogenen Contextcast. Das folgende Kapitel beschäftigt sich mit Optimierungspotentialen des Entwurfs, um die Effizienz des Systems nach bestimmten Metriken weiter zu steigern. Der Schwerpunkt liegt dabei auf dem Routing der verschiedenen zeitbezogenen Nachrichten.

5 Exploration der Routingverfahren

Optimierungen der Routingverfahren und die damit verbunden Verbesserungen des Matchings zeitbezogener Nachrichten bilden den Schwerpunkt dieses Kapitels. Ziel hierbei ist die Steigerung der Effizienz des im letzten Kapitel erläuterten zeitbezogenen Contextcast. Nach einer kurzen Einführung folgt in Abschnitt 5.1 die Erläuterung der Verfahren zur Vermeidung von Nachrichtenduplikaten. Diese werden parallel zu den Routingverfahren der zeitbezogenen Nachrichten durchgeführt. Abschnitt 5.1.1 behandelt die Vermeidung von Nachrichtenduplikaten an den gleichen Empfänger, die an verschiedenen Komponenten des Overlays durch das Matching entstehen. Nachrichtenduplikate an unterschiedliche Empfänger durch das Matching von historischen und hybriden Nachrichten bilden den Schwerpunkt des Abschnitts 5.1.2. Die alternativen Verfahren werden durch die Anordnung in Lösungsräumen anhand verschiedener Eigenschaften voneinander abgegrenzt. Anschließend folgt in Abschnitt 5.2 die Erläuterung des Verfahrens zur verteilten Auswertung futuristischer Nachrichten mit dem Ziel der Reduzierung des Nachrichtenaufwands bei der Übermittlung futuristischer Nachrichten. Abschnitt 5.3 behandelt als letzter Abschnitt dieses Kapitels die Verteilung der archivierten Kontexte auf die Knoten des zeitbezogenen Contextcast.

In den vorherigen Kapiteln wurde ein vollständiger Entwurf eines zeitbezogenen Contextcast beschrieben, der den Versand historischer, zukünftiger und hybrider Nachrichten neben den bisher bekannten Nachrichten eines Contextcast ermöglicht. Der Einfachheit halber wurde bisher angenommen, dass die Speicherung der Kontexte und die Auswertung der Nachrichten zentral beim ContextNode durchgeführt werden. Die Verfahren zur Vermeidung von Nachrichtenduplikaten beschränkten sich ausschließlich auf die ContextNodes. Obwohl damit bereits ein funktionierendes System festgelegt wurde, werden diese Einschränkungen durch die in diesem Kapitel erläuterten Verfahren aufgehoben bzw. aufgelockert. Die Verfahren optimieren das System gleichzeitig hinsichtlich verschiedener Aspekte. Die Gliederung der Verfahren orientiert sich an den Zielen der Optimierungen:

- Die Vermeidung bzw. Reduzierung von Nachrichtenduplikaten im gesamten System,
- die verteilte Auswertung futuristischer Nachrichten und
- die Verteilung der Kontextdatenbank auf mehrere Knoten.

Die Verfahren der beiden zuerst genannten Punkte bilden den Schwerpunkt dieses Kapitels. Sie können simultan eingesetzt werden und verbessern die Skalierbarkeit des Systems, das als übergeordnetes Ziel definiert wurde. Die Optimierungen betreffen ausschließlich die Erweiterungen des ursprünglichen Contextcast zu einem zeitbezogenen Contextcast wie beispielsweise das Routing und Matching der zeitbezogenen Nachrichten. Die einzelnen Verfahren richten sich an die verschiedenen Kategorien der zeitbezogenen Nachrichten, wobei eine Aufspaltung der zeitbezogenen Nachrichten in die verschiedenen zeitlichen Bestandteile vermieden wird. Eine Aufspaltung hybrider Nachrichten kann beispielsweise basierend auf dem zeitlichen Attributfilter in historische und futuristische Nachrichten erfolgen. Nach einer solchen Aufspaltung können zwar mehrere der im Weiteren beschriebenen Verfahren auf eine Nachricht angewendet werden, gleichzeitig wird jedoch die Multicasteigenschaft des Contextcast-Routings zerstört. Anstatt einer einzelnen Nachricht müssten nach der Aufspaltung mehrere Nachrichten getrennt übermittelt werden.
Die einzelnen im Folgenden erläuterten Verfahren werden in Kapitel 6 bezüglich verschiedener Aspekte durch Simulation untersucht und qualifiziert.

5.1 Duplikatvermeidung

Nachrichtenduplikate entstehen durch die Vervielfältigung von Nachrichten. In manchen Situationen werden Nachrichtenduplikate benötigt, beispielsweise wenn eine Nachricht an mehrere Empfänger übermittelt werden soll. Selbst bei der Anwendung einer Multicast-Übertragung erreicht die Nachricht beim Routing einen Punkt, an dem sie auf mehreren Verbindungen versandt werden muss, um die einzelnen Empfänger zu erreichen. Auf jeder Verbindung wird dann eine Kopie der Nachricht an die über die Verbindung erreichbaren Empfänger geschickt. In anderen Fällen, z.B. beim Matching, werden Nachrichtenduplikate erzeugt, die eventuell an die gleichen Empfänger übermittelt werden. Solche Nachrichtenduplikate sind überflüssig und belasten das System durch einen erhöhten Nachrichtenaufwand ohne den Mehrwert des Systems aus Benutzersicht zu steigern. Ein hoher Nachrichtenaufwand reduziert die Skalierbarkeit des Systems.
Ein weiterer Nachteil von Nachrichtenduplikate betrifft die Unterstützung mobiler Clients. Mobile Clients befinden sich aufgrund der geringen Ressourcenverfügbarkeit oft in unterschiedlichen Energiesparmodi. Jede Nachrichtenübermittlung an einen mobilen Client erzeugt bei diesem einen erhöhten Stromverbrauch. Möglicherweise muss zusätzlich bei jeder Nachrichtenübermittlung der entsprechende Stromsparmodus des Clients verlassen werden, was zu Folge hat, dass der Energieverbrauch weiter angehoben wird. Die Vermeidung von Nachrichtenduplikaten ist demnach ein primärer Optimierungsansatz.

Die Nachrichtenduplikate eines Contextcast können unter verschiedenen Umständen und an verschiedenen Orten entstehen. Existent sind Nachrichtenduplikate, die an ein und denselben Empfänger übermittelt werden und Nachrichtenduplikate für unterschiedliche Empfänger. Bereits erwähnt wurde, dass die Entstehung von Nachrichtenduplikaten für unterschiedliche Empfänger kein Problem darstellen muss. Ein Problem ergibt sich erst, wenn die Nachricht vorerst an bestimmte ContextNodes übermittelt und erst anschließend an die Benutzer als eigentlichen Empfänger weitergeleitet wird, wie es bei historischen Nachrichten der Fall ist. Die Nachricht wird dann recht früh in mehrere Duplikate aufgeteilt und von den einzelnen ContextNodes getrennt an die eigentlichen Empfänger per Contextcast übermittelt. Eine Übermittlung der Nachrichten an die Empfänger durch eine gemeinsame Multicast-Übermittlung könnte den Nachrichtenaufwand reduzieren. Im Folgenden wird strikt zwischen der Duplikatvermeidung von Nachrichten mit dem gleichen Empfänger und der Duplikatvermeidung bzw. -reduzierung von Nachrichten mit unterschiedlichen Empfängern aber gleichem Inhalt durch Multicast-Übermittlung unterschieden.

5.1.1 Nachrichtenduplikate mit gleichem Empfänger

Eine frühzeitige Erkennung von Nachrichtenduplikaten an den gleichen Empfänger ist anzustreben, damit ein möglichst geringes Verkehrsaufkommen an Nachrichten durch Duplikate erreicht wird. Prinzipiell kann die Erkennung von Nachrichtenduplikaten an drei Orten vollzogen werden: Bei den ContextNodes, den Trusted Nodes und den Clients selbst.

Die Erkennung entspricht einer Prüfung, ob eine Nachricht bereits an einen bestimmten Benutzer ausgeliefert wurde und ob mehrere identische Nachrichten an den gleichen Empfänger übermittelt werden sollen, die dann zu einer Nachricht zusammengefasst werden können. In Abschnitt 4.5 wurde bereits ein Verfahren zur Duplikaterkennung beschrieben das Duplikate erkennt, welche durch das wiederholte Matching einer Nachricht auf ein und denselben Kontext entstehen. Darüber hinaus entstehen Duplikate durch das Matching mit unterschiedlichen Kontexten des gleichen Client-ID bei einem ContextNode.
Beim Matching historischer Kontexte können diese Nachrichtenduplikate verhindert werden, da eine vollständige Auswertung direkt und ohne zeitliche Verzögerung durchgeführt wird. Für hybride und futuristische Nachrichten müsste bei einer Duplikaterkennung durch den

ContextNode zu jeder Nachricht eine Liste von Client-IDs jedes Benutzers abgespeichert werden, welche die Nachricht bereits erhalten haben. Dieses Verfahren erzeugt einen erheblichen Speicheraufwand und kann außerdem nur für Duplikate des gleichen ContextNodes angewandt werden.
Zeitbezogene Nachrichten können durch die Lokationsangabe für das Empfängergebiet an mehrere ContextNodes adressiert werden. Im Zuge des Routings werden die Nachrichten repliziert und an die ContextNodes übermittelt. Ein Benutzer kann und wird auch höchstwahrscheinlich seine Position im zeitlichen Verlauf ändern. Dadurch entstehen historische Kontexte bei den ContextNodes, an denen ein Benutzer angemeldet war. Wenn eine zeitbezogene Nachricht mehrere dieser ContextNodes adressiert und mit mehreren historischen Kontexten eines Benutzers übereinstimmt, entstehen Nachrichtenduplikate an unterschiedlichen ContextNodes, die an den gleichen Mailboxknoten übermittelt werden. Dies gilt hauptsächlich für historische Nachrichten, da futuristische Nachrichten und hybride Nachrichten für gegenwärtig angemeldete und zukünftige Kontexte direkt an die Empfänger übermittelt werden. Nachrichten, die aufgrund der Abwesenheit eines Benutzers nach dem Ablauf einer TTL nicht übertragen werden können, bilden eine Ausnahme und werden ebenfalls an den Mailboxknoten übermittelt. Zusammenfassend werden also folgende Szenarien für die Entstehung von Nachrichtenduplikaten mit dem gleichen Empfänger unterschieden:

1. Duplikate durch Mehrfachauswertungen einer Nachricht auf ein und denselben Kontext.

2. Duplikate durch Auswertungen einer Nachricht auf mehrere Kontexte eines Benutzers.

3. Duplikate durch Auswertungen einer Nachricht bei unterschiedlichen ContextNodes auf mehrere Kontexte eines Benutzers.

Nicht jedes dieser Szenarien kann durch eine Duplikaterkennung an einem beliebigen Knoten behandelt werden. Die ContextNodes, Trusted Nodes und Benutzer haben, bedingt durch das Routing der Nachrichten, unterschiedliche breite Sicht auf die Duplikate einer Nachricht. Ein ContextNode beispielsweise erkennt keine Duplikate, die von unterschiedlichen ContextNodes erzeugt werden und an den gleichen Empfänger adressiert sind.

Unabhängig von dem Ort, an welchem die Duplikaterkennung durchgeführt wird, müssen Informationen gespeichert werden, die eine Identifikation über das mehrfache Auftreten einer Nachrichten zulassen. Wird die Duplikaterkennung an einem mit dem Benutzer verknüpften Ort durchgeführt, wie beispielsweise dem Trusted Node eines Benutzers, muss ein eindeutiger Bezeichner für die erhaltenen Nachrichten abgespeichert werden. Dies gilt natürlich nur für Nachrichten, die in Zukunft nochmals eintreffen könnten, deren Lebensdauer also noch nicht abgelaufen ist. Diese Bedingung lässt sich aus den Löschbedingungen der Nachrichten ableiten. Ab dem Zeitpunkt, bei dem die Löschbedingung einer Nachricht zu *true* ausgewertet wird, entfernt das System die Nachricht. Dies geschieht durch das Löschen der Nachricht an all den Knoten, an denen sie gespeichert wurde und aus dem Grund, dass ab diesem Zeitpunkt kein neuer Kontext mehr mit der Nachricht übereinstimmen kann. Kann ein erfolgreiches Matching der Nachricht nicht mehr durchgeführt werden, wird der Benutzer diese Nachricht auch in Zukunft nicht mehr erhalten. Aus diesem Grund können auch bereits gespeicherte Bezeichner von Nachrichten, für die der Löschzeitpunkt erreicht wurde, gelöscht werden. Da Nachrichten durch das Matching und die Übertragung im Netz verzögert beim Empfänger eintreffen, muss der Löschzeitpunkt für die Bezeichner um ein festgesetztes δ verzögert werden. Für den Bezeichner der Nachrichten kann bei der Implementierung des Systems ein beliebiges eindeutiges Merkmal, z.B. der Nachrichten-ID oder ein Hashwert der Nachricht, gewählt werden.

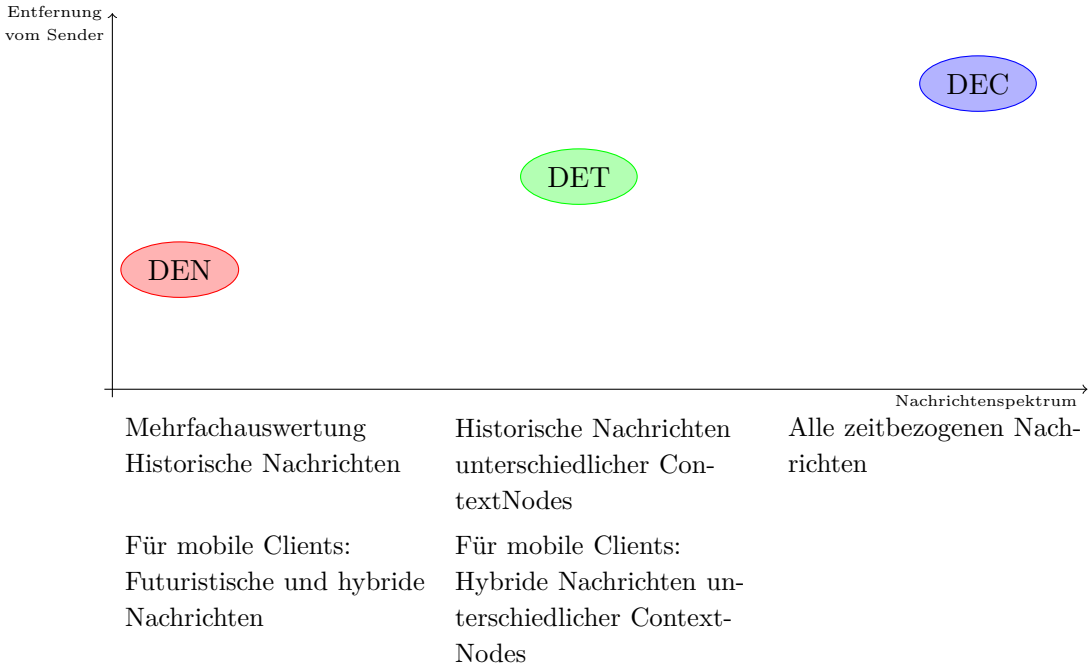


Abbildung 5.1: Duplikaterkennung

Wird für die Duplikaterkennung ein mit der Nachricht verknüpfter Ort, z.B. der Senderknoten, gewählt, muss für die Nachricht eine Liste der Empfänger gespeichert werden, bis der Löschzeitpunkt der Nachricht erreicht ist. Die Löschung der Nachricht muss in diesem Fall nicht verzögert werden. Da eine Benutzeridentifizierung aus Gründen des Schutzes der Privatsphäre nur auf Basis des Client-ID möglich ist und dieser im Laufe der Zeit durch den Benutzer gewechselt werden kann, ist dieser Ansatz mit Vorsicht zu genießen. Bei häufigem ID-Wechsel können keine Duplikate mehr erkannt werden und die mit den Nachrichten gespeicherten Benutzerlisten wachsen endlos an, was einen nicht geringen Speicheraufwand erzeugt. Da eine Nachricht im Contextcast nicht zentral ausgewertet wird, kann auch die Duplikaterkennung an einem mit der Nachricht verknüpften Ort nicht alle Nachrichtenduplikate für einen Empfänger ermitteln.

Abbildung 5.1 zeigt den Lösungsraum für die Duplikatvermeidung an verschiedenen Orten. Entlang der Abszisse ist das Nachrichtenspektrum aufgetragen, das mit einer Duplikatvermeidung an einem bestimmten Ort erfasst werden kann. Die Ordinate bildet die Entfernung eines Knotens vom Ursprung der Nachricht ab. Aus dieser Darstellung heraus lässt sich ein Trade-off erkennen: Je weiter der gewählte Knoten für die Duplikatvermeidung vom Ursprung der Nachricht entfernt ist, desto breiter ist das Spektrum der verfügbaren Nachrichten zur Duplikaterkennung, aber desto größer auch der bereits entstandene Nachrichtenaufwand durch Duplikate. Die Duplikatvermeidung darf also nicht ausschließlich an einem Ort vollzogen werden. Im Gegenteil: Duplikate sollten mit einem vertretbaren Speicheraufwand so früh wie möglich erkannt und gelöscht werden. Aus diesem Grund empfiehlt sich eine Anwendung aller in Abbildung 5.1 dargestellten Verfahren zur Duplikatvermeidung.

Die Duplikaterkennung beim ContextNode, kurz DEN, verhindert Duplikate durch Mehrfachauswertungen auf gleiche Kontexte, Duplikate bei der Auswertung der historischen Nachrichten und gegebenenfalls Duplikate durch das Matching hybrider und futuristischer Nachrichten auf Kontexte des gleichen Benutzers bei einem ContextNode im zeitlichen Verlauf. Dazu werden unter Umständen die Listen der bereits mit der Nachricht erreichten Empfänger abgespeichert.

Gerade bei mobilen Clients ist ein erhöhter Speicheraufwand für die Duplikatvermeidung eventuell zu rechtfertigen, da nur beschränkte Energiequellen, eine möglicherweise schlechte Verbindung mit geringer Bandbreite und begrenzter Speicher zur Verfügung stehen. In diesem Fall kann ein Teil der sonst vom Client durchgeführten Duplikaterkennung durch den ContextNode abgenommen werden. Dazu meldet sich der mobile Client mit einem Kontext und einem speziellen boolschen Attribut an. Dieses Attribut weist den Client beim ContextNode als mobilen Client aus. Für hybride und futuristische Nachrichten wird für jeden mobilen Client bei einem erfolgreichen Matching mit einem einem ihrer Kontexte der Client-ID gespeichert. Bei einer erfolgreichen Übertragung der Nachricht an den Client wird anschließend kein Kontext des Clients mehr auf die Nachricht ausgewertet. Durch diese Maßnahme wird eine einmalige Übertragung der Nachricht durch den ContextNode an den Client garantiert. Das zusätzliche Attribut ist nur für den ContextNode relevant, an dem sich der Client angemeldet hat. Es wird bei der Archivierung eines Kontextes auch mit in die Kontextdatenbank übernommen, so dass eine mehrfache Auswertung hybrider Nachrichten auf die Kontexte des Benutzers verhindert werden kann. Die Duplikaterkennung durch den Client selbst wird dadurch nicht hinfällig. Bei einem Wechsel des Client zu einem anderen ContextNode, bei dem dieselbe Nachricht gespeichert ist, wird die Nachricht bei einem erfolgreichen Matching zumindest ein Mal neu an den Client übertragen.

Bei historischen Nachrichten, die vom Sender zum ContextNode und über den Trusted Node hin zum eigentlichen Empfänger geroutet werden, stellt der Client den letzten Knoten auf dem Pfad der Nachricht dar. Dies gilt ebenfalls für futuristische Nachrichten, die idealerweise durch den ContextNode direkt an den Empfänger geleitet werden. Bei ihm gehen alle für ihn bestimmten Nachrichten ein. Eine Duplikaterkennung ausschließlich beim Client wäre völlig ausreichend, um alle Nachrichtenduplikate vor der Zustellung an die Anwendungen zu filtern. Wenn man sich die langen Pfade speziell der historischen Nachrichten anschaut, stellt dies jedoch keine effiziente Methode dar. Auf Grund dessen wird vorab eine Filterung beim ContextNode und Trusted Node vollzogen. Bestimmte Duplikate, die z.B. wie bereits beschrieben bei einem Standortwechsel des Benutzers entstehen können, können jedoch nur durch eine Filterung beim Client vermieden werden.

Bei dem Trusted Node eines Benutzers, der sich auf dem gleichen Knoten wie das Mailboxsystem des Benutzers befindet, gehen nur die historischen Nachrichten, hybride Nachrichten und Nachrichten ein, die durch die Abwesenheit von Benutzern nicht direkt zugestellt werden konnten. Die Nachrichten treffen dort zentral von den verschiedenen ContextNodes, an denen der Benutzer in der Vergangenheit einen Kontext angemeldet hat, für den Benutzer zusammen. Eine Duplikaterkennung beim Trusted Node, kurz DET, kann daher Nachrichten mit einbeziehen, die von den verschiedensten ContextNodes stammen. Futuristische Nachrichten und hybride Nachrichten, die zukünftig mit einem Kontext des Benutzers übereinstimmen, erhält der Benutzer direkt durch den ContextNode und nicht über das Mailboxsystem. Die DET filtert somit hauptsächlich Duplikate historischer Nachrichten von verschiedenen ContextNodes, die an einen Benutzer adressiert sind. Für mobile Clients besteht hier ebenfalls eine optionale Optimierungsmöglichkeit. Werden hybride Nachrichten auf den Kontext eines mobilen Clients ausgewertet und direkt durch den ContextNode zugestellt, kann der ContextNode zusätzlich einen eindeutigen Bezeichner der Nachricht an das Mailboxsystem des mobilen Client senden. Dadurch kann beim Trusted Node eine Duplikaterkennung mit den Nachrichten durchgeführt werden, die durch das Matching der hybriden Nachricht bei einem anderen ContextNode auf historische Kontexte des gleichen Benutzers entstehen könnten. Selbstverständlich ist diese Optimierung nicht nur an mobile Clients gebunden, doch gerade bei mobilen Clients ist oftmals ein zusätzlicher Speicher- und Nachrichtenaufwand für die Duplikaterkennung zu rechtfertigen.

Bei der Anwendung der Duplikatvermeidung durch die ContextNodes und Trusted Nodes

bleiben für die Duplikaterkennung durch den Client selbst nur noch Nachrichten übrig, die durch die ContextNodes direkt an den Benutzer übermittelt werden. Streng genommen entstehen diese Duplikate nur bei einem Client-ID- oder Standortwechsel des Benutzers, da ein ContextNode nicht weiß, welcher andere ContextNode die gleiche Nachricht bereits an den Benutzer übermittelt hat. Der Client-ID ist aus Gründen des Schutzes der Privatsphäre der einzig eindeutige Bezeichner des Benutzers. Nach einem Wechsel dieses Bezeichners kann ein ContextNode nicht mehr ermitteln, ob eine Nachricht bereits an den Benutzer übertragen wurde, da dieser aus Sicht des ContextNodes nun eine neue Identität darstellt. Bei mobilen Clients empfiehlt sich infolgedessen ein Wechsel des Client-ID erst bei einem Standortwechsel, allgemein zwischen der Abmeldung und Anmeldung eines Kontextes, sodass ferner eine optimale Roamingunterstützung umgesetzt werden kann.

5.1.2 Gebündeltes Contextcast historischer Nachrichten

Im Unterschied zu der Duplikatvermeidung für Nachrichten, die an den gleichen Empfänger gerichtet sind, werden Nachrichtenduplikate an unterschiedliche Empfänger nicht verworfen, sobald sie erkannt wurden. Da sich die Empfänger an unterschiedlichen Orten befinden, muss die Nachricht zwangsläufig dupliziert werden. In diesem Abschnitt werden Duplikate behandelt, die sich bis auf die Empfängerangaben gleichen.

Zeitbezogene Nachrichten enthalten immer eine Ziellokation, auf Basis derer die ContextNodes als vorläufige Empfänger ermittelt werden. Auf dem Routing zu den Empfängern wird die Nachricht an verschiedenen Knoten aufgespalten, an denen eine Kopie der Nachricht an die Verbindungen zu den ContextNodes geschickt wird. Dies setzt voraus, dass die Nachricht mehrere ContextNode-Empfänger enthält. Für historische Nachrichten und den historischen Nachrichtenteil einer hybrider Nachricht wird dann ein Matching auf die verschiedenen Kontextdatenbanken der ContextNodes durchgeführt. Im Anschluss wird die Nachricht an die Trusted Nodes der Empfänger mittels Xcast weitergeleitet, bei der statt den IDs der ContextNode-Empfänger die IDs der Trusted Nodes enthalten sind. Die weitergeleiteten Nachrichten selbst sind bei den unterschiedlichen ContextNodes die gleichen. Sie unterscheiden sich nur in den Empfängerlisten der Trusted Nodes und der Benutzer. Ein Ansatz zur Duplikatvermeidung und damit Reduzierung der Nachrichtenlast des zeitbezogenen Contextcast-Systems ist in diesem Zusammenhang die ausgefeiltere Umsetzung der Multicast-ähnlichen Übertragung. Die hier betrachteten Duplikate lassen sich durch eine Nachricht substituieren, die an die Vereinigungsmenge der Empfänger der einzelnen Nachrichten adressiert ist. Diese kann dann mittels Xcast an die gesamte Menge an Empfängern übermittelt werden. Ein schöner Nebeneffekt dieser Methode ist die Eliminierung der einzelnen Duplikate, die von den unterschiedlichen ContextNodes an den gleichen Empfänger versendet werden. Diese Duplikate werden ebenfalls durch die Duplikaterkennung beim Trusted Node ausgefiltert. Eine Filterung beim Trusted Node verhindert jedoch nicht den Nachrichtenaufwand, der bei der Übermittlung der Duplikate von den ContextNodes zu den Trusted Nodes entsteht. Selbst wenn ein Benutzer seinen Client-ID im zeitlichen Verlauf gewechselt hat, so bleibt sein Trusted Node, der ebenfalls Teil des Client-ID ist und im Regelfall nicht gewechselt wird, derselbe. Bei der Weiterleitung der Nachricht sind die Trusted Nodes die vorübergehenden Empfänger, so dass zwar mehrere Client-IDs eines Benutzers in der neu konstruierten Nachricht enthalten sein können, aber nur eine Nachricht an den betreffenden Trusted Node verschickt wird. Mit den im Folgenden beschriebenen Verfahren wird die Filterung beim Trusted Node jedoch nicht überflüssig, da die unterschiedlichen Client-IDs eines Benutzers ausgefiltert werden müssen und eventuell eine zusätzliche Duplikaterkennung für mobile Clients erfolgt. Die Verfahren lassen sich nur auf historische Nachrichten und den historischen Nachrichtenteil hybrider Nachrichten anwenden. Eine Integration des

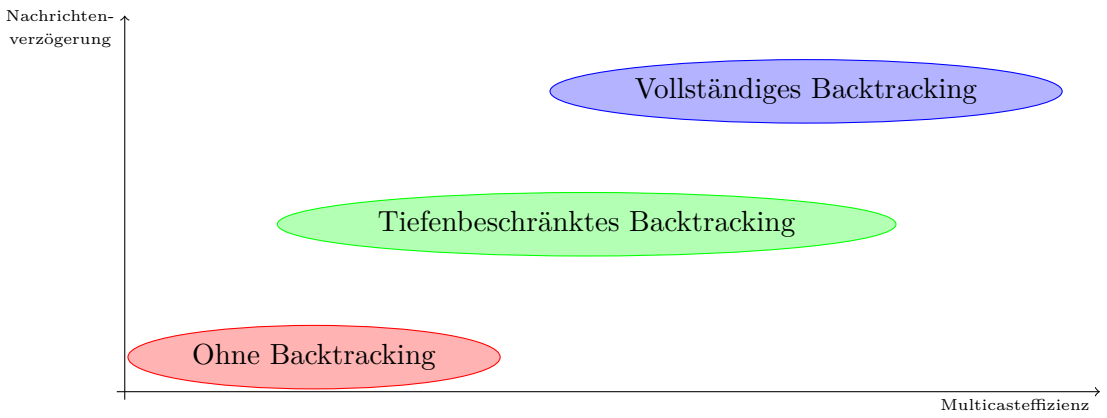
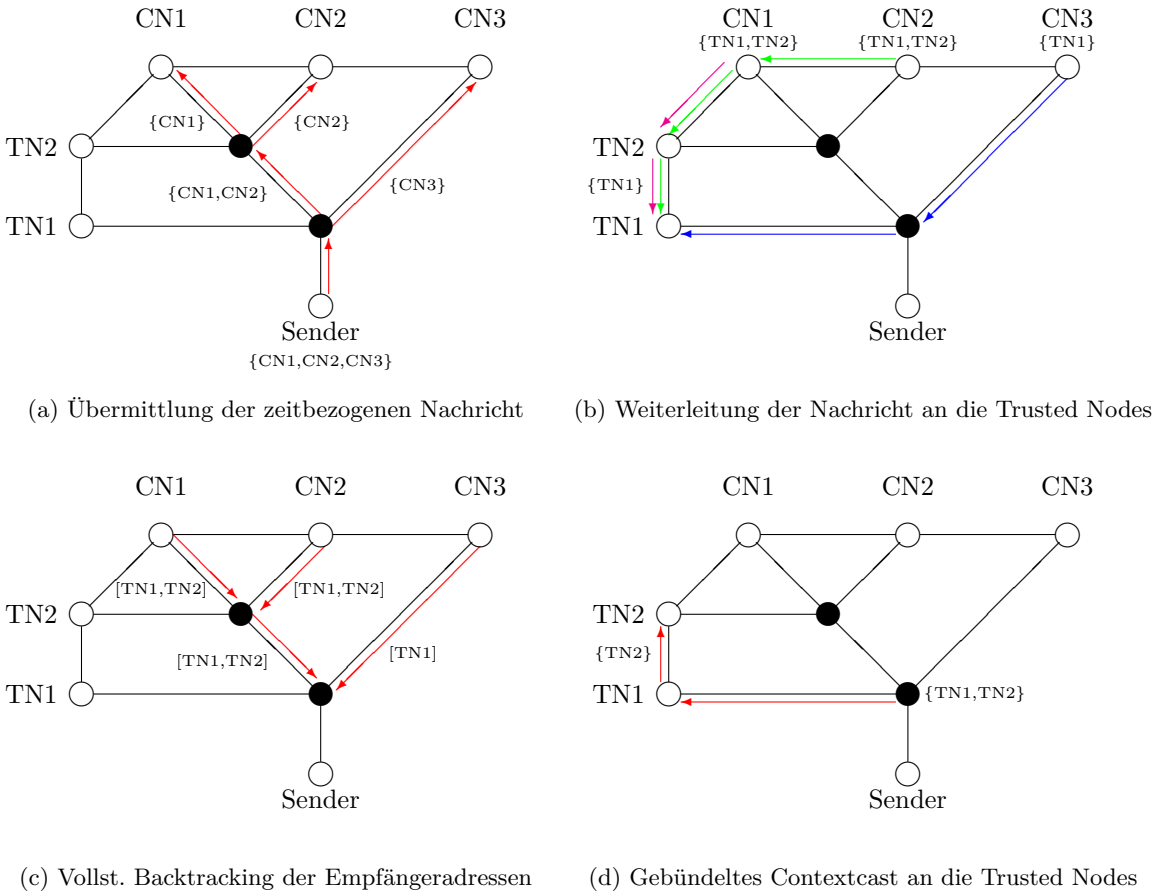


Abbildung 5.2: Backtracking Varianten

futuristischen Nachrichtenteils würde die Nachrichtenankunft bei den Trusted Nodes zu lange verzögern, da sich die Auswertung futuristischer Nachrichten im Allgemeinen über einen längeren Zeitraum erstreckt.

Backtracking-Verfahren im Vergleich Abbildung 5.2 zeigt die im Weiteren erläuterten Verfahren zur Reduzierung des Nachrichtenaufwands bei der Weiterleitung der historischen Nachrichten auf. Die Variante ohne *Backtracking* entspricht der bereits beschriebenen Methode zur Weiterleitung der Nachrichten. Sobald das Matching bei den ContextNodes durchgeführt wurde, kann die Nachricht sofort an die Trusted Nodes der Empfänger weitergeleitet werden. Beim vollständigen Backtracking wird versucht, die einzelnen Nachrichten als eine gebündelte Nachricht über Xcast an die Trusted Nodes zu versenden. Dazu wird ein bzw. werden mehrere Mergepoints benötigt, an denen die einzelnen Nachrichten zusammengefasst werden können. Für das Backtracking-Verfahren wurden dafür die Splitpoints gewählt, an denen die Nachricht auf dem Weg zu den ContextNodes aufgespaltet wurde. Die Nachrichten lassen sich auf dem Reverse Path des Verteilbaums in Richtung Ursprung der Nachricht wieder zu einer Nachricht zusammenführen. Damit dies möglich wird, muss jeder Knoten auf dem Hinweg der ursprünglichen Nachricht die Verbindung mit einem Bezeichner der Nachricht verknüpfen, auf der die Nachricht bei dem Knoten angekommen ist. Dadurch weiß der Knoten, auf welcher Verbindung er die Nachricht beim Backtracking wieder zurücksenden muss. Eine Alternative stellt die Übermittlung des jeweils letzten Splitpoint-Knoten mit der ursprünglichen Nachricht auf dem Hinweg dar. Die Nachrichten lassen sich dann gezielt auf dem Reverse Path Schritt für Schritt zu dem vorherigen Splitpoint-Knoten übertragen. ContextNode-Empfänger der Nachricht, die auf dem Pfad zu weiteren Empfängern liegen, werden ebenfalls als Splitpoint-Knoten betrachtet. Im Unterschied zu den Mergepoints wird bei diesen Knoten zusätzlich ein lokales Matching durchgeführt.

Außer dass bei der Weiterleitung der Nachricht zu den Trusted Nodes die Attributfilter entfernt werden und die Nachricht neue Empfängeradressen erhält, bleibt die Nachricht unverändert, d.h. der Nachrichteninhalt selbst wird nicht verändert. Aus dieser Tatsache heraus erscheint es überflüssig, die gesamte Nachricht zwei Mal an die Splitpoints zu übertragen, einmal auf dem Hinweg der Nachricht, das andere Mal beim Backtracking. Um dies zu verhindern, wird die Nachricht auf dem Hinweg bei dem ersten Splitpoint-Knoten zwischengespeichert. Die einzelnen ContextNode-Empfänger senden nach dem Matching nur noch eine Nachricht mit einem Bezeichner der ursprünglichen Nachricht und den Empfängeradressen an die Splitpoint-Knoten, die durch das Matching ermittelt wurden. An jedem Splitpoint-Knoten muss auf die einzelnen Nachrichten der ContextNodes gewartet werden, an welche



(a) Übermittlung der zeitbezogenen Nachricht

(b) Weiterleitung der Nachricht an die Trusted Nodes

(c) Vollst. Backtracking der Empfängeradressen

(d) Gebündeltes Contextcast an die Trusted Nodes

Abbildung 5.3: Veranschaulichung des Backtracking-Verfahrens an einem Beispiel

die historische Nachricht zuvor übermittelt wurde. Daraufhin werden die Empfängeradressen zusammengefasst, Duplikate in den Adresslisten entfernt und eine neue Nachricht mit den Empfängeradressen an den nächsten Splitpoint-Knoten übermittelt. Diese Prozedur wiederholt sich bis der letzte Splitpoint-Knoten erreicht ist. Von hier aus kann die eigentliche Nachricht nun mittels Xcast an die empfangene Liste der Trusted Nodes gesendet werden. Im Anschluß daran kann die zwischengespeicherte Nachricht entfernt werden. Die Erkennung des ersten Splitpoint-Knotens geschieht auf dem Hinweg der Nachricht durch das Setzen eines Flags. Der erste Splitpoint-Knoten setzt eine boolsche Konstante in der Nachricht die besagt, dass der erste Splitpoint-Knoten schon gefunden wurde und speichert die Nachricht. Alle weiteren Splitpoint-Knoten erkennen die Konstante und leiten die Nachricht einfach weiter. Das Bit für die Konstante kann nach dem Empfang der Nachricht durch die ContextNodes entfernt werden.

In Abbildung 5.3 ist ein Beispiel für ein vollständiges Backtracking dargestellt. Dabei soll eine historische Nachricht an drei verschiedene ContextNodes gesendet werden. Die erste Abbildung zeigt die Pfade für die Übermittlung der Nachricht an die ContextNodes als rote Pfeile. Die ausgefüllten Kreise stellen ContextNodes und Trusted Nodes dar, die nicht ausgefüllten Kreise sind ContextRouter. In geschweiften Klammern sind die Adressaten der einzelnen Nachrichten angegeben. In der zweiten Abbildung sind die Nachrichten zu sehen, die durch die von der Nachricht adressierten ContextNodes an die Trusted Nodes nach dem ursprünglichen Verfahren weitergeleitet werden. Die Adressen der Trusted Nodes, die zusammen mit den Benutzern als Empfänger der Nachricht beim Matching ermittelt wurden,

sind bei den ContextNodes in geschweiften Klammern angegeben. Für das Backtracking sind nur die Nachrichten bis zu den Trusted Nodes relevant. Daher sind die Benutzer als endgültige Empfänger der ursprünglichen Nachricht nicht abgebildet. Für die Übermittlung der Nachrichten an die Trusted Nodes sind in diesem Beispiel sieben Nachrichten nötig.
Die dritte Abbildung zeigt die bei einem vollständigen Backtracking verschickten Nachrichten. Die in den Nachrichten beinhalteten Trusted Nodes, an welche die gebündelte Contextcast-Nachricht nach dem Backtracking adressiert werden soll, sind in eckigen Klammern angegeben. Duplikate der Trusted Nodes und Client-IDs werden jeweils bei den Splitpoint-Knoten entfernt. Am Ende des Backtrackings stehen die Knoten TN1 und TN2 als Empfänger der gebündelten Contextcast-Nachricht fest. Für die Übermittlung der Nachricht durch den ersten Splitpoint-Knoten an die Trusted Nodes sind insgesamt zwei Nachrichten im Gegensatz zu den sieben Nachrichten des ursprünglichen Verfahrens erforderlich, wie die vierte Abbildung zeigt. Die Ersparnis des Nachrichtenaufwands durch das Backtracking-Verfahren liegt in diesem Beispiel bei ungefähr 70 Prozent. Dieser Ersparnis muss natürlich noch der Mehraufwand an Nachrichten gegenübergestellt werden, der durch das Backtracking erzeugt wird. Im Gegensatz zu den Contextcast-Nachrichten sind die Backtracking Nachrichten jedoch sehr klein, da sie nur die durch das Matching erhaltenen Client-IDs transportieren und nicht den kompletten Nachrichteninhalt.

Durch das Backtracking werden einzelne Contextcast-Nachrichten zu einer Contextcast-Nachricht zusammengeführt. Wie im Vergleich zwischen Unicast und Multicast werden dadurch Nachrichten eingespart und damit die Nachrichtenbelastung des Netzes reduziert. Beim Backtracking ist dazu hingegen ein zusätzlicher Nachrichtenaufwand nötig. Die Nachrichten, die beim Backtracking verschickt werden und den Splitpoints die Client-IDs mitteilen, reduzieren die Einsparung an Nachrichten durch das gebündelte Contextcast. Ein weiterer Nachteil des Backtracking ist die Verzögerung der Nachrichtenankunft bei den Trusted Nodes. Beim Backtracking muss bei jedem Splitpoint auf die Antworten aller Knoten gewartet werden, an die eine Nachricht auf dem Hinweg zu den ContextNodes verschickt wurde. Stellt man den Verteilweg der Nachricht grafisch als Baum dar, so sind das alle Kindknoten des jeweiligen Splitpoint-Knoten. Erst beim ersten Splitpoint-Knoten wird die Nachricht dann endgültig verschickt. Dieser Knoten wird beim Backtracking als letzter Knoten erreicht. Die Nachricht wird also mindestens um die Zeit verzögert, die das Backtracking selbst benötigt. Weitere Verzögerungen können durch Pfadverlängerungen entstehen, wenn der erste Splitpoint-Knoten weiter von des Trusted Nodes entfernt liegt als die einzelnen ContextNodes. Es liegt also eine Trade-off Beziehung zwischen dem erzeugten Nachrichtenaufwand und der Nachrichtenübertragungszeit vor. Wie Abbildung 5.2 zu entnehmen ist, erzeugt ein vollständiges Backtracking die längste Verzögerung.

Die Effizienz der Reduzierung des Nachrichtenaufwands, die durch ein gebündeltes Backtracking angestrebt wird, ist von verschiedenen Faktoren abhängig. Dies sind beispielsweise die Struktur und die Verbindungseigenschaften des Netzes, die Anzahl der Trusted Nodes als Ziele für die Weiterleitung der Nachricht, die Position der Trusted Nodes im Netz, die Anzahl an Duplikaten, die bei einem Matching bei den ContextNodes ermittelt werden, die Anzahl der beteiligten ContextNodes und die Tiefe und Breite des Verteilbaums. Die Tiefe des Verteilbaums gibt Auskunft über die Länge der Pfade beim Backtracking. Sie ist demnach ausschlaggebend für den zusätzlichen Nachrichtenaufwand und die zusätzliche Verzögerung. Die Breite des Verteilbaums wird durch die Anzahl der beteiligten ContextNodes bestimmt. Generell wird der Nachrichtenaufwand des Backtrackings, ohne die Betrachtung der Mengen von Trusted Nodes, die beim Matching als Empfänger ermittelt werden, mit einer steigenden Anzahl an ContextNodes erhöht. Gleichzeitig erhöht sich damit aber auch die Einsparung der Nachrichten durch ein gebündeltes Contextcast. Der wichtigste Indikator für den Aufwand des Backtrackings ist demnach die Tiefe des Verteilbaums. Die Lokalitätseigenschaften

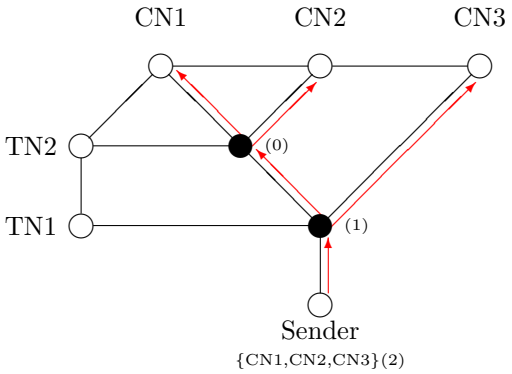
des Contextcast begünstigen das Backtracking hinsichtlich des Nachrichtenaufwands. Knoten, die sich geographisch nah beieinander befinden, sind auch im Contextcast über kurze Pfade verbunden. Da Nachrichten im Contextcast ursprünglich an ein bestimmtes Gebiet adressiert werden, liegen die ContextNode-Empfänger der Nachrichten eng beieinander. Dadurch wird eine Aufspaltung der Nachricht bei der Übertragung relativ spät durchgeführt, was wiederum zu kürzeren Verteilbäumen führt, als wenn die Knoten des Netzes beliebig angeordnet wären.

Aus der Tatsache heraus, dass der Aufwand des Backtrackings die Einsparung an Nachrichten durch ein gebündeltes Contextcast reduziert, stellt sich die Frage, wann ein vollständiges Backtracking dann noch sinnvoll ist. Ohne Betrachtung der Nachrichtenverzögerung lohnt ein Backtracking, wenn der zusätzlich erzeugte Nachrichtenaufwand die Einsparung an Nachrichten nicht übersteigt. Eine derartige Gegenüberstellung ist im praktischen Umfeld jedoch nicht quantifizierbar. Die Einsparung an Nachrichten durch ein gebündeltes Contextcast lässt sich bereits quantitativ ohne globale Kenntnis der Struktur des Netzes nicht erfassen.

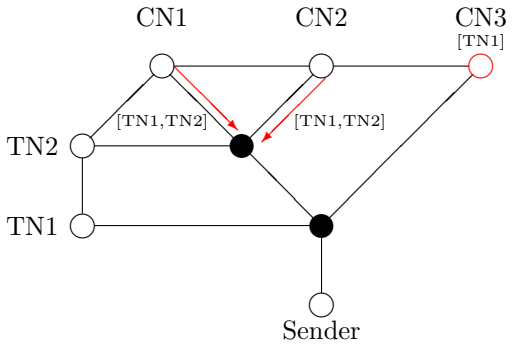
Ein „tiefenbeschränktes Backtracking“[1] zielt auf die Reduzierung des zusätzlichen Aufwands des Backtrackings und eine Verringerung der Nachrichtenverzögerung ab. Dies geschieht auf Kosten der Einsparung an Nachrichten durch ein gebündeltes Contextcast. Es stellt ein Kompromiss zwischen dem vollständigen Backtracking und dem ursprünglichen Verfahren der Weiterleitung von Nachrichten durch die ContextNodes dar. Beim tiefenbeschränkten Backtracking kann die Tiefe bzw. der Level der Splitpoint-Knoten des Verteilbaums – im Folgenden „TREELEVEL“ genannt – gewählt werden, bis zu welchem ein Backtracking erfolgen soll. Mit zunehmendem TREELEVEL bildet das tiefenbeschränkte Backtracking den Übergang vom vollständigen Backtracking zur ursprünglichen Methode der Nachrichtenverteilung ohne Backtracking. Entspricht der gewählte Level nicht dem des ersten Splitpoint-Knoten, was gleichbedeutend mit einem vollständigen Backtracking ist, so werden mehrere gebündelte Contextcast-Nachrichten durch verschiedene Splitpoint-Knoten an die Trusted Nodes versendet. Eine vollständige Duplikatvermeidung der historischen Nachrichten unterschiedlicher ContextNodes wird dann im Gegensatz zum vollständigen Backtracking nicht mehr durchgeführt. Eine Duplikatvermeidung bei den Trusted Nodes gewinnt damit wieder an Bedeutung, auf Kosten des Nachrichtenaufwands durch die Zustellung der Duplikate. Die Wahl des Levels und damit der Splitpoint-Knoten, bis zu welchen ein Backtracking erfolgen soll, kann prinzipiell auf dem Weg der Nachricht hin zu den ContextNodes oder auf dem Rückweg der Nachricht beim Backtracking selbst getroffen werden. Eine einfache Methode zur Wahl der Splitpoint-Knoten auf dem Hinweg der Nachricht zu den ContextNodes ist das TTL-basierte Verfahren. Dazu wird mit der Nachricht ein Zähler mitgeführt, der durch jeden Splitpoint-Knoten bei der Verteilung der Nachricht um eins verringert wird. Der Wert, auf den der Zähler anfangs, beispielsweise durch den ContextNode des Senders, gesetzt wird, bestimmt den TREELEVEL des tiefenbeschränkten Backtrackings. Die Nachricht wird bei den Splitpoint-Knoten zwischengespeichert, bei denen der Zähler auf Null steht. Der Zähler kann dann bei diesen Knoten verworfen werden, wird somit nicht weiter benötigt und muss auch nicht zwischengespeichert werden. Das boolesche Attribut für den ersten Splitpoint-Knoten in der Nachricht muss ebenfalls gesetzt werden. Die damit gewählten Splitpoint-Knoten werden die Nachricht nach dem Backtracking der Empfängeradressen an die Trusted Nodes weiterleiten. Alle Knoten unterhalb der gewählten Splitpoint-Knoten, also in Richtung der ContextNodes, verfahren nach der gleichen Vorgehensweise wie beim vollständigen Backtracking.

In Abbildung 5.4 ist das in Abbildung 5.3 gezeigte Beispiel für ein TTL-basiertes tiefenbeschränktes Backtracking dargestellt. In der Teilabbildung 5.4(a) wird die Nachricht mit einem Zähler zu den ContextNode-Empfängern geroutet. Der TREELEVEL des Backtrackings ist

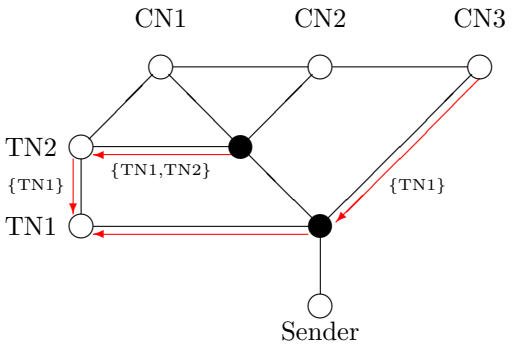
[1] „Tiefenbeschränkt“ bedeutet an dieser Stelle die Reduzierung der Tiefe des vollständigen Backtrackings



(a) Übermittlung der zeitbezogenen Nachricht



(b) TTL-basiertes tiefenbeschränktes Backtracking



(c) Gebündeltes Contextcast

Abbildung 5.4: Beispiel für ein TTL-basiertes tiefenbeschränktes Backtracking

auf den Wert zwei festgelegt und in der Abbildung an letzter Stelle in der runden Klammerung angegeben. Die ContextRouter, die in der Abbildung mit gefüllten schwarzen Kreisen dargestellt sind, stellen die Splitpoint-Knoten der zeitbezogenen Nachricht dar. Bei dem ContextRouter, dessen Kindknoten im Verteilbaum der Nachricht die ContextNodes $CN1$ und $CN2$ sind, steht der Zähler für die Tiefe des Backtrackings auf Null. Dieser Splitpoint-Knoten muss demnach die Nachricht zwischenspeichern und auf die Antworten der ContextNodes warten. Bei dem ContextNode $CN3$ kommt die Nachricht mit einem Wert des Zählers größer Null an. In diesem Fall muss der ContextNode die Nachricht nach dem Matching selbst an die Trusted Nodes weiterleiten, ohne ein Backtracking der Nachricht durchzuführen. Teilabbildung 5.4(b) zeigt das tiefenbeschränkte Backtracking der Trusted-Node-Empfängerlisten an die Splitpoint-Knoten nach einem Matching der Nachricht an den ContextNodes. In dem gewählten Fall ist das nur ein Splitpoint-Knoten für die ContextNodes $CN1$ und $CN2$. Nach einer Duplikatfilterung schickt der Splitpoint-Knoten die gebündelte Nachricht an die Empfänger $TN1$ und $TN2$, wie in Teilabbildung 5.4(c) gezeigt. Zusammen mit der Weiterleitung der Nachricht durch den ContextNode $CN3$ sind insgesamt vier Nachrichten erforderlich. Vergleicht man die Anzahl der insgesamt benötigten Nachrichten ohne Beachtung der Größen der einzelnen Nachrichten mit denen des vollständigen Backtrackings für das gleiche Szenario aus Abbildung 5.3, kommt man auf einen gleichen Nachrichtenaufwand. Die Übermittlung der gebündelten Contextcast-Nachrichten benötigt zwar beim tiefenbeschränkten Backtracking zwei Nachrichten mehr, die gleiche Anzahl an Nachrichten kann jedoch beim Backtracking eingespart werden. Aus dieser Sichtweise heraus lohnt sich in diesem Fall das tiefenbeschränkte Backtracking, da zwar ein gleicher Nachrichtenaufwand vorliegt, aber die Verzögerung der Nachrichtenübermittlung reduziert wird. Generell lohnt ein tiefenbeschränktes Backtracking gegenüber einem vollständigen Backtracking bei tiefen Verteilbäumen mit einer im Schnitt kleinen Ordnung der Splitpoint-Knoten.

Die TTL-basierte Vorgehensweise beim tiefenbeschränkten Backtracking reduziert die Tiefe des Backtrackings und damit den zusätzlichen Nachrichtenaufwand. Ein wesentlicher Nachteil dieser Methode besteht in der Ignoranz der Tiefe des Verteilbaums. Diese wurde als wesentlicher Faktor für den Aufwand des Verfahrens herauskristallisiert. Die Länge der einzelnen Pfade zwischen den Splitpoint-Knoten kann durch einen – der Nachricht angehängten – Zähler auf dem Hinweg der Nachricht zu den ContextNodes eruiert werden. Diese Information muss den Splitpoint-Knoten auf dem Rückweg der Nachrichten, also beim Backtracking, mitgeteilt werden, damit diese basierend auf der Kenntnis der Pfadlängen zum nächsten Splitpoint-Knoten eine Entscheidung über die Fortsetzung des Backtrackings treffen können. Die Pfadlängen selbst sind ausschließlich ein Indikator für den Aufwand des Backtracking-Verfahrens. Eine Entscheidung über die Fortsetzung des Verfahrens sollte auf Basis der Relation zwischen Pfadlänge und Anzahl der Kindknoten eines Splitpoint-Knoten getroffen werden. Die Information über die Anzahl der Kindknoten eines übergeordneten Splitpoint-Knotens kann auf dem Hinweg der Nachricht zu den ContextNodes von einem Splitpoint-Knoten an seine „Kinder“ übermittelt werden. Sie ist ein Indikator für die Effektivität des gebündelten Contextcast. Damit stehen den Splitpoint-Knoten alle Informationen für eine Entscheidung über einen Abbruch oder eine Fortsetzung des Backtrackings zur Verfügung. Diese Entscheidung, ob ein Backtracking von einem Level des Verteilbaums aus auf den nächst höheren Level weiter geführt wird, wird unabhängig von allen Splitpoint-Knoten durchgeführt, welche die Kindknoten des nächst höheren Splitpoint-Knoten darstellen. Ein Problem bei dieser durch die Knoten unabhängig geführten Entscheidung ist die Annahme der Knoten, dass die anderen an der Entscheidung beteiligten Knoten positiv über eine Fortsetzung entscheiden. Ist dies nicht der Fall, da beispielsweise ein Kindknoten das Backtracking abbricht und alle anderen Kindknoten eine Fortsetzung anstreben, hat sich dadurch die Effektivität des gebündelten Contextcast beim nächst höheren Splitpoint-Knoten

verringert. Die genannte Annahme ist durch die Einbeziehung der Anzahl der Kindknoten des übergeordneten Splitpoint-Knoten in die Entscheidung über die Fortsetzung begründet. Eine einheitliche Entscheidung der beteiligten Knoten lässt sich also nur treffen, wenn sich die Knoten kennen und gegenseitig über die einzelnen Pfadlängen zum nächst höheren Splitpoint-Knoten informieren können. Dies ist jedoch nicht der Fall. Da des Weiteren die Entscheidung erst auf dem Rückweg während des Backtrackings getroffen wird, müssen die zwischengespeicherten Nachrichten bereits auf dem Hinweg der Nachricht bei allen Splitpoint-Knoten auf dem Weg zu den ContextNodes gespeichert werden. Damit keine Speicherleichen entstehen, müssen die übergeordneten Splitpoint-Knoten der Knoten über bereits versandte Nachrichten informiert werden, die für die Weiterleitung der Nachricht an die Trusted Nodes ausgewählt wurden. Nach der Bestätigung des Nachrichtenversands durch die Kindknoten können die zwischengespeicherten Nachrichten verworfen werden. Der damit verbundene zusätzliche Nachrichtenaufwand und die Nachrichten, die zu einer einheitlichen Entscheidungsfindung der Splitpoint-Knoten eines Levels erforderlich sind, übertreffen den Nachrichtenaufwand des vollständigen Backtrackings zwischen zwei Level des Verteilbaums bei weitem. Aus diesem Grund werden derartige Ansätze nicht weiter verfolgt. Eine Evaluierung der Effektivität der einzelnen Verfahren, vor allem für unterschiedlich gewählte TTLs des tiefenbeschränkten Backtrackings, erfolgt in Kapitel 6 durch die Simulation.

Verteilte Kategorisierung der zeitbezogenen Nachrichten Wie in Abschnitt 4.5 bereits erläutert wurde, findet die Kategorisierung der zeitbezogenen Nachrichten bei den ContextNodes statt, die ein Matching der Nachrichten auf die einzelnen Kontexte durchführen. Die Kategorie einer Nachricht ist prinzipiell keine zeit- und auch ortsunabhängige Eigenschaft (siehe Abschnitt 4.5). Bei der Anwendung des Backtracking-Verfahrens muss eine Nachricht bereits vor der Ankunft bei den ContextNodes kategorisiert werden. Dies liegt darin begründet, dass eine historische oder hybride Nachricht beim Backtracking an verschiedenen Splitpoint-Knoten auf dem Hinweg der Nachricht zu den ContextNodes zwischengespeichert werden muss. Dies gilt für futuristische Nachrichten nicht. Kann davon ausgegangen werden, dass alle Knoten des Contextcast-Systems durch Einsatz eines Protokolls wie dem NTP über synchrone Uhren verfügen, besteht schon beim ersten Splitpoint-Knoten die Möglichkeit, die Nachricht zu kategorisieren. Für die Kategorisierung sind die lokalen Abweichungen der Zeitzonen irrelevant, da alle Zeitvergleiche basierend auf der UTC-Repräsentation[2] durchgeführt werden. Eine Nachricht kann jedoch ihre Kategorie auf dem Pfad zu den ContextNodes durch die Nachrichtenübertragungszeit wechseln. Für eine Kategorisierung der Nachrichten beim Splitpoint-Knoten ist die Kategorie der Nachricht beim ContextNode entscheidend.

Historische und hybride Nachrichten beim Splitpoint-Knoten können nicht zu futuristischen Nachrichten beim ContextNode werden, weshalb sie auf jeden Fall für das Backtracking zwischengespeichert werden. Futuristische Nachrichten beim Splitpoint-Knoten stellen in dem Sinne ein Problem dar, da sie beim ContextNode zu historischen oder hybriden Nachrichten werden können, oder dass sie sogar durch die verschiedenen ContextNodes unterschiedlich kategorisiert werden. Da die Übertragungszeiten der Nachrichten zu den einzelnen ContextNodes vorab nicht ermittelt werden kann, ist also eine genaue Kategorisierung bei den Splitpoint-Knoten, die der Kategorisierung der ContextNodes entspricht, nicht durchführbar.

Die durch den ersten Splitpoint-Knoten, auf Basis seiner lokalen Uhr und ohne Berücksichtigung von Übertragungszeiten, als futuristisch eingestufte zeitbezogene Nachrichten werden nicht durch die Splitpoint-Knoten zwischengespeichert. Damit eine wiederholte Einstufung der Nachricht als futuristisch nicht durch jeden folgenden Splitpoint-Knoten durchgeführt

[2] Zeitangabe in Zulu

werden muss und eine einheitliche Kategorisierung garantiert werden kann, wird mit der Nachricht ein boolesches Attribut mitgeführt, das sie als futuristisch kennzeichnet. Für eine als futuristisch eingestufte Nachricht, die während der Übertragung ihre Nachrichtenkategorie in eine hybride oder historische Nachricht ändert, wird also kein Backtracking verwendet. Bei einer Anwendung des Backtrackings für diese Nachrichten würden allenfalls nur historische Kontexten berücksichtigt werden, die während der Übertragungszeit vom Splitpoint-Knoten an die ContextNodes abgemeldet wurden. Unter den Voraussetzungen, dass die Übertragungszeiten allgemein relativ gering sind, dass im Netz benachbarte Knoten auch geographisch nahe beieinander liegen und dass die Mehrzahl der zeitbezogenen Nachrichten lokationsbedingt so eingeschränkt ist, dass die Splitpoint-Knoten nahe an den ContextNodes liegen, sind dies relativ wenig Kontexte. Ein Backtracking würde für diese Nachrichten nur zu einem geringen Erfolg führen. Aus diesem Grund wird das Backtracking in diesem Fall nicht angewandt.

Alle Backtracking-Verfahren betreffen ausschließlich historische und hybride Nachrichten. Das Optimierungsziel liegt vorwiegend in der Reduzierung des Nachrichtenaufwands für die Zustellung der Nachrichten. Im folgenden Abschnitt wird ein Verfahren zur Reduzierung des Nachrichtenaufwands bei der Übermittlung von futuristischen Nachrichten erschlossen. Damit sind die Optimierungen dieser Arbeit für die Zustellung von Nachrichten unterschiedlicher Kategorie im zeitbezogenen Contextcast vollständig.

5.2 Verteilte Auswertung futuristischer Nachrichten

Mit der in Kapitel 4 erläuterten Konzeption eines zeitbezogenen Contextcast ist die Lokationsangabe einer zeitbezogenen Nachricht zwingend notwendig. Diese Lokationsangabe wird in eine Liste von ContextNodes aufgelöst, an welche die Nachricht dann mittels Xcast zugestellt wird. Das Matching der zeitbezogenen Nachrichten auf die Kontexte erfolgt ausschließlich bei den adressierten ContextNodes. Für hybride und historische Nachrichten kann ein Matching nur bei der Lokation der Kontextdatenbank, in diesem Fall dem ContextNode, erfolgen. Bei futuristischen Nachrichten hingegen kann ein Matching prinzipiell an einem beliebigen Knoten des Contextcast-Systems durchgeführt werden, falls die gegenwärtig gültigen Kontexte im gesamten System propagiert werden. Dies setzt voraus, dass die Kontexte sowohl bei der Anmeldung, als auch die Abmeldungen der Kontexte selbst propagiert werden. Bei der An- und Abmeldung der Kontexte wird dann, wie beim ContextNode, ein Matching der Nachricht auf die Kontexte bei dem für das Matching gewählten Knoten durchgeführt. Der dazu notwendige Startzeitpunkt der Kontexte wird mit den Kontexten selbst propagiert. Der Abmeldezeitpunkt ist Teil der Cancel-Nachricht, die durch den ContextNode verschickt wird und für eine Löschung des betreffenden Kontextes sorgt. Bei jedem erfolgreichen Matching einer Nachricht auf einen Kontext wird die Nachricht an den ContextNode weitergeleitet, dessen ID in dem Kontext enthalten ist. Anschließend erfolgt eine Übertragung der Nachricht an den entsprechenden Benutzer bzw. Sender des Kontextes. Die Intention des Verfahrens liegt in der Reduzierung des für die Übermittlung einer futuristischen Nachricht notwendigen Nachrichtenaufwands. Eine direkte Übermittlung einer futuristischen Nachricht an alle ContextNodes des Zielgebiets ist in dem Fall ineffizient, falls ein ContextNode-Empfänger im zeitlichen Verlauf keine Kontextanmeldung durchführt, dessen Kontext zu einem erfolgreichen Matching mit der Nachricht führt. Die Übermittlung der Nachricht an solche ContextNodes ist überflüssig. Erfolgt ein Matching der Nachricht auf die Kontexte vor der Übermittlung der Nachricht an die ContextNodes, können diese überflüssigen Nachrichtenübermittlungen verhindert werden.

Exkurs: Aggregationskonzepte Aggregationskonzepte für die Kontexte eines Contextcast-Systems schränken die Propagation der Kontexte ein. Vorteile der Verfahren liegen in der schnelleren Auswertung des Matchings von Nachrichten auf Kontexte durch die ContextRouter des Systems und der Reduzierung der Routingtabelleneinträge. Diese Vorteile werden zu Lasten des Nachrichtenaufwands erzielt. Eine Aggregation von Kontexten erschwert jedoch die verteilte Auswertung futuristischer Nachrichten. Die Aggregationskonzepte sollen dennoch nicht durch das Verfahren ausgeschlossen werden. Zum besseren Verständnis der Konzepte wird im Folgenden kurz auf zwei verbreitete Verfahren zur Aggregation von Kontexten eingegangen.

Beim Covering werden Kontexte aus der Routingtabelle entfernt oder zumindest ausgelagert, welche durch andere Kontexte überdeckt werden und der gleichen Verbindungsleitung zugeordnet sind.

Definition 5.2.1 *In einem Contextcast-System wird ein Kontext $k_1 \in K$ von einem anderen Kontext $k_2 \in K$ im Rahmen des Coverings überdeckt, falls folgende Bedingung erfüllt ist:*

$$k_2 \sqsubset k_1 \Leftrightarrow \forall n \in N : Match(n, k_1) \Rightarrow Match(n, k_2).$$

Nach der Löschung bzw. Auslagerung eines durch den Kontext k_2 überdeckten Kontextes k_1 werden nach wie vor Nachrichten, die zu einem erfolgreichen Matching mit dem Kontext k_1 führen, über die richtige Verbindungsleitung eines Knotens weitergeleitet. Dies ist dadurch begründet, dass der Kontext k_2 ebenfalls mit der Nachricht übereinstimmt und der gleichen Verbindungsleitung zugeordnet ist.
Bei einem Merging werden mehrere Kontexte $k_1, k_2, \ldots, k_n$ einer Routingtabelle durch einen neu konstruierten Kontext k_m ersetzt, der die einzelnen Kontexte zusammenfasst.

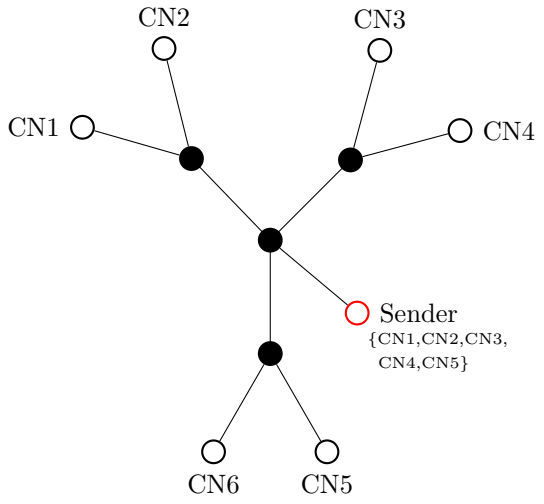
Definition 5.2.2 *In einem Contextcast-System können mehrere Kontexte $k_1, k_2, \ldots, k_n$ durch einen konstruierten Kontext k_m ersetzt werden, für den folgende Bedingung zutrifft:*

$$\forall n \in N : Match(n, k_1) \vee Match(n, k_2) \vee \ldots \vee Match(n, k_n) \Rightarrow Match(n, k_m).$$

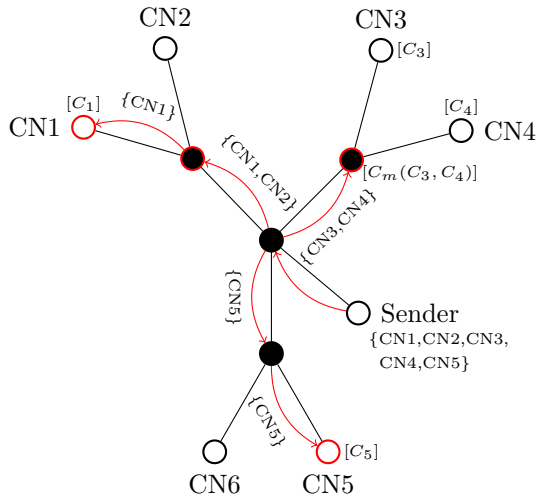
Die Aggregationskonzepte für Kontexte im Contextcast sind noch Teil der aktuellen Forschung und können daher an dieser Stelle nicht detailliert erläutert werden. Für die im Weiteren geschilderten Verfahren ist dies auch nicht von Nöten. Relevant ist nur eine Unterscheidung von regulären und zusammengefassten Kontexten, wobei zusammengefasste Kontexte mehrere reguläre Kontexte repräsentieren. Nebenbei sei noch angemerkt, dass einige mit den Aggregationskonzepten verbundene Probleme noch nicht gelöst wurden. In [MFB02] wird ein Problem beschrieben, welches beim Merging von Subscriptions in Publish/Subscribe-Systemen entsteht und auch so für das Merging von Kontexten im Contextcast gilt. Dieses ist ein Grund für den zusätzlichen Nachrichtenaufwand bei der Anwendung von Aggregationskonzepten. Die Problematik basiert darauf, dass der neu konstruierte Kontext keinem regulären ContextNode direkt zugeordnet werden kann. Bei der Propagation dieses Kontextes über Reverse Path Forwarding kann also nicht exakt nachgeprüft werden, ob der Kontext auf kürzestem Weg von den Sendern zum Empfänger eingetroffen ist. Für diese Prüfung wird der Verfasser der Nachricht benötigt. Dieser ist in diesem Fall der Knoten, bei dem der neue Kontext konstruiert wurde. Der Kontext steht jedoch für mehrere ContextNodes, die eventuell über einen kürzeren Pfad als über den Knoten erreichbar sind, der den zusammengefassten Kontext erzeugt hat. Für Nachrichten, die mit einem der zusammengefassten Kontexte übereinstimmen, werden dann möglicherweise Duplikate erzeugt. Ein Covering von Kontexten ohne eine Veränderung der regulären Kontexte existiert so im Contextcast nicht, da in jedem Kontext eindeutige Bezeichner, wie z.B. der des ContextNodes, beinhaltet sind.

Damit die verteilte Auswertung futuristischer Nachrichten durch die Verwendung von Konzepten wie der Aggregation nicht ausgeschlossen wird, kann das Matching der Nachrichten nicht an einem beliebigen Knoten durchgeführt werden. Das im Folgenden beschriebene Verfahren geht von einer verpflichtenden Angabe der Lokation für zeitbezogene Nachrichten aus und lässt sich selbst bei angewandten Konzepten wie der Aggregation von Kontexten einsetzen. Eine zentrale Frage hierbei ist, wo das Matching der Nachricht mit den Kontexten durchgeführt werden kann. Dazu sind, abhängig von der Operation des zeitlichen Attributfilters, der An- und Abmeldezeitpunkt der einzelnen Kontexte erforderlich. Werden Kontexte auf dem kürzesten Pfad von dem Sender einer Nachricht zu den Empfängern überdeckt oder zusammengefasst, so ist nur der überdeckende oder zusammengefasste Kontext von dem Sender zum Knoten hin bekannt, der die Aggregation für die Kontexte durchgeführt hat. Damit werden die An- und Abmeldungen der einzelnen aggregierten Kontexte so nur von dem Empfängern bis zu diesen Knoten weitergeleitet. Sind die An- und Abmeldezeiten dieser Kontexte für das Matching mit einer Nachricht erforderlich, kann das Matching nur auf dem Pfad zwischen den Empfängern und den Knoten durchgeführt werden, welche die Aggregation durchgeführt haben. Die An- und Abmeldezeiten sind für ein Matching mit einer Nachricht relevant, falls die Nachricht bis auf den zeitlichen Attributfilter zu dem überdeckenden oder zusammengefassten Kontext passt. Im Folgenden wird das Verfahren der verteilten Auswertung futuristischer Nachrichten ausführlich beschrieben.

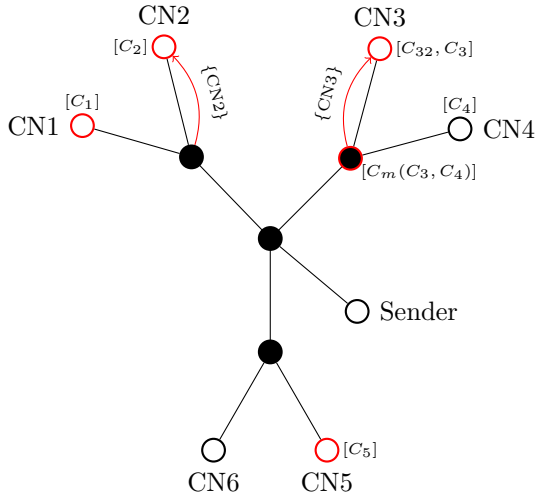
Matching und Routing Nach der Anmeldung einer futuristischen Nachricht an einem ContextNode wird sofort ein Matching auf die gegenwärtig angemeldeten Kontexte durchgeführt und bei einem erfolgreichen Matching gegebenenfalls an die ContextNode-Empfänger weitergeleitet. Dabei müssen nur Kontexte in Betracht gezogen werden, deren Schnitt der ContextNode-IDs mit der Empfängerliste der Nachricht eine nicht-leere Menge ergibt. Im Regelfall besitzen die einzelnen Kontexte genau einen ContextNode-ID. Zusammengefasste Kontexte können durchaus mehrere ContextNode-IDs oder ein Muster für diese enthalten. Beim Matching kann unter Umständen der zeitliche Attributfilter der Nachricht nicht vollständig ausgewertet werden, da notwendige Zeitpunkte des Gültigkeitsintervalls von Kontexten noch nicht festgelegt sind. Sollte aufgrund der Anmeldezeitpunkte für zusammengefasste Kontexte ein erfolgreiches Matching möglich sein, gilt es in diesem Fall als erfolgreiches Matching und die Nachricht wird zum Verfasser des Kontextes weitergeleitet. Bei diesem treffen die An- und Abmeldungen der zusammengefassten Kontexte ein, die für das vollständige Matching mit der Nachricht notwendig sind. Bei nicht aggregierten Kontexten, für die der Abmeldezeitpunkt für ein vollständiges Matching benötigt wird, erfolgt keine Weiterleitung. Eine Unterscheidungsmöglichkeit von originären und aggregierten Kontexten, zum Beispiel in Form eines zusätzlichen Attributs, wird an dieser Stelle vorausgesetzt. Das Matching von originären Kontexten und zeitbezogenen Nachrichten erfolgt wie beim ContextNode. Bei der Abmeldung der Kontexte muss, wie auch bei der Anmeldung, eine Propagation der Cancel-Nachricht erfolgen. Diese enthält gleichzeitig den für das Matching notwendigen Abmeldezeitpunkt. Falls dieser Abmeldezeitpunkt für ein Matching erforderlich ist, wird die Nachricht erst nach dem Eintreffen der Cancel-Nachricht und dem Matching an die entsprechenden ContextNode-Empfänger weitergeleitet.
Solange die Nachricht noch nicht an alle ContextNode-Empfänger weitergeleitet wurde, da z.B. einige keinen passenden Kontext angemeldet haben, muss die Nachricht zwischengespeichert werden. Sobald eine Weiterleitung an alle Empfänger erfolgt ist, oder der Löschzeitpunkt der Nachricht erreicht ist, kann die Nachricht verworfen werden. Die Zwischenspeicherung der Nachricht hat den Zweck, dass ContextNodes der Empfängerliste im weiteren Verlauf passende Kontexte an- oder abmelden können und die Nachricht an diese ausgeliefert werden kann. Bei jeder Weiterleitung der Nachricht enthält der Nachrichtenheader nur



(a) Beispielszenario



(b) Verteilung der futuristischen Nachricht



(c) Übermittlung zwischengespeicherter Nachrichten

Abbildung 5.5: Beispiel für eine verteilte Auswertung futuristischer Nachrichten

noch die ContextNode-Empfänger, die über die jeweilige Verbindung erreichbar sind, auch wenn für einzelne Empfänger noch kein passender Kontext angemeldet wurde. Die Empfänger der weitergeleiteten Nachricht werden von der Empfängerliste der Nachricht entfernt, die eventuell noch zwischengespeichert werden muss. Eine mögliche Zwischenspeicherung der Nachricht und die Erzeugung der Kopien für die verschiedenen Verbindungsleitungen erfolgt an jedem Splitpoint-Knoten bei der Weiterleitung der Nachricht. Analog zum gebündelten Contextcast historischer Nachrichten gelten ContextNode-Empfänger der Nachricht ebenfalls als Splitpoint-Knoten, bei denen die Nachricht eventuell zwischengespeichert werden muss.

Beispielszenario Ein Beispiel zur Verdeutlichung des Verfahrens der verteilten Auswertung futuristischer Nachrichten ist in Abbildung 5.5 zu sehen. Das Netz aus Teilabbildung 5.5(a) besteht aus drei ContextRouter und sieben ContextNodes, dargestellt durch gefüllte und leere Kreise. Der mit „Sender“ bezeichnete ContextNode sendet eine futuristische Nachricht an die ContextNodes CN1, CN2, CN3, CN4 und CN5. Diese wird im Gegensatz zum ursprüng-

lichen Routing nicht direkt mittels Xcast an die Empfänger übermittelt, sondern direkt beim ContextNode des Senders auf die angemeldeten Kontexte der Empfänger ausgewertet. Da die Empfänger keine passenden Kontexte, in diesem Fall überhaupt keine Kontexte, angemeldet haben, wird die Nachricht vorerst beim ContextNode gespeichert, dargestellt durch die rote Umrandung. In Teilabbildung 5.5(b) melden die ContextNodes CN1, CN3, CN4 und CN5 die Kontexte C_1, C_3, C_4 und C_5 am System an, von denen die Kontexte C_1 und C_5 zu einem erfolgreichen Matching mit der Nachricht führen. Die Kontexte C_3 und C_4 werden durch ein Merging am übergeordneten ContextRouter zu dem Kontext $C_m(C_3, C_4)$ zusammengefasst, der ebenfalls zu der Nachricht passt und wie die anderen Kontexte im Netz propagiert wird. Da nun zu der Nachricht passende Kontexte angemeldet wurden, kann der ContextNode „Sender“ die Nachricht weiter an die Empfänger übermitteln. Zuerst erfolgt eine Übermittlung der zwischengespeicherten Nachricht an den inneren ContextRouter. Da über die Verbindungsleitung zu diesem ContextRouter alle Empfänger der Nachricht erreichbar sind, kann die Nachricht beim ContextNode „Sender“ gelöscht werden. Der einfachste Fall der weiteren Verteilung der Nachricht ist die Übermittlung an den ContextNode CN5. Über die untere Verbindungsleitung des inneren ContextRouters ist nur dieser ContextNode als Empfänger der Nachricht erreichbar. Dadurch kann die Nachricht direkt an den ContextNode übertragen werden, ohne dass sie an dem Splitpoint-Knoten zu CN6 und CN5 zwischengespeichert werden muss. Die Zwischenspeicherung der Nachricht bei den adressierten ContextNodes sorgt dafür, dass ein Matching weiterer angemeldeter Kontexte mit der Nachricht im zeitlichen Verlauf bei den ContextNodes durchgeführt werden kann. Die Nachricht wird so lange bei den ContextNodes zwischengespeichert, bis der Löschzeitpunkt erreicht ist. Die Übermittlung der Nachricht durch den inneren ContextRouter an die Empfänger CN1 und CN2 erfolgt über die Verbindungsleitung in Richtung links oben (siehe Abbildung 5.5). Der ContextRouter zwischen dem inneren ContextRouter und den ContextNodes CN1 und CN2 muss die Nachricht für den ContextNode CN2 zwischenspeichern, da dieser keinen zu der Nachricht passenden Kontext angemeldet hat. Sollte der ContextNode in Zukunft einen zur Nachricht passenden Kontext anmelden, kann die Nachricht durch den ContextRouter an den ContextNode CN2 übertragen werden. Der Löschzeitpunkt der Nachricht bei den ContextRoutern muss um die maximale Übertragungszeit für Kontexte als Nachrichten zu dem ContextRouter verzögert werden, damit Kontexte, die sich noch auf dem Weg zum ContextRouter befinden, auf die Nachricht ausgewertet werden können. An den ContextNode CN1 kann die Nachricht sofort übermittelt werden. Die Übertragung der Nachricht durch den inneren ContextRouter an die ContextNodes CN3 und CN4 wird über die in der Darstellung rechts oben befindliche Verbindungsleitung abgewickelt. Da die Nachricht nun durch den inneren ContextRouter an alle Empfänger weitergeleitet wurde, muss sie durch ihn nicht länger zwischengespeichert werden. Die Kontexte C_3 und C_4 stimmen nicht mit der Nachricht überein, weshalb die Nachricht an dem ContextRouter vor den ContextNodes CN3 und CN4 zwischengespeichert werden muss. Die Speicherung an diesem ContextRouter ist notwendig, da weitere zur Nachricht passende und in der Zukunft angemeldete Kontexte durch den Kontext $C_m(C_3, C_4)$ überdeckt werden könnten. An- und Abmeldungen der Kontexte gelangen dann nur bis zu dem ContextRouter vor CN3 und CN4. Da die Zeitpunkte dieser Events für das Matching mit der Nachricht relevant sind, muss diese bei dem ContextRouter verfügbar sein. Teilabbildung 5.5(c) zeigt die Übermittlung an den ContextNode CN3, der nun zu einem späteren Zeitpunkt einen neuen Kontext C_{32} angemeldet hat, der zu der Nachricht passt. Sollte der ContextNode CN4 bis zum Löschzeitpunkt der Nachricht keinen Kontext anmelden, wird die Nachricht beim ContextRouter gelöscht. Die Nachrichtenübermittlung an diesen ContextNode ist der Aufwand, der durch das beschriebene Verfahren im Gegensatz zum Verfahren aus 4.5 eingespart wird. Der ContextNode CN2 hat in der Zwischenzeit auch einen Kontext C_2 angemeldet, der mit der Nachricht übereinstimmt. Nach

einer Übertragung der Nachricht an CN2 kann sie durch den ContextRouter vor CN2 und CN1 gelöscht werden.
Die Annäherung an die maximale Übertragungszeit einer Nachricht im Overlay kann durch das Link-State-Protokoll ermittelt werden. Diese ergibt sich jeweils für die einzelnen Knoten aus der Summe der einzelnen Leitungsverzögerungen des längsten Pfades eines Knotens. Zu diesem Wert sollte noch ein festes δ hinzu addiert werden, das unvorhergesehene Verzögerungen abdeckt und von der konkreten Implementierung gewählt werden kann.

Bewertung Gegenüber dem ursprünglichen Xcast der futuristischen Nachrichten reduziert die verteilte Auswertung futuristischer Nachrichten den Nachrichtenaufwand um die Zustellung an die ContextNode-Empfänger, die während der Gültigkeit der Nachricht keinen zur Nachricht passenden Kontext anmelden. Dieser Vorteil wird auf Kosten des zusätzlichen Speicheraufwands erzielt, der durch die Speicherung der Nachrichten bei verschiedenen Splitpoint-Knoten erzeugt wird. Ein Mehraufwand an Nachrichten im Gegensatz zur ursprünglichen Zustellung kann im Grunde nicht entstehen, da die gleichen Pfade zur Verteilung gewählt werden, nur dass der Verteilbaum an verschiedenen Stellen beschnitten wird. Generell kann der Speicheraufwand durch die ausschließliche Speicherung der Nachricht bei dem ContextNode des Senders stark verringert werden. In diesem Fall degeneriert die Zustellung der Nachricht an Erzeuger der nicht zusammengefassten Kontexte jedoch zu einer Unicast-Übertragung, was widerrum einen hohen Nachrichtenaufwand erzeugt.

Nachrichtenspektrum Ähnlich wie beim Backtracking-Verfahren für historische Nachrichten muss bei einem Matching futuristischer Nachrichten an anderen Knoten als den ContextNodes eine Kategorisierung der zeitbezogenen Nachrichten vor der Übermittlung an die ContextNodes erfolgen. Für die verteilte Auswertung futuristischer Nachrichten sind lediglich futuristische Nachrichten von Bedeutung, da hybride und historische Nachrichten bereits an die ContextNodes für das Matching auf archivierte Kontexte übermittelt werden müssen. Für hybride und historische Nachrichten stellt das Verfahren also keine Optimierung dar. Die Kategorisierung einer Nachricht kann direkt beim ContextNode des Senders ohne Berücksichtigung von Übertragungszeiten der Nachricht erfolgen. Die Kategorie muss also nicht mit denen der ContextNode-Empfänger der Nachricht übereinstimmen. Sofern die Nachricht als futuristisch eingestuft wird, setzt der ContextNode des Senders ein spezielles Attribut der Nachricht, um ein Backtracking zu verhindern. Dieses Attribut wurde bereits in Abschnitt 5.1.2 eingeführt, jedoch erst durch den ersten Splitpoint-Knoten gesetzt. Alle durch den ContextNode als futuristisch eingestuften Nachrichten sind Kandidaten für die verteilte Auswertung, auch diejenigen, die während der Übermittlung ihre Kategorie wechseln. Hintergrund ist, dass falls die Nachricht vom ContextNode des Senders als futuristisch eingestuft wird, alle folgend angemeldeten Kontexte in das Matching mit einbezogen werden können. Eine Ausnahme stellen die zusammengefassten und überdeckenden Kontexte dar. Die durch diese überdeckten Kontexte könnten zwischen der Zeit der Anmeldung der Nachricht und der Ankunft bei den Knoten, die eine Aggregation durchgeführt haben, abgemeldet werden. Bei der Ankunft der Nachricht am Knoten liegen diese dann bereits archiviert in der Kontextdatenbank der ContextNodes. In solchen Situationen, also wenn durch die Knoten der Aggregation eine als futuristisch gekennzeichnete Nachricht als hybride oder historisch Nachricht eingestuft wird, muss die Nachricht als Xcast an die übrigen Empfänger der Nachricht weitergeleitet werden. Dadurch könnten dann durchaus ContextNodes die Nachricht erhalten, die keinen zur Nachricht passenden Kontext angemeldet hatten. Die gilt nicht für alle anderen Empfänger der Nachricht, deren Kontexte unverändert propagiert wurden oder welche keine zur Nachricht passenden Kontexte angemeldet haben.
Ein grundsätzliches Problem ergibt sich, wenn ein erfolgreiches Matching mit Kontexten

durchgeführt wurde, die bei der Weiterleitung bei den folgenden Knoten auf dem Zustellungspfad nicht mehr verfügbar sind, da sie bereits abgemeldet und gelöscht wurden. In diesem Fall überschneidet sich die Löschung der Kontexte mit der Zustellung von Nachrichten, die zu dem Kontext passen. Dies ist jedoch kein spezielles Problem des zeitbezogenen Contextcast, sondern tritt bereits im ursprünglichen Contextcast auf. Bezüglich der verteilten Auswertung futuristischer Nachrichten kann in diesem Fall ebenfalls auf ein Xcast an die Empfänger der Nachricht zurückgegriffen werden. Dies hat jedoch Auswirkungen auf die Reduzierung des Nachrichtenaufwands. Dieser Fall tritt ebenfalls bei Nachrichten auf, für die bei einem Matching der Abmeldezeitpunkt der Kontexte relevant ist, da erst nach Eintreffen der Cancel-Nachricht eine Übermittlung der futuristischen Nachricht erfolgt. Diese Fälle können jedoch als seltene Ausnahme angesehen werden, wenn von schnellen Übertragungszeiten des Netzes ausgegangen werden kann.

Ein alternativer Lösungsansatz dieser Problematik, statt dem Xcast der Nachricht an die übrigen Empfänger, ist die durch einen festgelegten zeitlichen Zähler verzögerte Löschung der Kontexte bei den Knoten bei Erhalt einer Abmeldebenachrichtigung. Die zur Löschung markierten Kontexte müssen mit dem Abmeldezeitpunkt der Cancel-Nachricht versehen werden. Trifft die futuristische Nachricht bei der Zustellung auf einen Knoten, bei dem ein passender Kontext als gelöscht markiert ist, kann das Xcast an den bzw. die ContextNode-Empfänger gezielter durchgeführt werden. Weitere Empfänger, die noch keinen passenden Kontext angemeldet haben, müssen dann natürlich wieder mit der Nachricht zwischengespeichert werden. Bei nicht zusammengefassten Kontexten und Nachrichten, für die der Abmeldezeitpunkt für ein Matching irrelevant ist, ist eine direkte Übertragung der Nachricht an den im Kontext angegebenen Trusted Node denkbar. Dann darf der betroffene ContextNode-Empfänger jedoch nicht aus der Empfängerliste der verbleibenden Nachricht entfernt werden. Meldet dieser ContextNode im weiteren Verlauf keinen zur Nachricht passenden Kontext an, kann der Aufwand für die Nachrichtenübermittlung zu diesem ContextNode eingespart werden. Ob eine Übertragung an den Trusted Node, bzw. an das Mailboxsystem durchgeführt werden muss, entscheidet letztendlich das Persistenzattribut der jeweiligen Nachricht.

Der Nachrichtenaufwand für die Übermittlung an die Trusted Nodes stellt im Allgemeinen keinen Zusatzaufwand des Verfahrens dar. Diese Übermittlung muss auch bei einer direkten Übertragung der futuristischen Nachricht an alle Empfänger durchgeführt werden, falls der betroffene Benutzer die Verbindung zum System direkt nach der Abmeldung trennt. Der zusätzliche Aufwand entsteht, unter gewissen Umständen, bei der Übermittlung der Nachrichten, für die bei einem Matching der Abmeldezeitpunkt der Kontexte notwendig ist und bei Kontexten, die zur Nachricht passen, aber im Zeitraum der Übertragung der Nachricht bereits wieder abgemeldet wurden. Bei Nachrichten mit relevantem Abmeldezeitpunkt für das Matching betrifft diese Benutzer, die sich zwischen dem Zeitpunkt des Eintreffens der Cancel-Nachricht beim die Nachricht speichernden Knoten und dem Zeitpunkt vom System trennen, an dem die weitergeleitete Nachricht beim ContextNode-Empfänger eintrifft. Diese Fälle stellen jedoch, wie schon erwähnt, bei schnellen Übertragungszeiten des Netzes eine Seltenheit dar.
Die Übertragung der Nachricht an die Trusted Nodes erfolgt erst bei den ContextNodes, nachdem die Nachricht an die ContextNode-Empfänger übermittelt wurde. Nur dann können weitere Kontexte nach ihrer Abmeldung direkt beim ContextNode auf die zwischengespeicherte Nachricht ausgewertet und eventuell direkt an den Benutzer übertragen werden. Eine Überschneidung des Backtracking-Verfahrens und der verteilten Auswertung futuristischer Nachrichten wird durch die Kategorisierung beim ContextNode des Senders und dem Setzen eines Attributs für futuristische Nachrichten verhindert.

Ziellokation der Nachrichten Im Allgemeinen muss für futuristische Nachrichten keine Ziellokation angegeben werden. Diese ist hauptsächlich für das Matching mit archivierten Kontexten erforderlich. Da diese nicht im Netz propagiert werden, muss, um einen nicht zu rechtfertigenden Nachrichtenaufwand zu verhindern, eine Ziellokation für die ContextNodes als Empfänger der Nachricht angegeben werden. Futuristische Nachrichten werden jedoch nicht auf archivierte Kontexte ausgewertet. Infolgedessen ist eine Ziellokation für futuristische Nachrichten eigentlich nicht erforderlich. Für die verteilte Auswertung futuristischer Nachrichten könnten die Nachrichten demnach ohne Angabe einer Ziellokation an die Empfänger übermittelt werden. Dies betrifft natürlich nur Nachrichten, die beim ContextNode des Senders zu dem gegenwärtigen Zeitpunkt plus der maximalen Übertragungszeit für die Nachricht als futuristisch eingestuft werden. Die Übertragungszeit der Nachricht muss berücksichtigt werden, damit die zuletzt genannten Probleme mit zusammengefassten Kontexten ausgeschlossen werden können, da eine Xcast-Übertragung ohne die Angabe von Empfängern nicht durchgeführt werden kann. Alternativ bietet sich auch hier die verzögerte Löschung der Kontexte an. Bei dieser entfällt die Notwendigkeit der Einbeziehung einer maximalen Übertragungszeit.
Für die Unterstützung futuristischer Nachrichten ohne Empfängerliste bedarf es einer Änderung in der Vorgehensweise zur verteilten Auswertung futuristischer Nachrichten. Das Matching der Nachricht auf die Kontexte erfolgt somit ohne die Auswertung der Lokation oder der Prüfung der Empfänger. Zusätzlich muss die Nachricht bei jedem Knoten des Overlays zwischengespeichert werden, der bei der Übertragung der Nachricht entlang des Verteilbaums besucht wird und abgesehen von der Eingangsleitung und den Ausgangsleitungen der Nachricht mindestens eine weitere Verbindungen besitzt. Entlang dieser Verbindungen wurde die Nachricht noch nicht weiter geleitet. Es können also im zeitlichen Verlauf weitere zu der Nachricht passende Kontexte an diesen Verbindungen angemeldet werden.

Das eben beschriebene Verfahren erzeugt einen erheblichen Speicheraufwand, da die Nachricht bei der Weiterleitung an vielen Knoten zwischengespeichert werden muss. Eine ausschließliche Speicherung der Nachricht beim ContextNode des Senders stellt, wie zuvor erwähnt, ebenfalls keine zufriedenstellende Lösung dar. Für futuristische Nachrichten wird also weiterhin von einer verpflichtenden Angabe einer Ziellokation ausgegangen, über welche die ContextNode-Empfänger ermittelt werden kann. Dieser Umstand schränkt die Flexibilität des Systems nicht ein, da das Zielgebiet der Nachricht beliebig groß gewählt werden darf. Eine den Bedürfnissen des Benutzers angepasste Wahl des Zielgebiets ist allerdings empfehlenswert, da mit zunehmender Größe des Zielgebiets die Speicherbelastung des Netzes steigt. Unnötig adressierte Bereiche des Zielgebiets der Nachricht sollten demnach vermieden werden.

Der folgende Abschnitt beschäftigt sich mit einigen Aspekten der Kontextdatenbankverteilung auf mehrere Knoten des Overlays. In diesem Zusammenhang ergibt sich eine Fülle von alternativen Lösungsmöglichkeiten, die sich bezüglich verschiedener Metriken zur Beurteilung der Effizienz mehr oder minder gut eignen. Nach einer kurzen Erläuterung der Problematik des bisherigen Ansatzes zur Speicherung der Kontexte bei den jeweiligen ContextNodes wird kurz auf eine alternative Lösung eingegangen. Der Schwerpunkt liegt auf der Verdeutlichung der Herausforderungen beim Entwurf eines Verfahrens zur Verteilung der Kontextdatenbank.

5.3 Kontextdatenbankverteilung

Der bisherige Ansatz zur Archivierung der historischen Kontexte weist einige Mängel auf. Die zentrale Speicherung der Kontexte beim ContextNode, an dem sie angemeldet wurden, reicht

für den fehlerfreien Betrieb eines zeitbezogenen Contextcast völlig aus, solange die ContextNodes über ausreichende Ressourcen verfügen. Sie stellt keinen Single-Point-of-Failure dar, da das System selbst bei Ausfall einiger ContextNodes weiterhin funktionsfähig bleibt. Nachteile dieses Ansatzes ergeben sich durch die mangelnde Replikation, Persistenz und Verteilung der Daten. Nach einem Ausfall eines ContextNodes sind die bei diesem gespeicherten historischen Kontexte, selbst bei einer persistenten Speicherung durch den ContextNode, frühestens wieder im System verfügbar, wenn sich der ContextNode erneut funktionsfähig in das Overlay eingliedert. In der Zwischenzeit kann auf die archivierten Kontexte nicht zugegriffen werden, auch wenn die Erzeuger der archivierten Kontexte eigentlich im Overlay erreichbar sind. Diese Tatsache kann durch die Replikation der Daten an anderen Knoten verhindert werden. ContextNodes, die für einen großen Lokationsbereich zuständig sind, im allgemeinen eine große Anzahl von Benutzern verwalten müssen oder häufige Kontextanmeldungen verzeichnen, werden durch den von der Kontextdatenbank benötigten Speicher stark belastet. Bei mangelnder Ressourcenverfügbarkeit bezüglich Speicher und Bandbreite der Verbindungen droht infolgedessen ein Datenverlust beim ContextNode, höhere Nachrichtenübertragungszeiten bis hin zum Ausfall des Knotens. Solche Ereignisse können durch die Verteilung der zu speichernden Daten auf andere Knoten im Overlay unterbunden werden. Wichtig ist dabei, dass nicht nur die Daten in einer Art „verteilten Datenbank“ möglichst ausgewogen auf verschiedenen Knoten verteilt werden, sondern dass der Zugang zu dieser Datenbank ebenfalls über mehrere Knoten möglich ist. Nachfolgend wird ein möglicher Ansatz zur Verteilung der Kontextdatenbanken auf mehrere Knoten des Systems geschildert. Da dieser Themenbereich keinen Schwerpunkt dieser Arbeit darstellt, wird kein ausführliches Verfahren zur Verteilung der Kontextdatenbanken erläutert.

Da die Kontexte an den ContextNodes an- und abgemeldet und auch die historischen und hybriden Nachrichten zu den ContextNodes geroutet werden, bietet sich eine Speicherung der Kontexte nahe beim ContextNode an. Damit Knoten identifiziert werden können, die eine Kontextdatenbank zur Verfügung stellen, müssen sich diese durch einen Kontext bekannt geben. Dieser wird im Netz propagiert und enthält einen eindeutigen Bezeichner als Wert des *type*-Attributs, z.B. den Wert „database“ vom Typ „String“. Ein ContextNode kann nun eine Auswahl von Knoten treffen, bei denen er seine historischen Kontexte speichern möchte. Diese sollten sich möglichst nahe beim ContextNode befinden, um den Nachrichtenaufwand für die Abfrage historischer Kontexte möglichst gering zu halten. Die Entfernungen zu anderen Knoten kann durch ECHO-REPLY-Nachrichten und Messungen der RTT abgeschätzt werden. Anschließend muss durch den ContextNode festgelegt werden, an welchen Knoten die entsprechenden historischen Kontexte abgespeichert werden. Der Entschluss diesbezüglich trifft gleichzeitig die Entscheidung über die Replikation der Kontexte und die Eigenschaften der Speicherlastverteilung. Für die Replikation muss die Anzahl der Replikate, die zu replizierenden Kontexte und der Speicherort für die Replikate festgelegt werden. Eine optimale Speicherlastverteilung wird durch die gleichmäßige Verteilung der Kontexte, abhängig von der Speichergröße der Kontexte und der Speicherkapazität, auf alle beteiligten Knoten erreicht. Knoten, die eine Kontextdatenbank zur Verfügung stellen, müssen somit zusätzlich die Abfrage ihrer momentanen Speicherbelastung und Speicherkapazität ermöglichen. Damit beim ContextNode keine Liste der Speicherorte für die einzelnen Kontexte verwaltet werden muss, welche im Laufe der Zeit überaus lang werden kann, sollten die Kontexte abhängig von ihren Attributwerten gruppiert und auf die Knoten verteilt werden. Eine Einteilung der Kontexte in Gruppen – im Folgenden „Indexierung“ genannt –, kann anhand von Pflichtattributen in Kontexten erfolgen. Diese Pflichtattribute definieren eine feste Menge an möglichen Werten und müssen in jedem Kontext vorhanden sein. Ein Beispiel hierfür wäre die Gruppierung von Kontexten anhand der festen Werte eines Attributs „type“, welches den Typ eines Kontextes determiniert. Die Speicherung der Kontexte bei den ContextNodes, wie

sie bisher gehandhabt wurde, stellt beispielsweise eine Indexierung der Kontexte anhand des Lokationsattributs dar. Falls nötig, also wenn sich beispielsweise die einzelnen Gruppen in ihrer Größe stark unterscheiden, können die Gruppen selbst anhand weiterer Pflichtattribute indexiert und dadurch zusätzliche Untergruppen gebildet werden. Die Gruppen können anschließend durch den jeweiligen ContextNode verschiedenen Knoten zugeteilt werden, an welche die Kontexte der Gruppen übertragen werden. Damit eine Abbildung der Kontexte verschiedener Gruppen auf die Speicherknoten auch bei einem Ausfall des ContextNodes möglich ist, wird die Abbildungsliste der Gruppen auf die zugehörigen Speicherknoten durch den ContextNode an seine Nachbarknoten verteilt. Ist der ContextNode vorübergehend nicht erreichbar, kann der Nachbar, an dem eine Nachricht eingegangen ist, die Lokationen der archivierten Kontexte ausfindig machen. Abhängig von den Attributfiltern der Nachricht wird diese dann an die jeweiligen Knoten weitergeleitet, bei denen archivierte Kontexte abgespeichert wurden, die anschließend auf die Nachricht ausgewertet werden. Die Abbildungsliste kann durch eine beim ContextNode festgelegte TTL mit beschränkter Tiefe ringförmig um den ContextNode an die Nachbarn verteilt werden. Bei einer Änderung der Struktur des Netzes durch Verbindungsabbrüche oder Ähnliches, muss eine erneute Verteilung der Abbildungslisten nach einer Durchführung des Link-State-Protokolls erfolgen.

Der hier beschriebene Ansatz zur Verteilung der Kontextdatenbanken verteilt die historischen Kontexte als Daten bewusst auf Ebene des Overlays. Prinzipiell kann die Kontextdatenbank beim ContextNode auch als verteilte Datenbank auf einer Schicht unter dem Overlay realisiert werden. Der Nachteil dieser Methode liegt darin, dass nur ein zentraler Zugriff auf die Daten über einen ContextNode möglich ist und dadurch die zuvor genannten Schwachstellen entstehen. Ein verteilter Zugriff auf die Daten kann nur auf Overlay-Ebene realisiert werden.

Durch das gebündelte Contextcast historischer Nachrichten unter Einsatz des Backtrackings, der Duplikatvermeidung an unterschiedlichen Knoten des Contextcast und der verteilten Auswertung futuristischer Nachrichten wird der Nachrichtenaufwand des Contextcast für die Übermittlung zeitbezogener Nachrichten als auch die Belastung mobiler Clients reduziert. Das übergeordnete Ziel dieser Optimierungen ist die verbesserte Effizienz und damit Skalierbarkeit und Benutzerakzeptanz des Systems. Gestaltungsspielräume für die Verfahren der Kontextdatenbankverteilung, die ebenfalls kurz geschildert wurden, ermöglichen weiterhin eine Steigerung der Persistenz gespeicherter Kontexte und einen Lastausgleich für den Speicher der Knoten. Um die Effizienz der Verfahren nach den genannten Metriken – Nachrichtenaufwand und Nachrichtenverzögerung – zumindest abschätzen zu können, werden diese im folgenden Kapitel durch eine prototypische Implementierung des zeitbezogenen Contextcast unter verschiedenen Simulationsbedingungen evaluiert.

6 Evaluierung

Die Evaluierung der Leistungsfähigkeit des Contextcast wurde bereits durch Simulationen in [Gei07] durchgeführt. Im Folgenden wird analog dazu die Effizienz eines zeitbezogenen Contextcast durch Simulation analysiert. Für die Simulationen wurde ein Prototyp des zeitbezogenen Contextcast für den Peer-to-Peer Simulator *PeerSim* [JMJV06] implementiert. Anhand verschiedener Simulationsläufe und der resultierenden Messwerte werden die zuvor vorgestellten unterschiedlichen Verfahren für die Zustellung zeitbezogener Nachrichten bezüglich ihrer Leistungsfähigkeit und ihres Aufwands beurteilt. Abschnitt 6.1 liefert eine kurze Beschreibung des verwendeten Simulators PeerSim. In Abschnitt 6.2 sind die unterschiedlichen Komponenten des Prototyps beschrieben. Anschließend wird in Abschnitt 6.3 der Aufbau und die verwendeten Parameterwerte für die einzelnen Simulationsläufe erläutert. Nach der graphischen Darstellung der Resultate und deren Interpretation in Abschnitt 6.4 folgt schließlich in Abschnitt 6.5 das Fazit.

6.1 Simulator

PeerSim ist ein in der Sprache Java programmierter, komponentenbasierter Simulator. Er wurde speziell für die Simulation hochskalierbarer und dynamischer Peer-to-Peer-Systeme mit bis zu mehreren Millionen von Knoten entwickelt. Dazu stehen den Entwicklern der Prototypen zwei unterschiedliche Funktionseinheiten zur Verfügung: eine zyklenbasierte und eine ereignisbasierte Einheit. Einfacher und daher auch effizienter ist die zyklenbasierte Einheit. Sie abstrahiert von den Details des unter dem Prototypen liegenden Transportprotokolls und erlaubt weiterhin keine parallele Ausführung. Da Details wie Nachrichtenverluste und -verzögerungen auf Transportschicht in der Evaluierung der in dieser Arbeit erläuterten Verfahren nicht betrachtet werden, wurde die zyklenbasierte Einheit für die Simulationen verwendet. Die Effizienz der unterschiedlichen Verfahren wird anhand von Eigenschaften des Nachrichtenversands auf Overlay-Ebene analysiert, z.B. durch Zählen der übertragenen Nachrichten. Generell kann jedes Underlay zum Einsatz kommen, wobei im Transportprotokoll fehlende Anforderungen durch Ende-zu-Ende-Kommunikation umgesetzt werden müssen. Die damit verbundenen Aspekte liegen bei der Simulation von Overlay-Algorithmen nicht im Vordergrund der Evaluierung.
Bei Verwendung der zyklenbasierten Einheit kommunizieren die Knoten des Overlays direkt miteinander, indem sie Methoden anderer Objekte aufrufen. Jedem Knoten wird in einer festgelegten Reihenfolge pro Zyklus genau ein Mal die Ausführungskontrolle übergeben, indem eine spezielle Methode seiner Objektinstanzen aufgerufen wird. Nachfolgend können Berechnungen durchgeführt und Methoden von Objekten anderer Knoten aufgerufen werden. Die wichtigsten Schnittstellen für die Klassen der Objekte eines Knotens, die von PeerSim bereitgestellt werden, sind in Tabelle 6.1 aufgeführt. Eine Einführung in PeerSim inklusive einer Anleitung für die Entwicklung eigener Prototypen ist in [Jes05] gegeben.

Alle Objekte und Parameter einer Simulation werden aus einer Konfigurationsdatei zu Beginn jeder Simulation ausgelesen. Für jedes dort definierte Protokoll wird jeweils eine Instanz pro Knoten erzeugt. Eine Simulation beginnt mit dem Erzeugen der vorgegebenen Anzahl an Knoten und Protokollinstanzen. Anschließend werden alle Instanzen von Klassen der Control-Schnittstelle aufgerufen, die in der Konfiguration als Initialisierer ausgewiesen wurden. Diese legen beispielsweise die Verbindungen zwischen den Knoten fest und definieren damit die Topologie der Simulation. Anschließend erfolgt in jedem Zyklus der einmalige Aufruf der Control-Instanzen und der Protokollinstanzen eines jeden Knotens in genannter Reihenfolge. Die Reihenfolge der Aufrufe einzelner Control- und Protokollinstanzen innerhalb

Node	Jeder Knoten besteht aus mehreren Protokollen. Diese Schnittstelle bietet Zugriff auf die unterschiedlichen Protokolle und weist jedem Knoten eine eindeutigen ID zu.
CDProtocol	Diese Schnittstelle muss von jedem Protokoll einer zyklenbasierten Simulation implementiert werden. Es definiert eine Methode, die bei jedem Zyklus von dem Simulator aufgerufen wird.
Linkable	Über Protokolle dieser Schnittstelle wird aufgrund von Nachbarschaftsinformationen die Overlay-Topologie aufgebaut. Sie bieten somit Zugriff auf die Protokolle anderer Knoten des Overlays.
Control	Instanzen von Klassen dieser Schnittstelle werden zu bestimmten Zeitpunkten der Simulation aufgerufen. Sie sind keinen Knoten zugeordnet und dienen hauptsächlich der Modifikation und Überwachung der Simulation.

Tabelle 6.1: PeerSim-Schnittstellen der zyklenbasierten Einheit

eines Zyklus kann in der Konfiguration geändert werden, falls keine alphabetische Reihenfolge gewünscht ist. Zusätzlich kann die Ausführung einzelner Control-Instanzen für bestimmte Zyklen festgelegt werden. Die Anzahl der insgesamt durchgeführten Zyklen bei der Simulation wird ebenfalls durch die Konfiguration diktiert. Ein Abbruch der Simulation kann ferner durch eine Control-Instanz veranlasst werden.

6.2 Komponenten

Für die durchgeführten Simulationen wurden folgende Komponenten entwickelt, die kurz erläutert werden:

- Protokolle:
  - RTProtocol
  - InetCoordinates
  - TempMsgRouting
- Control-Klassen:
  - SendFutMsg
  - PrintMsgsStats
  - AbortSimulationRun
- Initialisierer:
  - InetInitializer
  - WireInetTopology
  - ComponentDistribution
  - SendHistMsg.

Die Klasse *RTProtocol* implementiert die Schnittstellen Protocol und Linkable. Instanzen dieser Klasse speichern für jeden Knoten eine Routingtabelle, die Nachbarn des Knotens und die über die einzelnen Verbindungen erreichbaren Knoten. Mit diesem Protokoll wird

die Topologie des Overlays festgelegt und auf die Informationen zum Routing von Nachrichten zugegriffen.
Die Knoten des Netzes werden auf ein zweidimensionales Koordinatensystem platziert. *InetCoordinates* implementiert die Schnittstelle Protocol und speichert die Koordinaten der einzelnen Knoten.
Die Klasse *TempMsgRouting* ist die Kernkomponente des Prototyps für ein zeitbezogenes Contextcast. Sie implementiert die Schnittstelle CDProtocol und ist hauptsächlich für das Routing der unterschiedlichen zeitbezogenen Nachrichten zuständig. Generell besitzt jeder Knoten bzw. Peer in PeerSim die gleiche Funktionalität. Um eine Unterscheidung zwischen Trusted Node, normalem ContextRouter und ContextNode herstellen zu können, beinhaltet die Klasse TempMsgRouting boolesche Attribute, die durch Control-Instanzen bei der Initialisierung gesetzt werden können. Abhängig von den Werten dieser Attribute besitzt ein Knoten unterschiedliche Routingfunktionalitäten.

Die Control-Klasse *SendFutMsg* implementiert ebenfalls die Schnittstelle CDProtocol und wird daher in jedem Zyklus ausgeführt. Im ersten Zyklus werden die zu evaluierenden futuristischen Nachrichten an bestimmten ContextNodes des Netzes angemeldet, indem die betreffende Methode des Protokolls TempMsgRouting des jeweiligen ContextNodes mit der Nachricht als Parameter aufgerufen wird. In den weiteren Zyklen wird vereinfachend jedem Knoten des Netzes direkt mitgeteilt, welche Empfänger der Nachrichten passende Kontexte angemeldet haben. Die Propagation der Kontexte im Netz und ein detailliertes Matching der Kontexte auf die futuristischen Nachrichten ist für die Evaluierung der Performanz der Algorithmen durch eine Gegenüberstellung nicht von Bedeutung.
Die Klasse *PrintMsgsStats* dient lediglich der Ausgabe von Simulationsergebnissen und wird daher nicht näher erläutert.
AbortSimulationRun prüft in jedem Zyklus, ob sich noch zu bearbeitende Nachrichten in den Warteschlangen der Knoten befinden. Sobald alle Nachrichten zugestellt wurden oder die Lebenszeiten abgelaufen sind, wird der aktuelle Simulationslauf abgeschlossen.

Die oben aufgeführten Initialisierer dienen dem Simulationsaufbau. Die Komponenten *InetInitializer* und InetCoordinates wurden ohne Änderungen aus dem Paket *example.hot* von PeerSim v1.0.3 übernommen. InetInitializer platziert den ersten Knoten in der Liste der verfügbaren Knoten als Wurzelknoten in der Mitte des Koordinatensystems. Alle anderen Knoten werden zufällig auf den Koordinatenraum verteilt. Weitere Informationen zum Aufbau einer statischen Topologie unter PeerSim und dem genannten Paket können der Literatur [Jes06] entnommen werden.
Die in den Simulationen verwendete Klasse *WireInetTopology* ähnelt der in [Jes06] aufgeführten gleichnamigen Klasse. Sie stellt die Verbindungen zwischen den Knoten des Systems her, die in dem Protokoll RTProtocol gespeichert sind. Eine zentrale Anforderung an den Simulationsaufbau war die Verwendung einer Topologie, die der Struktur des Internets ähnelt. Aus diesem Grund wurde für die Knotengrade eine Power-Law-Verteilung gewählt. Diese kommt konstruktiv zu Stande, indem jeweils ein zufällig gewählter Knoten, der noch keine Verbindung zum Graphen besitzt, mit dem Knoten verbunden wird, zu welchem eine minimale euklidische Distanz besteht und der Weg zum Wurzelknoten gleichzeitig minimal ist. Dies führt zu der mit α gewichteten Summe $w(x_i) + \alpha * dist(x_i, x_j)$, die für den Partner der Verbindung unter allen möglichen Knoten einer Verbindung minimal ist. Die Funktion $w(x)$ steht für die Hops des Knotens x zum Wurzelknoten, $dist(x, y)$ für die euklidische Distanz zwischen zwei Knoten. Weiterführende Informationen zu diesem Ansatz und speziell zur Wahl des Wertes für α können in [FKP02] nachgelesen werden. Das Resultat ist eine Baum-Topologie mit dem Wurzelknoten als Wurzel. Die Grade der Knoten nehmen mit zunehmendem Level polynomiell ab, so dass viele Knoten einen kleinen Grad und nur wenig Knoten einen hohen Grad aufweisen.

Nachdem die Verbindungen hergestellt wurden, müssen die Routingtabellen der Knoten gefüllt werden. Unter dem Umstand, dass die Topologie einen Baum darstellt und daher keine Zyklen besitzt, muss kein Dijkstra-Algorithmus ausgeführt werden. Es reicht völlig, den Baum von den Blattknoten aus zum Wurzelknoten abzulaufen und die dabei besuchten Knoten den Verbindungen in Richtung der Blattknoten zuzuordnen. Den Verbindungen baumaufwärts, die bei diesem Verfahren nicht berücksichtigt werden, wird die Restmenge an Knoten zugeordnet, die sich im System befinden und nicht bereits anderen Verbindungen des jeweiligen Knotens zugeordnet wurden.
Der Initialisierer *ComponentDistribution* weist den Knoten im System verschiedene Rollen zu. Diese Klasse entscheidet also darüber, welcher Knoten als normaler ContextRouter, ContextNode und/oder Trusted Node fungiert.
SendHistMsg ist das Pendant zu SendFutMsg: Nach der Auswahl bestimmter ContextNodes werden bei diesen die zu simulierenden historischen Nachrichten abgesetzt. Im Gegensatz zu SendFutMsg wird SendHistMsg als Initialisierer für einen Simulationslauf nur ein einziges Mal aufgerufen. Dies liegt darin begründet, dass die Kontexte für historische Nachrichten sofort ausgewertet werden können. Die zu den Nachrichten passenden Kontexte werden der Einfachheit halber ebenfalls durch SendHistMsg auf die Empfänger verteilt.

6.3 Simulationsaufbau und -ablauf

Für die einzelnen Simulationsläufe werden jeweils 10.000 Knoten in das Overlay aufgenommen. Die Knoten und die Transportschicht unterhalb des Overlays werden nicht betrachtet, da sie für die Analyse der hier im Fokus stehenden Eigenschaften der Algorithmen des Overlays nicht von Bedeutung sind. Der Koordinatenraum ist zweidimensional und beschränkt sich für beide Dimensionen auf die reellen Zahlen im Intervall $[0, 1[$. Der erste Knoten der Liste verfügbarer Knoten wird als Wurzelknoten in der Mitte des Koordinatensystems, alle anderen Knoten auf zufällig gewählte Punkte des Koordinatenraums platziert. Anschließend können die Verbindungen und Routingtabellen der einzelnen Knoten aufgebaut werden. Für die Wahl der Verbindungen wird die bereits im letzten Abschnitt erwähnte Formel mit dem Parameterwert $\alpha = 20$ gewählt, so dass eine Power-Law-Verteilung der Knotengrade entsteht. Da der Wert für α deutlich unter dem Wert $\sqrt{10000} = 100$ liegt, bilden sich deutliche Cluster von Knoten. Nachdem die Topologie und Routingtabellen des Netzes soweit erzeugt sind, werden 7.000 von den insgesamt 10.000 Knoten als ContextNodes ausgewählt. Da sich die ContextNodes als Zugangspunkte in einem realen System eher am Rand des Systems befinden, wurden die Knoten nach aufsteigendem Grad sortiert. Die Verteilung der ContextNodes auf diese Liste von Knoten erfolgt im Anschluss durch Anwendung einer 80-20-Zipf-Verteilung. Dabei liegen 80% der zufällig gewählten Knoten in den ersten 20% der Liste. Die Wahrscheinlichkeit für die Wahl eines Knotens mit Listenindex x liegt nach der Zipf-Verteilung bei $P(X = x) = \frac{x^{-s}}{\xi(s)}$. Der Bezeichner $\xi(s)$ steht für die riemannsche Zetafunktion. Der Parameter s für die Verteilung liegt bei der hier verwendeten 80-20-Zipf-Verteilung mit 10.000 möglichen Werten für x bei ca. 0,95. Die Auswahl von zufälligen Knoten der Liste für die Verteilung der ContextNodes endet mit der erfolgreichen Wahl von 7.000 unterschiedlichen Knoten.

Gegenstand der Evaluierung sind die Verfahren zur Übermittlung der zeitbezogenen Nachrichten. Verfahren für historische und futuristische Nachrichten werden getrennt voneinander simuliert, untersucht und analysiert. Dabei werden die verschiedenen austauschbaren Verfahren gegenübergestellt und anhand verschiedener Metriken aufgrund der Simulationsergebnisse bezüglich Aufwand und Effizienz verglichen. Damit die Verfahren vergleichbar sind, müssen die Nachrichten, die durch unterschiedliche Algorithmen zugestellt werden, bestimmte Voraussetzungen erfüllen. Die Auswahl der Nachrichtenempfänger erfolgt durch

die zufällige Positionierung eines Quadrats festgelegter Breite im Koordinatenraum. Alle ContextNodes, die sich in diesem Quadrat befinden, werden als Empfänger der Nachrichten ausgewählt. Der Sender der Nachricht wird zufällig aus einer List verfügbarer ContextNodes gewählt. Falls nun Nachrichten mit gleicher Quadratbreite für die Auswahl der Empfänger durch unterschiedliche Algorithmen zugestellt werden, ist die Liste der Empfänger und der Senderknoten der Nachrichten gleich. Die Übermittlung einer Nachricht von einem Knoten zu einem Nachbarknoten wird in der Simulation innerhalb eines Zyklus vollzogen.

Backtracking historischer Nachrichten Im Rahmen der Evaluierung des Backtrackings historischer Nachrichten zur Reduzierung der Nachrichtenlast des Netzes wurden jeweils vier verschiedene Algorithmen zur Übermittlung der historischen Nachrichten simuliert: Das vollständige Backtracking, das tiefenbeschränkte Backtracking mit den TREELEVEL 1, 2 und 3 und die originäre Zustellung durch die direkte Weiterleitung der Nachrichten an die Trusted Nodes. Simuliert wird jeweils nur das Routing der historischen Nachrichten vom Senderknoten zu den ContextNodes der Empfängerliste einer Nachricht, das anschließende Backtracking der Trusted-Node-Empfänger als Resultat des Matchings und die anschließende Weiterleitung der Nachricht an die Trusted Nodes.

Insgesamt 300 Knoten wird die Rolle des Trusted Nodes zugeteilt. Die Wahl der Trusted Nodes erfolgt zufällig und folgt einer Gleichverteilung. Zusätzlich zur gleichen Liste der Empfänger und dem gleichen Senderknoten enthalten die Nachrichten exakt die gleichen Filter, d.h. die Trusted Nodes als weitere Empfänger der Nachrichten sind für die unterschiedlichen Nachrichten ebenfalls gleich. Für die Analyse der Verfahren sind nur Kontexte bei den ContextNode-Empfängern von Interesse, die zu den Nachrichten passen und beim gleichen ContextNode auf unterschiedliche Trusted Nodes als Empfänger verweisen. Unterschiedliche Kontexte mit gleichem Trusted Node werden beim ContextNode nach dem Matching im Rahmen der Duplikatvermeidung verworfen.

Vereinfachend werden bei der Simulation zu jeder einzelnen Nachricht passende Kontexte mit unterschiedlichen Trusted Nodes als Empfänger konstruiert. Dies erfolgt für jede Nachricht und jeden ContextNode als Empfänger einer Nachricht, wobei die Nachrichten der unterschiedlichen Zustellungsverfahren mit gleichen Empfängern an die selben Trusted Nodes gerichtet werden. Die Anzahl der Kontexte (und damit Trusted Nodes), die zu einer Nachricht passen und bei einem ContextNode abgespeichert werden, wird gleichverteilt auf dem Wertebereich 0..300 gewählt. Die Wahl der einzelnen Trusted Nodes für die Nachrichten mit unterschiedlicher Empfängerliste erfolgt für jeden ContextNode durch eine 80-20-Zipf-Verteilung auf eine unsortierte Liste der verfügbaren Trusted Nodes. Wie bei der Wahl der ContextNodes des Netzes bricht die Verteilung erst ab, sobald die Trusted Nodes der gewünschten Anzahl ausgewählt wurden. Die Verteilung kommt der Vermutung nach, dass in einem zeitbezogenen Contextcast bestimmte Trusted Nodes weitaus populärer sind als andere und daher eine Zipf-Verteilung der Benutzer auf die Trusted Nodes nahe liegt.

Verteilte Auswertung futuristischer Nachrichten Bei der Simulation der verteilten Auswertung futuristischer Nachrichten werden, im Gegensatz zu der Simulation der Zustellung historischer Nachrichten, die zur Nachricht passenden Kontexte erst im Laufe der Simulation erzeugt. Dazu wird in jedem Zyklus ein Empfänger mit 10% Wahrscheinlickeit für eine Nachricht freigeschalten. Erst nach der Freischaltung wird die eigentliche Nachricht zu diesem Empfänger übermittelt. Die Freischaltung ist in diesem Zusammenhang gleichbedeutend mit der Anmeldung eines zur Nachricht passenden Kontextes durch den Empfängerknoten. Benutzer, die einen passenden Kontext anmelden und anschließend abmelden, bevor die Nachricht zugestellt werden konnte, oder die direkt nach der Abmeldung nicht mehr erreichbar sind und an welche die Nachricht eigentlich über einen Trusted Node zugestellt werden

müsste, werden durch diese Simulation nicht betrachtet. Da davon auszugehen ist, dass diese aus Benutzersicht bereits eher seltenen Fälle bei heutigen Übertragungsgeschwindigkeiten doch sehr selten auftreten und sich daher nur marginal auf den Aufwand des Verfahrens auswirken, wurden sie absichtlich aus der Simulation ausgeschlossen.

Um die verteilte Auswertung futuristischer Nachrichten nicht nur bei unterschiedlich hoher Anzahl an Nachrichtenempfängern, sondern auch bei unterschiedlich hoher Anzahl freigeschalteter Empfänger ($\triangleq$ Empfänger mit zur Nachricht passendem angemeldeten Kontext) analysieren zu können, werden mehrere Nachrichten mit unterschiedlicher Anzahl an freigeschalteten Empfängern erzeugt. Beträgt diese Grenze beispielsweise $\frac{1}{2}$, werden bis zum Ende der Lebenszeit der Nachricht die Hälfte der Empfänger freigeschaltet.

6.4 Resultat

Nachfolgend sind die Simulationsergebnisse in Diagrammen dargestellt. Die genauen Messdaten sind in Anhang A aufgeführt. Die einzelnen Werte sind gemittelte Werte über insgesamt zehn separat durchgeführte Simulationsläufe. Die Abweichungen der Werte vom Durchschnitt sind über die minimalen und maximalen Werte abgebildet. In den drei- und vierdimensionalen Abbildungen sind nur die Durchschnittswerte angegeben. Die minimalen und maximalen Abweichungen sind in den zweidimensionalen Diagrammen durch waagerechte Striche dargestellt. Die Klasse der Verfahren für die Übermittlung historischer und futuristischer Nachrichten wurden in getrennten Simulationsläufen ausgeführt. Für alle zu evaluierenden Verfahren wurden Nachrichten mit der Quadratbreite von 0,05 bis 0,25 für den Empfängerraum in Schritten von 0,05 erzeugt.

6.4.1 Backtracking historischer Nachrichten

Für die Übermittlung der historischen Nachrichten wurden das originäre Zustellungsverfahren, das tiefenbeschränkte Backtracking und das vollständige Backtracking simuliert. Beim tiefenbeschränkten Backtracking liegt die Hoffnung darin, dass es die Verzögerung der Nachrichtenzustellung mit einem geringem Mehraufwand an Nachrichten bei der Zustellung historischer Nachrichten und einem reduzierten Backtracking-Aufwand im Gegensatz zum vollständigen Backtracking mindert. Im Rahmen der Simulationen wurden die TREELEVEL 1, 2 und 3 für das tiefenbeschränkte Backtracking festgelegt.

Abbildung 6.1 zeigt die Anzahl der ContextNode-Empfänger, die durch ein Quadrat mit entlang der x-Achse abgebildeten Breiten ausgewählt wurden. Da für alle Zustellungsverfahren und damit Nachrichten mit der gleichen Quadratbreite die gleichen Empfänger ausgewählt wurden, ist in der Abbildung nur ein Schaubild aufgezeigt. Die Messergebnisse bestätigen die Erwartung, dass sich die Anzahl der ContextNode-Empfänger c nach der Formel $c = 7.000 * (b^2)$ linear mit steigendem Flächeninhalt des Quadrats bei einer Gleichverteilung der Knoten auf den Koordinatenraum erhöht. Aus Abbildung 6.1 ist demnach eine quadratische Steigung der Empfänger in Bezug zur Quadratbreite b zu erkennen.

Die in Abbildung 6.2 dargestellte maximale Backtracking-Tiefe für die einzelnen Verfahren der Zustellung bezeichnet die maximale Tiefe des Backtracking-Baumes, wobei nur potentielle Splitpoint-Knoten als Knoten des Baumes gezählt wurden. Dies sind die Knoten, an denen bei Bedarf eine Zwischenspeicherung der historischen Nachrichten durchgeführt wird. Für das tiefenbeschränkte Backtracking werden genau diese Knoten übersprungen, bis der gewünschte TREELEVEL erreicht ist. Anhand der maximalen Backtracking-Tiefe kann also festgestellt werden, mit welchem TREELEVEL ein tiefenbeschränktes Backtracking noch Sinn macht, bevor es dem originären Verfahren ohne Backtracking gleicht. Wie bezüglich der Backtracking-Tiefe ebenfalls zu erwarten war, nimmt die maximale Tiefe mit zunehmender

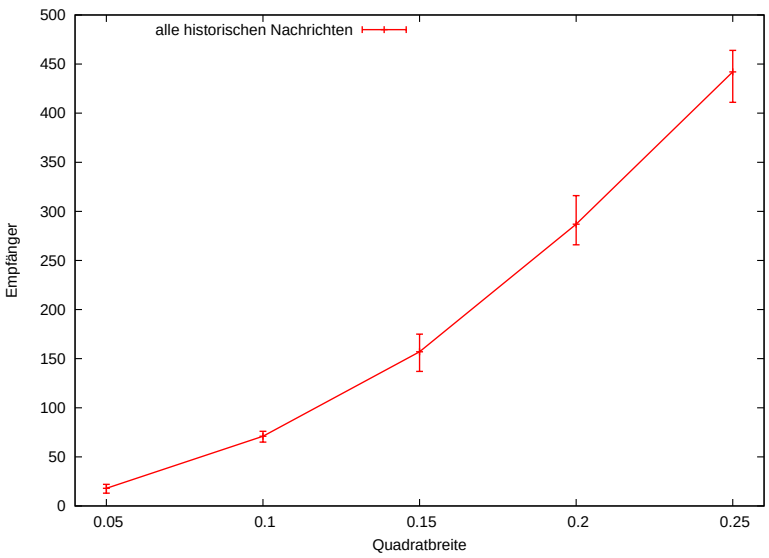


Abbildung 6.1: ContextNode-Empfänger der historischen Nachrichten

Erhöhung des TREELEVELs für das tiefenbeschränkte Backtracking in gleicher Schrittweite ab. Die Abweichung von dieser Erwartung bei einer Quadratbreite von 0,1 zwischen dem tiefenbeschränkten Backtracking mit TREELEVEL 2 und 3 entsteht ausschließlich durch die Bildung des Durchschnittwertes aus zehn Simulationsläufen und der anschließenden Rundung. Die maximalen Tiefen zwischen den Nachrichten unterschiedlicher Quadratbreite für die Empfänger kann so nicht direkt verglichen werden, da in diesem Fall das Quadrat unterschiedlich positioniert und ein anderer Senderknoten gewählt wurde. Es kann jedoch statistisch davon ausgegangen werden, dass mit einer zunehmenden Anzahl an Empfängern die maximale Tiefe des Backtrackings steigt. Die maximale Tiefe ist gleichzeitig ausschlaggebend für die Verzögerung bei der Zustellung der Nachrichten, da vor der Weiterleitung der Nachrichten an die Trusted Nodes alle Ergebnisse des Backtrackings abgewartet werden müssen. Dies sind insbesondere auch die Ergebnisse, die entlang des Pfades der maximalen Tiefe verschickt werden. Leider hängt die maximale Tiefe von verschiedenen Faktoren, wie z.B. der Grade der Knoten im Baum und der Struktur des Baumes insgesamt, ab, welche im voraus nicht bekannt sind. Sind bestimmte Einschränkungen bezüglich der Höhe der Verzögerungen der Nachrichten erwünscht, kann der festzulegende TREELEVEL für das tiefenbeschränkte Backtracking nur annäherungsweise durch statistische Untersuchungen ermittelt werden.

Der Aufwand des Backtracking-Verfahrens in Nachrichten ist in Abbildung 6.3 für die verschiedenen Verfahren dargestellt, die ein Backtracking durchführen. Wie aus der Abbildung schnell ersichtlich ist, nimmt der Nachrichtenaufwand für das Backtracking mit zunehmendem Backtracking-TREELEVEL ab. Das vollständige Backtracking benötigt den größten Nachrichtenaufwand. Die Anzahl an Knoten eines Levels nimmt für Bäume, abhängig von den Graden der Knoten, in aller Regel exponentiell zu. Das gleiche Wachstum gilt für die Verbindungen zwischen den Knoten der unterschiedlichen, aneinander angrenzenden Level. Werden nun von der Wurzel des Backtracking-Baums aus zunehmende Level aus dem Backtracking ausgeschlossen, nimmt die Anzahl der nicht am Backtracking beteiligten Knoten (und damit Verbindungen) exponentiell zu. Daraus resultiert die aus Abbildung 6.3 ersichtliche exponentielle Abnahme des Nachrichtenaufwands für das Backtracking mit zunehmendem Backtracking-TREELEVEL. Interessant ist, dass der Nachrichtenaufwand nur knapp über der Anzahl an ContextNode-Empfängern liegt und in etwa den gleichen Verlauf

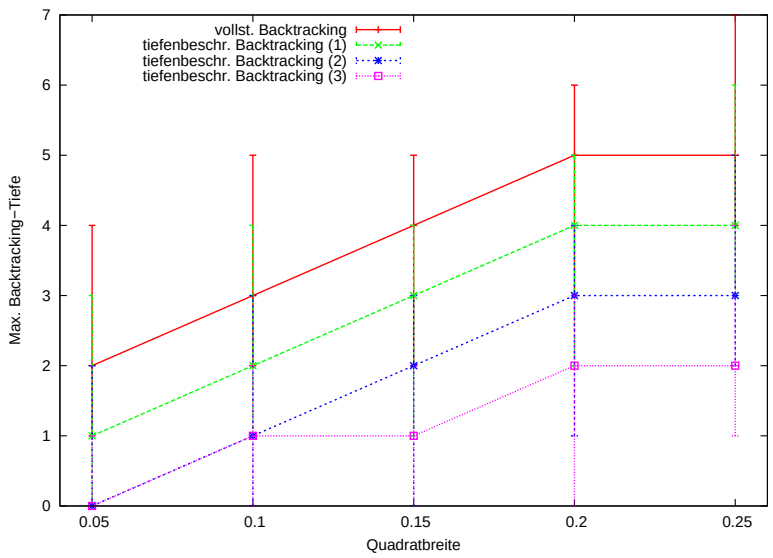


Abbildung 6.2: Maximale Backtracking-Tiefe der unterschiedlichen Backtracking-Verfahren

mit zunehmender Quadratbreite aufweist. Bei einer Quadratbreite von 0,25, was in etwa 450 ContextNode-Empfängern entspricht, können fast $\frac{3}{5}$ des Nachrichtenaufwands für das vollständige Backtracking durch ein tiefenbeschränktes Backtracking mit TREELEVEL 3 eingespart werden.

Die folgende Abbildung 6.4 zeigt die Anzahl der Bestätigungen der Trusted Nodes für die Zustellungen der historischen Nachrichten mit und ohne Backtracking. Diese Anzahl ist ein Indikator für den Nachrichtenaufwand, der durch die Zustellung von Nachrichtenduplikaten entsteht. Insgesamt sind ausschließlich 300 unterschiedliche Trusted Nodes im Netz verfügbar. Selbst wenn eine Nachricht an all diese Trusted Nodes nach dem Matching bei den ContextNode-Empfängern zugestellt werden muss, liegt es nahe, dass bei einer Anzahl von mehr als 300 Trusted-Node-Bestätigungen für eine Nachricht Duplikate bei den Context-Nodes eingegangen sind. Bei einem vollständigen Backtracking werden all diese Duplikate nach dem Backtracking beim Wurzelknoten verworfen, so dass die Anzahl der Trusted-Node-Bestätigungen exakt mit der Anzahl der tatsächlichen Trusted-Node-Empfänger der Nachricht übereinstimmt. Mit zunehmendem TREELEVEL des tiefenbeschränkten Backtrackings nimmt die Zahl der Knoten zu, bei denen die Nachricht zu den Trusted Nodes verschickt wird und an denen eine Duplikatvermeidung durchgeführt wird. Dadurch wird die Sicht über die Menge der Trusted-Node-Empfänger bei der Duplikatvermeidung immer weiter eingeschränkt. Bei dem originären Verfahren ohne Backtracking erfolgt eine Duplikatvermeidung ausschließlich bei den ContextNode-Empfängern. Dieses Verfahren weist daher die größte Anzahl an Nachrichtenduplikaten und damit Trusted-Node-Bestätigungen auf. Wie man in Abbildung 6.4 sehen kann, ist der Nachrichtenaufwand durch Duplikatzustellungen enorm: Bei einer Quadratbreite von 0,25 reicht die Skala bis zu 70.000 Bestätigungen für das Verfahren ohne Backtracking im Gegensatz zu den 300 Bestätigungen bei einem vollständigen Backtracking.

Der Nachrichtenaufwand für die Übermittlung der simulierten historischen Nachrichten vom Senderknoten bis hin zu den Trusted Nodes ist in Abbildung 6.5 skizziert. Der dargestellte Aufwand enthält wohlgemerkt nicht den Nachrichtenaufwand für das Backtracking, der in Abbildung 6.3 zu sehen ist. Wie bereits aus der Anzahl der bestätigten Nachrichten durch Trusted Nodes abgeleitet werden kann, muss für den zunehmenden TREELEVEL des

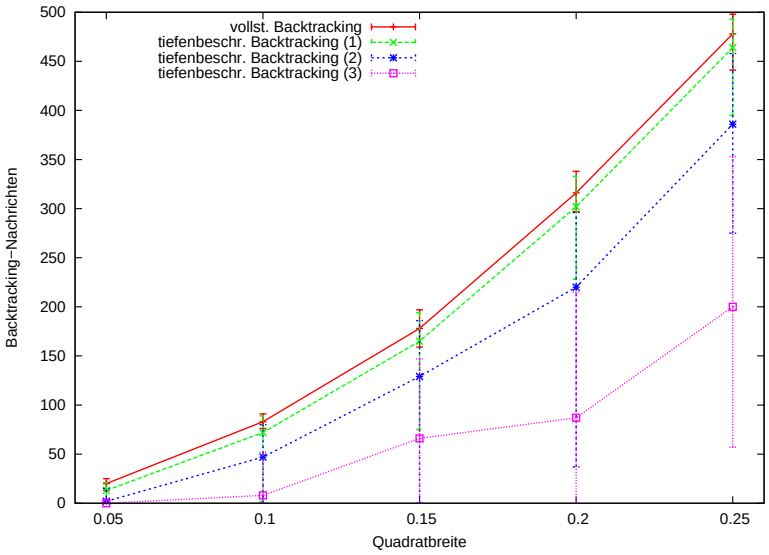


Abbildung 6.3: Nachrichtenaufwand des Backtrackings

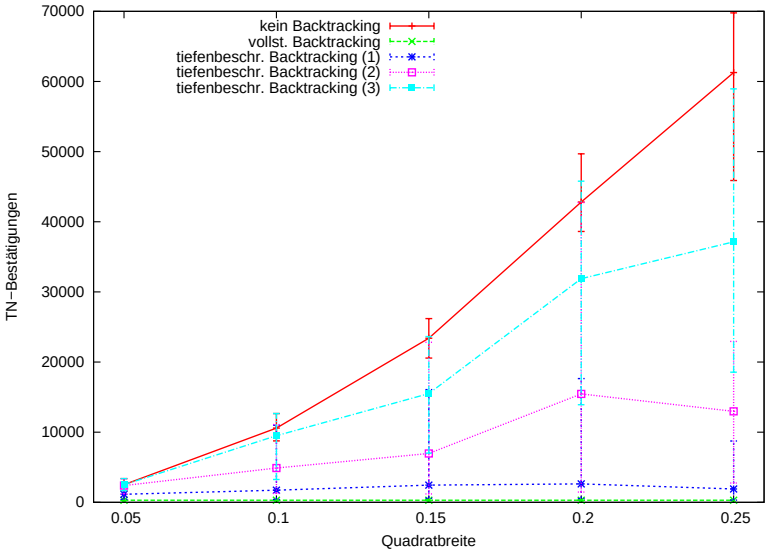


Abbildung 6.4: Nachrichtenbestätigungen durch Trusted Nodes

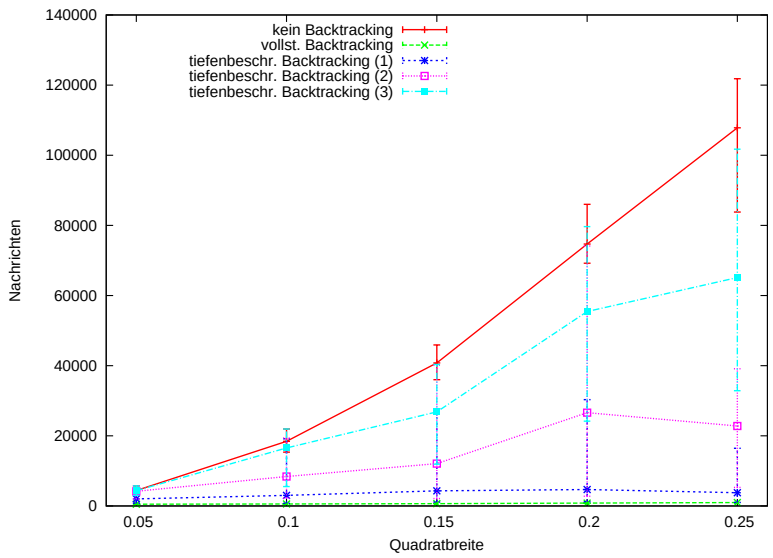


Abbildung 6.5: Nachrichtenaufwand für die Zustellung der historischen Nachrichten

tiefenbeschränkten Backtrackings und für das originäre Verfahren ein erheblicher Nachrichtenaufwand entstehen. Die Messergebnisse zeigen, dass zum Beispiel der Durchschnittswert für den Nachrichtenaufwand bei einer Quadratbreite von 0,25 beim originären Verfahren bei ca. 108.000 Nachrichten im Gegensatz zu knapp 1.000 Nachrichten für das vollständige Backtracking liegt. Dies entspricht über dem 100fachen an Nachrichten. Bei einer Einsparung von knapp 500 Nachrichten für das Backtracking bei Verwendung des originären Verfahrens lohnt das vollständige Backtracking unter dem ausschließlichen Gesichtspunkt des Nachrichtenaufwands allemal. Bezüglich der Verzögerung der Nachrichtenübermittlung muss anwendungsspezifisch abgewägt werden, ob sich der erhebliche Mehraufwand an Nachrichten für eine Reduzierung der Verzögerung rentiert. Es kann davon ausgegangen werden, dass der Nachrichtenaufwand mit jeder Erhöhung des TREELEVELs für das Backtracking exponentiell ansteigt. Die Messwerte zeigen, dass die zusätzliche Verzögerung durch das vollständige Backtracking bei der Simulation unter Verwendung einer Internet-ähnlichen Topologie und insgesamt knapp 450 ContextNode-Empfängern durchschnittlich bei fünf logischen Hops liegt. Bei den heutig herrschenden Übertragungsgeschwindigkeiten und deren Wachstum sollte diese geringe Anzahl an zusätzlichen Hops kein Problem darstellen. Weitaus wichtiger ist in diesem Zusammenhang die präzise Wahl oder zumindest die höchst mögliche Eingrenzung der Ziellokation der Nachricht auf den gewünschten Empfängerbereich, so dass keine unerwünschten Empfänger adressiert werden, die den Nachrichtenaufwand weiter erhöhen. In diesem Zusammenhang ist auch die Voraussetzung bedeutend, dass physisch nahe beieinander befindliche Knoten auch im Overlay nicht weit voneinander entfernt sind. Ist dies nicht der Fall, wird ein zusätzlicher Nachrichtenaufwand durch das frühe Aufspalten der Nachricht bei der Übermittlung an die ContextNode-Empfänger erzeugt, der wiederum den Nachrichtenaufwand für das Backtracking erhöht.

6.4.2 Verteilte Auswertung futuristischer Nachrichten

Bei der verteilten Auswertung futuristischer Daten macht man sich die Tatsache zur Verminderung des Nachrichtenaufwands zu Nutzen, dass ein Teil der adressierten Empfänger im Laufe der Lebenszeit einer Nachricht keinen passenden Kontext anmelden und die Nachrich-

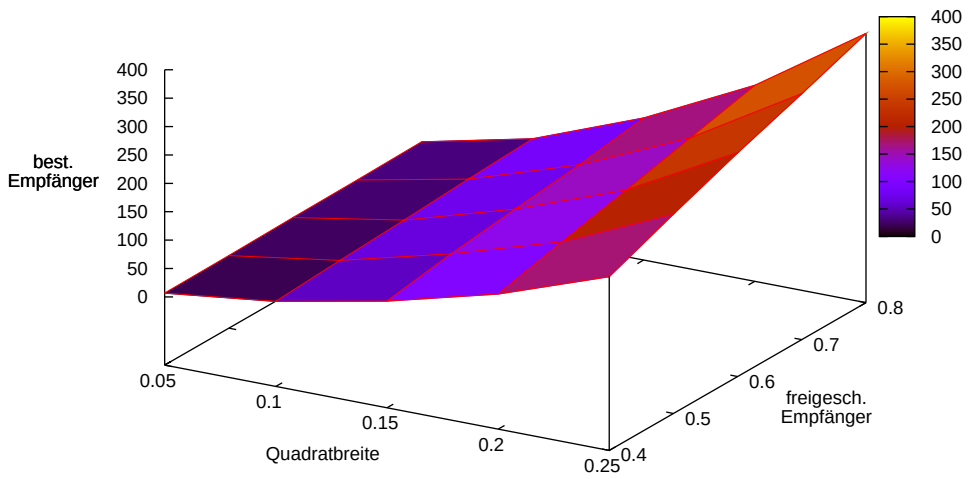


Abbildung 6.6: Bestätigte ContextNode-Empfänger der futuristischen Nachrichten

tenübermittlung zu diesen Empfängern daher unnötig ist. Diese Optimierung geschieht auf Kosten des zusätzlichen Speicheraufwands, der zur Zwischenspeicherung der futuristischen Nachrichten benötigt wird. Im schlimmsten Fall, wenn ausschließlich ein Knoten einen passenden Kontext angemeldet hat und alle anderen Knoten über separate Abzweigungen auf dem Pfad der Nachricht zu erreichen sind, liegt der Speicheraufwand in $O(n)$, wobei n für die Anzahl der Empfänger steht. Dieser zusätzliche Speicheraufwand und die Nachrichtenminderung der verteilten Auswertung werden im Folgenden anhand von Messwerten näher untersucht.

Abbildung 6.6 zeigt die Anzahl der ContextNode-Empfänger für die unterschiedlichen futuristischen Nachrichten als Mittel der zehn unabhängigen Simulationsläufe in einem dreidimensionalen Diagramm. Die dritte Dimension ist zusätzlich durch die Färbung des Schaubildes verdeutlicht. Es wurden nur die Empfänger gezählt, die während der Lebenszeit der Nachricht einen passenden Kontext angemeldet haben. Dieser Anteil der insgesamt adressierten Empfänger ist in Dezimalbruchdarstellung entlang der y-Achse aufgetragen. Die gesamte Quantität an adressierten Empfängern ähnelt denen aus Abbildung 6.1 und ist für die futuristischen Nachrichten in Abbildung 6.7 dargestellt. Die bestätigten Empfänger der beiden futuristischen Nachrichten mit 60% und 80% freigeschaltenen Empfängern stimmen in etwa mit den 60% und 80% der insgesamt adressierten Empfängern überein. Diese Tatsache ratifiziert, dass keine Duplikate der Nachrichten erzeugt und zugestellt werden.

Der in Abbildung 6.8 in einem dreidimensionalen Diagramm dargestellte Nachrichtenaufwand ähnelt dem in Abbildung 6.3 abgebildeten Nachrichtenaufwand für das vollständige Backtracking historischer Nachrichten. Dieser Sachverhalt ist darin begründet, dass die Verteilbäume beim Backtracking und der Zustellung futuristischer Nachrichten bis auf den Pfad vom Senderknoten hin zum ersten Splitpoint-Knoten gleich sind. Aus Gründen der Übersichtlichkeit sind in Abbildung 6.9 nur die Werte für die verteilte Auswertung mit 40%, 60% und 80% an freigeschalteten Empfänger im Vergleich zu der originären Zustellung dargestellt. Diese Prozentangaben können, wie aus den Schaubildern ersichtlich wird, so auch in etwa für die Berechnung des Nachrichtenaufwandes in Bezug auf den Nachrichtenaufwand der originären Verteilung verwendet werden. Genau genommen wird die Einsparung tatsächlich nicht so hoch sein, da nur gewisse Äste des Multicast-Baums bei dem Ausschluss von Empfängern aus der Zustellung gekürzt werden. Wie beim Nachrichtenaufwand für das

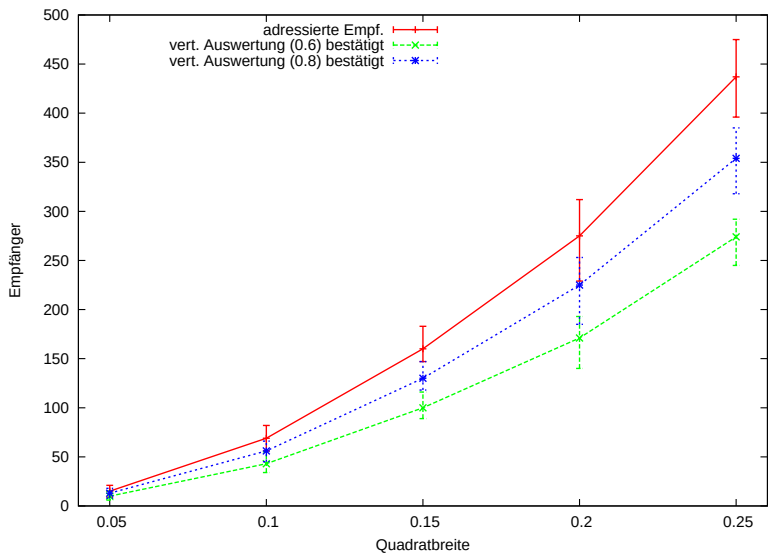


Abbildung 6.7: ContextNode-Empfänger im Vergleich

Backtracking liegt der Nachrichtenaufwand hier ebenfalls relativ knapp über der Anzahl an adressierten, bzw. freigeschalteten Empfängern und folgt in etwa der gleichen Steigung. Wie hoch die Einsparung des Nachrichtenaufwands bei der Zustellung einer futuristischen Nachricht im Vergleich zu der originären Übermittlung tatsächlich ist, hängt von den Empfängern ab, die zur Lebenszeit der Nachricht einen passenden Kontext anmelden. Da die Zahl dieser Empfänger vorab weder berechnet noch annäherungsweise abgeschätzt werden kann, bleibt die Einsparung den Nachrichtenfiltern, den Nachrichtenempfängern und dem Zufall überlassen. Fakt ist jedoch, dass der Nachrichtenaufwand der verteilten Auswertung futuristischer Nachrichten nur in sehr seltenen Fällen höher ist als der Aufwand der originären Zustellung.

Abbildung 6.10 zeigt in einer vierdimensionalen Darstellung die Durchschnittswerte der zwischengespeicherten Nachrichten im Overlay, die mit Hilfe der abgebildeten Farbskala abgelesen werden können. Anhand dieser Werte kann der Speicheraufwand des Verfahrens abgeschätzt werden. Entlang der x-Achse sind die Zyklen der Simulationsläufe angegeben. In Abbildung 6.11 ist das Diagramm aus Abbildung 6.10 in fünf dreidimensionalen Teilabbildungen dargestellt. Die zwischengespeicherten Nachrichten als dritte Dimension sind farblich gekennzeichnet. In jedem Zyklus der Simulation wurden die Replikate einer Nachricht gezählt, die sich in dem Netz befanden. Zu Beginn eines Simulationslaufes sind noch keine Empfänger einer Nachricht freigeschaltet. Im Laufe der Zeit werden dann immer mehr Empfänger freigeschaltet, so dass in Folge der Übermittlung der Nachrichten immer mehr Replikate zwischengespeichert werden. Ab einem gewissen Zeitpunkt nimmt diese Zahl wieder ab, da durch die zunehmende Freischaltung zwischengespeicherte Nachrichten entfernt werden. Dies liegt darin begründet, dass gespeicherte Nachrichten entfernt werden, sobald alle Empfänger der zwischengespeicherten Nachricht freigeschaltet wurden und die Nachricht übermittelt werden kann. Die Freischaltung erfolgt nicht sofort, sondern zunehmend mit der Zahl der simulierten Zyklen. Die Werte zwischen den angegebenen Quadratbreiten entlang der y-Achse aus Abbildung 6.10 sind nicht von Relevanz, da für diese Quadratbreiten keine Nachrichten simuliert wurden. Die Teilabbildungen aus Abbildung 6.11 zeigen daher nur die Diagramme für die einzelnen Quadratbreiten $0,05$ bis $0,25$ in $0,05$-Schritten. Mit Zunahme der Quadratbreite erhöhen sich in den späteren Zyklen ebenfalls die Anzahl

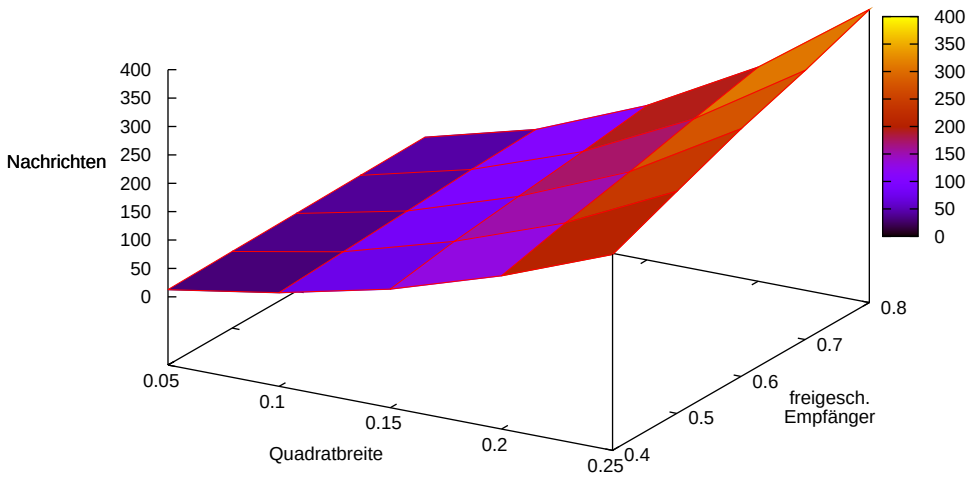


Abbildung 6.8: Nachrichtenaufwand für das Routing futuristischer Nachrichten

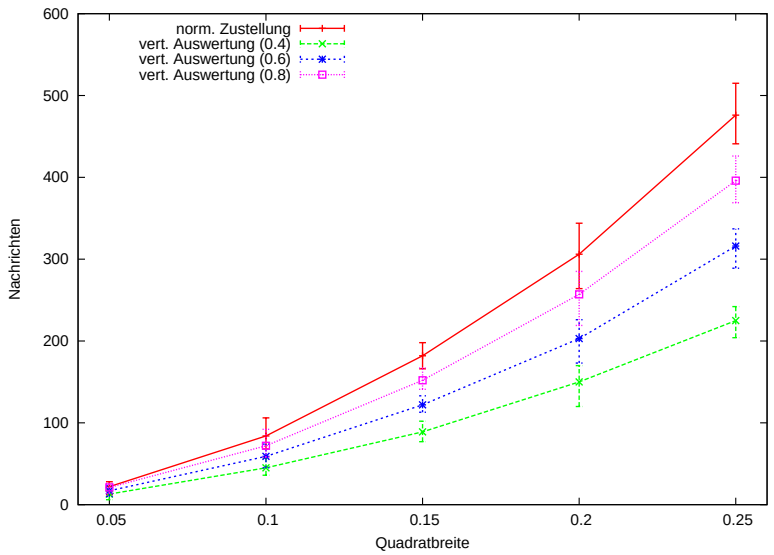


Abbildung 6.9: Nachrichtenaufwand im Vergleich

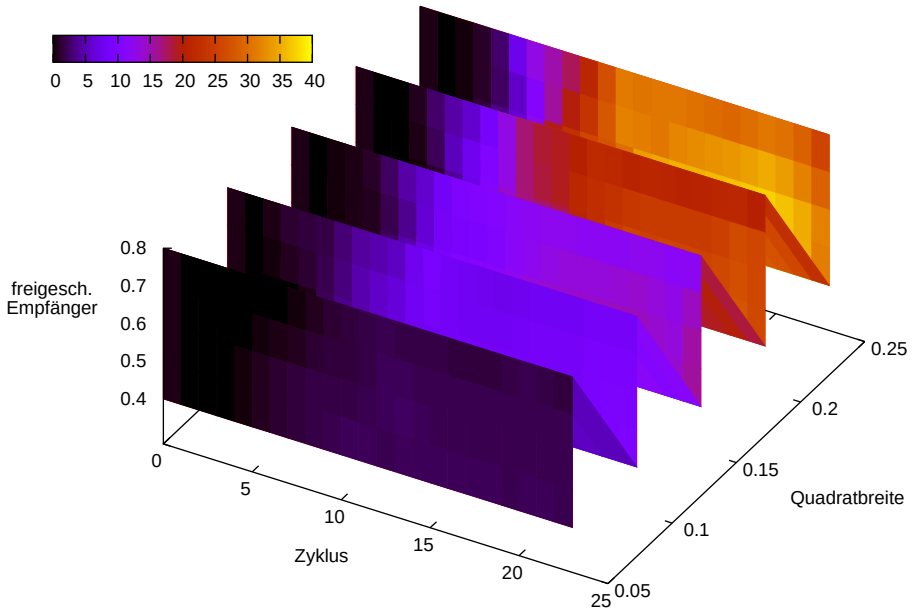
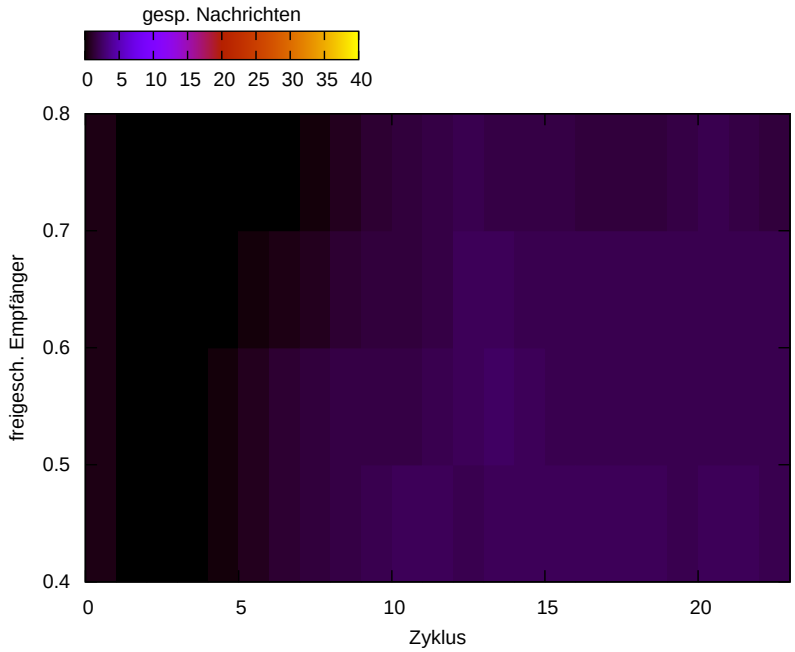


Abbildung 6.10: Zwischengespeicherte futuristische Nachrichten 4D

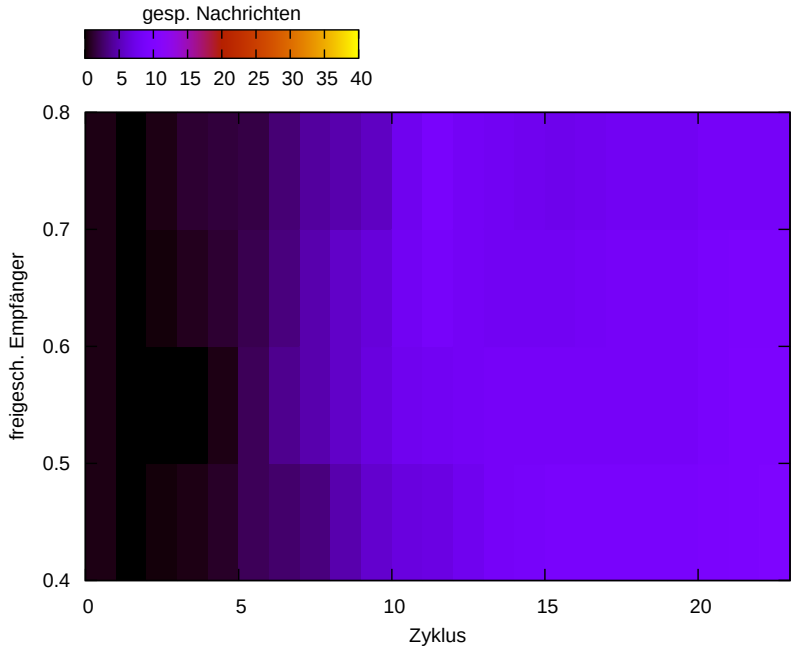
der gespeicherten Nachrichten, da mehr Empfänger adressiert werden, für die ein Objekt zwischengespeichert werden muss. Die Abnahme der zwischengespeicherten Nachrichten mit zunehmender prozentualer Freischaltung der Empfänger ist ebenfalls durch das Entfernen zwischengespeicherter Nachrichten begründet. Insgesamt reicht die Skala bis zu 40 zwischengespeicherten Nachrichten bei einer prozentualen Freischaltung der Empfänger von 40%, was bei einer Zahl von 450 insgesamt adressierten Empfängern recht gering ist. Je nach Anwendung muss auch hier bei der Wahl eines Übermittlungsverfahrens zwischen dem aufgezeigten Speicher- und Nachrichtenaufwand abgewägt werden. Wird eine hohe Rate an Empfängern einer Nachricht erwartet, die einen passenden Kontext anmelden, kann die verteilte Auswertung futuristischer Nachrichten und die damit verbundene Zwischenspeicherung durch Setzen des Attributs „Vert. Auswertung“ einer Nachricht auf den Wert false unterbunden werden. Die Zustellung der futuristischen Nachrichten entspricht dann dem ursprünglichen Verfahren.

6.5 Fazit

Unter Verwendung einer Internet-ähnlichen Topologie mit einer Vielzahl an Knoten und einem entsprechend den Simulationsanforderungen zweckmäßigen Prototypen eines zeitbezogenen Contextcast, stellen die Messwerte durchaus realistische Ergebnisse dar. Insgesamt sind die Durchschnittswerte der unabhängig durchgeführten Simulationsläufe stabil. Größere Abweichungen vom Durchschnittswert, gerade bei den Simulationsergebnissen bezüglich der Zustellung historischer Nachrichten, sind hauptsächlich in den starken Diskrepanzen bezüglich der Knotengrade durch eine Power-Law-Verteilung begründet. Klar ersichtlich ist die enorme Reduzierung des Nachrichtenaufwands bei der Zustellung historischer Nachrichten durch das vollständige Backtracking. Der Mehraufwand an Nachrichten durch das Backtracking selbst, der nur geringfügig über der Anzahl der Nachrichtenempfänger liegt, ist in Relation dazu vernachlässigbar. Die durchschnittlichen maximalen Backtracking-Tiefen,

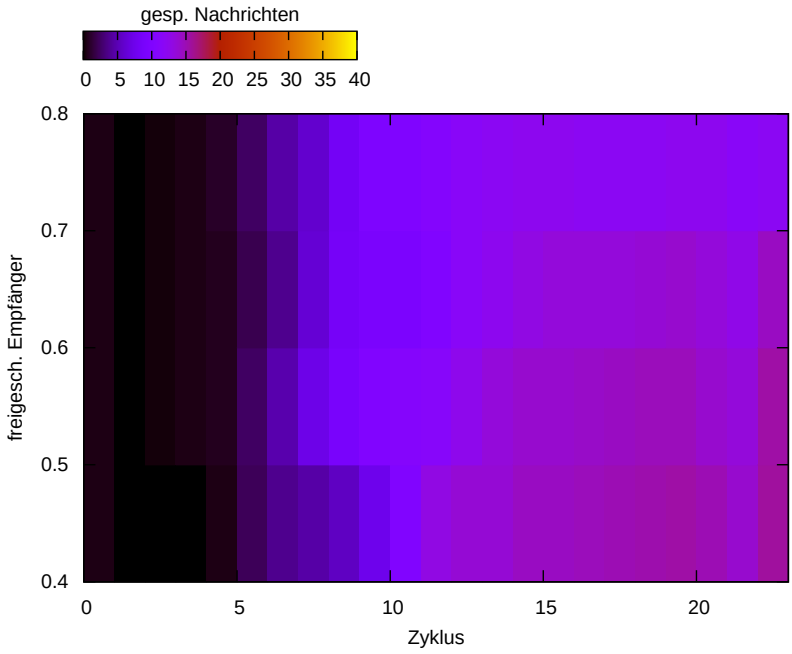


(a) Quadratbreite 0,05

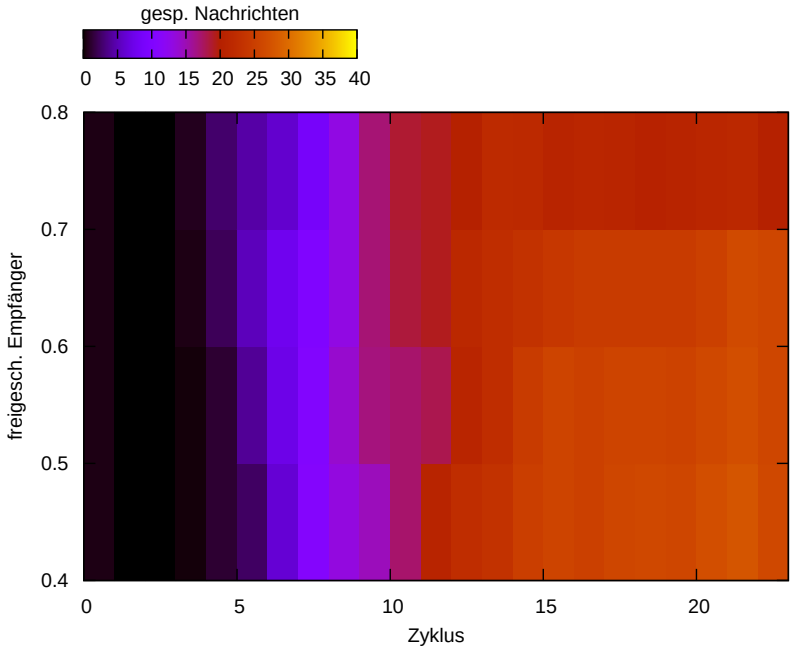


(b) Quadratbreite 0,10

Abbildung 6.11: Zwischengespeicherte futuristische Nachrichten 3D

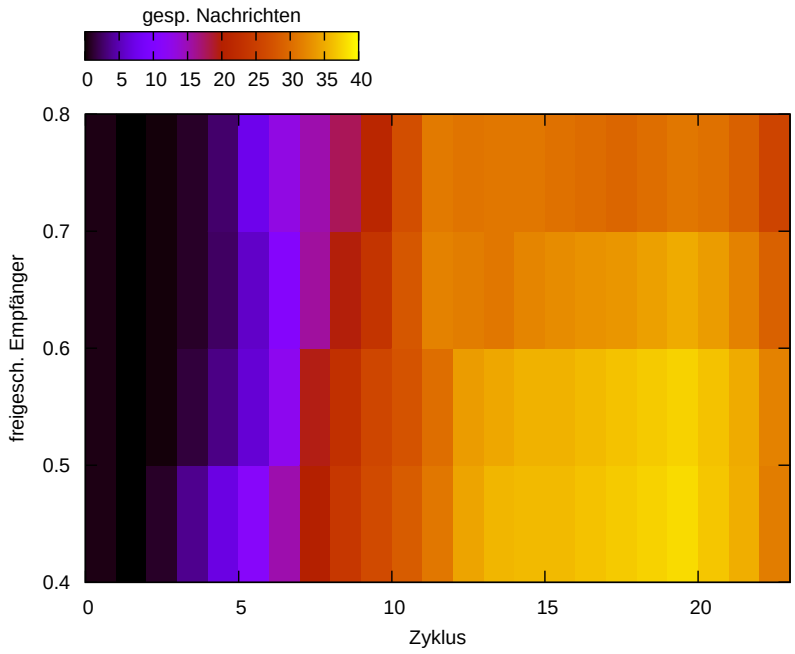


(c) Quadratbreite 0,15



(d) Quadratbreite 0,20

Abbildung 6.11: Zwischengespeicherte futuristische Nachrichten 3D



(e) Quadratbreite 0,25

Abbildung 6.11: Zwischengespeicherte futuristische Nachrichten 3D

als Indikator für die Nachrichtenverzögerung durch das Backtracking, fallen ebenfalls gering aus: Bei einer Anzahl von insgesamt 10.000 Knoten und ca. 450 Nachrichtenempfängern liegt diese gerade einmal bei fünf.

Die Minderung des Nachrichtenaufwands für die Zustellung futuristischer Nachrichten orientiert sich an der Anzahl der Empfänger, die zur Lebenszeit der Nachricht einen passenden Kontext anmelden. Ist diese Zahl relativ hoch, ergibt sich eine geringe Verminderung des Nachrichtenaufwands im Vergleich zur originären Zustellung, andererseits ist der Speicheraufwand für zwischengespeicherte Nachrichten ebenfalls gering. Bei dem gegensätzlichen Verhalten, d.h. falls die Zahl der Empfänger mit passendem Kontext eher gering ausfällt, resultiert eine starke Minderung des Nachrichtenaufwands, die jedoch durch einen ebenfalls relativ hohen Speicheraufwand erkauft wird. Im Großen und Ganzen kommt es bei der Entscheidung für ein Verfahren auf die verfügbare Speicherkapazität der Knoten an: Für 40% an verfügbaren Empfängern, die einen passenden Kontext anmelden, wurden in den Simulationen durchschnittlich gerade einmal maximal $\frac{1}{10}$ der Empfängeranzahl an Nachrichten zu einem Zeitpunkt zwischengespeichert.

7 Zusammenfassung und Ausblick

Nachfolgend wird der Gegenstand der Arbeit mit dem Fokus auf die wesentlichen Aspekte und Konzepte eines zeitbezogenen Contextcast kurz zusammengefasst. Anschließend werden in einem weiteren Abschnitt ergänzende Forschungsgebiete, Erweiterungen und Optimierungen bezüglich des zeitbezogenen Contextcast aufgezeigt.

7.1 Zusammenfassung

In dieser Arbeit wurde ein zeitbezogenes Contextcast beschrieben, das den in Kapitel 3 genannten Anforderungen genügt. Dieses System basiert auf dem bereits existierenden Contextcast. Für die Entwicklung des Systems wurden teilweise bereits vorhandene Konzepte wie das Xcast verwendet. Das Contextcast, das als Basissystem für das zeitbezogenen Contextcast dient, entspricht einem Overlay-Netz, das auf einem beliebigen Transportprotokoll für Nachrichten aufgesetzt werden kann und von dessen Eigenschaften abstrahiert. Es setzt eine kontextbezogene Kommunikation um, in der Nachrichten an Kontexte der Teilnehmer adressiert werden können. Eine enge Verwandtschaft zu Systemen wie dem Publish/Subscribe und Geocast ist erkennbar. Aus diesen wurden verschiedene Ideen und Konzepte durch Anpassungen in das Contextcast integriert, welches somit die verschiedenen Anforderungen an die einzelnen Systeme durch ein einziges System erfüllt. Kapitel 2 erläutert diese unterschiedlichen Systeme, Anforderungen und Konzepte in einem Grundlagenbereich zum zeitbezogenen Contextcast.

Die zentralen Anforderungen an ein zeitbezogenes Contextcast liegen in den Bereichen der Archivierung von Informationsobjekten, dem Schutz der Privatsphäre eines jeden Benutzers, der Definition der benötigten Komponenten, der Unterstützung mobiler Endgeräte und der effizienten Übermittlung der Nachrichten an die Empfänger. Primäres Ziel des zeitbezogenen Contextcast ist die effiziente Unterstützung zeitbezogener Nachrichten, die durch eine kontextbezogene Adressierung an vergangene und zukünftige Kontexte eines Benutzers adressiert werden können. Ein Kontext beschreibt dabei die aktuellen Situationseigenschaften einer Entität, die mit dem System verbunden ist. Kapitel 4 erläutert die einzelnen Aspekte eines zeitbezogenen Contextcast.

Die für das Routing der Nachrichten benötigten Informationsobjekte sind Kontexte und Nachrichten. Die Historisierung der Kontexte erfolgt nach deren Abmeldung durch den jeweiligen ContextNode. Dabei wird dem Kontext ein Zeitintervall seiner Gültigkeit im System zugewiesen, der durch die Zeitpunkte seiner An- und Abmeldung festgelegt wird. Für die Darstellung der Zeitpunkte wurde die UTC-Repräsentation gewählt, wobei lokale Abweichungen ebenfalls berücksichtigt werden. Für die Historisierung der Nachrichten wird ein Mailboxsystem verwendet, das durch bestimmte ContextRouter oder ContextNodes zur Verfügung gestellt wird. Dies geschieht in dem Fall, dass ein Benutzer für längere Zeit nicht mit dem System verbunden ist, so dass die Nachricht nicht direkt zugestellt werden kann und daher gespeichert werden muss.

Ein weiterer Auslöser für die Archivierung einer Nachricht ist der Wechsel des Zugangsbereichs eines Benutzers verbunden mit dem Wechsel der Client-ID. In diesem Fall können die vergangenen Kontexte eines Benutzers diesem durch das System nicht mehr zugeordnet werden. Die einzige Komponente des Systems die dies bewerkstelligen kann, ist der Trusted Node des Benutzers, der in der Client-ID angegeben ist. Diese zusätzliche Komponente befindet sich auf dem gleichen Knoten wie das Mailboxsystem eines Benutzers und dient dem Schutz der Privatsphäre. Durch den möglichen Wechsel der Client-ID eines Benutzers wird ein Tracking der Aufenthaltsorte und Eigenschaftswerte eines Benutzers verhindert.

Während für die Zeitinformationen eines Kontextes ein Zeitintervall ausreichend ist, müssen für Nachrichten konkrete Attributfilter definiert werden, die zusätzlich Operatoren auf Zeitintervalle beinhalten. Weiter muss für die konkreten Attributfilter festgelegt werden, wie und wann sie im Rahmen eines Matchings auf Kontexte ausgewertet werden können und ab wann die Nachricht im System gelöscht werden darf. Dies legt zugleich den Grundbaustein für ein Routing zeitbezogener Nachrichten.

In Abschnitt 4.5 sind, neben dem Matching, die einzelnen Verfahren für ein einfaches Routing der unterschiedlichen zeitbezogenen Nachrichten beschrieben. Historische Nachrichten werden demnach direkt an die ContextNode-Empfänger übertragen. Nach einem Matching kann die Nachricht von diesen aus entweder direkt an die Benutzer oder an die Trusted Nodes der Benutzer ausgeliefert werden. Futuristische und hybride Nachrichten werden ebenfalls an die ContextNode-Empfänger übertragen, müssen jedoch bei diesen, aufgrund einer längeren Auswertung, zwischengespeichert werden. Im Regelfall werden diese nach dem Matching ebenfalls direkt an die Empfänger übertragen.

Die geschilderten Routingverfahren lassen einige Optimierungen zu, mit denen die Effizienz und damit Skalierbarkeit des Systems weiter gesteigert werden kann. Neben Verfahren zur Duplikatvermeidung von Nachrichten an verschiedenen Knoten des Systems beschreibt Kapitel 5 eine Auswahl von optimierten Routingverfahren für die einzelnen Nachrichtenkategorien der zeitbezogenen Nachrichten. Historische Nachrichten und der historische Nachrichtenteil hybrider Nachrichten können durch ein gebündeltes Contextcast effizienter zugestellt werden, da der Nachrichtenaufwand geringer ausfällt. Dafür werden die Trusted-Node-Adressen nach einem Matching einer Nachricht bei den ContextNode-Empfängern in entgegengesetzter Richtung des Verteilbaums der Nachricht an den ersten Splitpoint-Knoten übertragen. Von dort aus kann eine Nachricht an alle Trusted-Node-Empfänger geschickt werden, wodurch eine bessere Umsetzung der Multicasteigenschaften erreicht wird. Über das Attribut „TREE-LEVEL“ einer Nachricht kann eine feingranulare Anpassung am Backtracking-Aufwand vorgenommen werden. Mit zunehmendem Wert des Attributs erfolgt ein fließender Übergang vom vollständigen Backtracking hin zum ursprünglichen Verfahren der Nachrichtenübermittlung. Die verteilte Auswertung futuristischer Nachrichten minimiert den Nachrichtenaufwand durch eine frühzeitige Auswertung der futuristischen Nachrichten, bevor diese die ContextNode-Empfänger erreichen. Dadurch lassen sich Äste des Verteilbaums zu Empfängern abschneiden, die in der Lebensdauer der Nachricht keine passenden Kontexte anmelden. Die Duplikatvermeidung von Nachrichten, die parallel zum Routing durchgeführt wird, verhindert die überflüssige Übertragung von Nachrichten und verbessert daher gezielt die Unterstützung von mobilen Endgeräten. Weitere spezielle Mechanismen zur Unterstützung dieser wurden in die Routingverfahren integriert, wobei die Erkennung mobiler Endgeräte durch ein Attribut in den Kontexten erfolgt.

Die Effizienz der Routingverfahren lässt sich durch entsprechende Simulationen abschätzen. Kapitel 6 präsentiert die Simulationsergebnisse anhand von Schaubildern. Das gebündelte Contextcast historischer Nachrichten verzeichnet im Gegensatz zum ursprünglichen Verfahren der Übermittlung historischer Nachrichten eine erhebliche Aufwandsreduzierung. Die zusätzliche Verzögerung der Nachrichtenübermittlung fällt dabei gering aus. Bei der verteilten Auswertung futuristischer Nachrichten entspricht der prozentuale Anteil der durch passende Kontexte freigeschalteten Empfänger in etwa dem Verhältnis des Nachrichtenaufwands zum Nachrichtenaufwand bei der Freischaltung aller Empfänger. Eine Nachrichtenverzögerung entsteht im Allgemeinen nicht, die notwendige Zwischenspeicherung der Nachrichten erfordert jedoch einen erhöhten Speicheraufwand. Falls dieser zusätzliche Speicheraufwand nicht angeboten werden kann oder von einer hohen Anzahl an Empfängern ausgegangen wird, kann die verteilte Auswertung für die einzelnen Nachrichten durch Setzen eines Attributs in der Nachricht abgeschaltet werden. Die genauen Messergebnisse der Simulationen

sind in Anhang A aufgelistet.

7.2 Ausblick

In dieser Arbeit ist ein zeitbezogenes Contextcast beschrieben, das eine hohe Effizienz und Skalierbarkeit aufweist. Das Basissystem Contextcast befindet sich zum aktuellen Zeitpunkt jedoch noch in der Entwicklungsphase, so dass kein lauffähiger Prototyp zur Verfügung steht. Ein Prototyp, der die Konzepte des Contextcast und die Erweiterungen zu einem zeitbezogenen Contextcast umsetzt, stellt den nächsten Schritt in die Richtung eines ausgereiften Systems dar. Mit diesem Prototypen lassen sich die einzelnen Verfahren in realen Umgebungen und durch die Teilnahme realer Endgeräte analysieren. Gewisse Mängel und Engpässe der Verfahren, sowie unvorhergesehene Problemszenarien werden ausschließlich durch den Einsatz eines lauffähigen Systems sichtbar. Hieraus lassen sich möglicherweise weitere Erweiterungen und Optimierungsansätze des Systems ableiten. In den folgenden Abschnitten wird eine Auswahl von weiteren bekannten Verbesserungsmöglichkeiten eines zeitbezogenen Contextcast vorgestellt.

Backtracking Das Backtracking im Rahmen des gebündelten Contextcast historischer Nachrichten lässt sich hinsichtlich bestimmter Anforderungen weiter optimieren. Ist es beispielsweise aufgrund des Anwendungsumfelds erforderlich, dass mehrere Nachrichten in gewissen zeitlichen Abständen an den gleichen Empfänger oder die gleiche Empfängergruppe geschickt werden, kann das Backtracking für die nach der ersten Nachricht folgenden Nachrichten entfallen. Beim Backtracking der Ergebnisse des Matchings der ersten Nachricht bei den ContextNode-Empfängern muss dazu lediglich die Trusted-Node-Liste weiter an den ContextNode des Senders geleitet werden. Diese werden beim ContextNode zwischengespeichert, so dass die darauf folgenden Nachrichten mit gleichen Empfängern direkt beim ContextNode an die Trusted Nodes der jeweiligen Empfänger verschickt werden können. Die Übermittlung der Nachricht an die ContextNode-Empfänger und das anschließende Backtracking werden dadurch vollständig umgangen. Das Resultat sind umfangreiche Einsparungen im Nachrichtenaufwand und eine Reduzierung der Nachrichtenverzögerungen. Dies gilt jedoch nur unter der Voraussetzung, dass der Versand von verschiedenen Nachrichten an die gleichen Empfänger Anwendung findet. Die Nachrichtenverzögerung lässt sich auf Kosten des Nachrichtenaufwands weiter reduzieren, wenn Nachrichten durch den ContextNode des Senders an aktuell angemeldete Benutzer direkt verschickt werden, die ihre Client-ID in der Zwischenzeit nicht geändert haben.

Eine weitere Optimierung bezüglich der Nachrichtenverzögerung und des Nachrichtenaufwands wird durch das Caching von historischen Kontexten erreicht. Diese können beispielsweise entlang des Verteilbaums einer historischen Nachricht beim Backtracking zwischengespeichert werden. Im Bereich der unstrukturierten Peer-to-Peer-Systeme wurden Caching-Verfahren bereits intensiv untersucht [CS02]. Inwieweit sich diese auf ein Contextcast übertragen lassen muss noch evaluiert werden. Durch das Caching im Contextcast verspricht man sich jedenfalls kürzere Pfade bei der Übermittlung historischer Nachrichten, da die Nachrichten schon vor den ContextNode-Empfängern auf zwischengespeicherte historische Kontexte ausgewertet werden können. Ein bekannter Nachteil bei dieser Art von Caching ist, dass sich die Nachrichten durch eine frühzeitige Auswertung den Verfahren zur Duplikatvermeidung entziehen und aus der einheitlichen Übertragung einer Nachricht fallen. Dies erhöht wiederum den Nachrichtenaufwand des Routings. Es ist folglich weitere Forschungsarbeit gefragt, um das Caching historischer Nachrichten effizient zu gestalten.

Verteilte Auswertung futuristischer Nachrichten Die verteilte Auswertung futuristischer Nachrichten kann durch ein Attribut in einer zeitbezogenen Nachricht abgeschaltet werden. Ein Kriterium, das die Abschaltung der Optimierung erforderlich macht, ist der zusätzliche Speicheraufwand des Verfahrens. Steht bei den Knoten des Netzes nicht genügend Speicher zur Verfügung, oder ist der Speicher eines Knotens bei der verteilten Auswertung futuristischer Nachrichten aufgebraucht, kann eine Zustellung der Nachricht an die entsprechenden ContextNode-Empfänger nicht mehr garantiert werden. Unter diese Umständen wird für das beschriebene zeitbezogene Contextcast eine Abschaltung der verteilten Auswertung futuristischer Nachrichten empfohlen.

Eine Lösung, die ohne eine komplette Abschaltung der verteilten Auswertung futuristischer Nachrichten auskommt, kann durch statistische Analysen im laufenden Betrieb des Systems erreicht werden. Anhand von ausgewählten Metriken und Berechnungen müssen ContextNodes identifiziert werden, die regelmäßig Kontexte an- und abmelden und zu welchen eine häufige Übertragung zwischengespeicherter futuristischer Nachrichten erfolgt. Werden futuristische Nachrichten an diese ContextNodes bei der verteilten Auswertung direkt übermittelt, ohne diese für bestimmte ContextNodes im Netz zwischenzuspeichern, wird der insgesamt notwendige Speicheraufwand reduziert und gleichzeitig an den Rand des Netzes verlagert. Falls der insgesamt im Netz verfügbare Speicher theoretisch ausreicht und nur einzelne Knoten überlastet werden, lässt sich eine einfachere Lösung anwenden, bei der auf statistische Untersuchungen verzichtet werden kann. Dazu leitet der überlastete Knoten eine futuristische Nachrichten direkt in Richtung der Empfänger weiter und delegiert dadurch die Zwischenspeicherung der Nachricht an seine Kindknoten des Verteilbaums der Nachricht. Die Kindknoten setzen dieses Verfahren rekursiv fort, falls diese ebenfalls überlastet sind. Dadurch erfolgt quasi eine Suche nach nicht überlasteten Knoten abwärts des Verteilbaums. Mit jedem Schritt weiter in Richtung ContextNode-Empfänger wird allerdings die Nachrichtenersparnis der verteilten Auswertung zu Nichte gemacht.

Weitere Verbesserungsmöglichkeiten bieten sich im Bereich der Kontextdatenbankverteilung an, das in dieser Arbeit relativ kurz behandelt wurde. Verfahren dieses Themengebiets haben starke Auswirkungen auf die Verfügbarkeit, Speicherlastverteilung und Persistenz der historischen Kontexte. Falls die in dieser Arbeit beschriebene Methode zur Kontextdatenbankverteilung mit der attributabhängigen Gruppierung von Kontexten zum Einsatz kommt, so bietet sich ein Advertisement von historischen Kontexten an. Das Gleiche gilt in dem Falle, dass bestimmte ContextNodes nur Kontexte mit einer bestimmten Einschränkung bezüglich der möglichen Werte von Attributen erlauben. Diese Information kann im Netz in Form von Advertisements bekannt gemacht werden, so dass Nachrichten, die grundsätzlich nicht zu den vom ContextNode freigegebenen Kontexten passen, erst gar nicht zu den ContextNodes weitergeleitet werden.

Bisher fand das Konzept des hierarchischen Routings im Zusammenhang mit dem Contextcast wenig Beachtung. Es wird strikt von einem einheitlich zusammenhängenden System ausgegangen, in dem die Kontexte meist vollständig im Netz propagiert werden. Oft kommt es jedoch vor, dass mobile Endgeräte, die sich beispielsweise in einem bestimmten Gebäude aufhalten, nur von Sendern des gleichen Gebäudes adressiert werden. In diesem Fall kann der Nachrichtenversand zwar auf ein bestimmtes geographisches Gebiet beschränkt werden, beispielsweise auf den Bereich des Gebäudes, die Kontexte werden aber unnötigerweise im gesamten System verteilt. Dies kann durch eine Unterteilung des Netzes in kleinere Netze, beispielsweise ähnlich dem Subnetting in TCP/IP-basierten Netzen, verhindert werden. Dabei wird das Netz in unabhängige aber zusammenhängende Administrationsdomänen aufgeteilt. Dadurch erhöht sich gleichzeitig der Sicherheitsaspekt in Bezug auf den Schutz der Privatsphäre eines Benutzers, da die Kontexte nicht zwangsläufig außerhalb ihres Subnetzes

propagiert werden müssen.

Eine weitere Möglichkeit zum Schutz der Privatsphäre ist die Verschlüsselung von Attributen des Kontextes. Generell kann eine Verschlüsselung, beispielsweise des Nachrichteninhalts, durch die Anwendung durchgeführt werden, ohne dass dazu Änderungen am System notwendig werden. Nach einer Verschlüsselung der Attribute eines Kontextes durch die Anwendung ist jedoch kein Matching auf Nachrichten mehr möglich. Ist es gewollt, dass nur bestimmte Sender einen Kontext durch eine Nachricht adressieren können und die Attribute des Kontextes durch andere Benutzer nicht zu entschlüsseln sind, müssen Verschlüsselungssysteme in das Contextcast integriert werden, die eine Verschlüsselung der Kontexte und eine Authentifizierung der Benutzer ermöglichen.

Anhang A: Simulationsergebnisse

Im Folgenden sind die durchschnittlichen, maximalen und minimalen Messwerte der zehn unabhängig durchgeführten Simulationsläufe je Nachrichtenkategorie tabellarisch dargestellt.

Tabelle A.1 listet die Messwerte für die historischen Nachrichten auf, die durch die originäre Zustellung übertragen wurden. Die erste Zeile gibt die Quadratbreite der Nachrichten wieder. Alle anderen Zeilen enthalten die Durchschnittswerte für den Nachrichtenaufwand, die ContextNode-Empfänger und die Trusted Node Empfänger mit den minimalen und maximalen Werten Min und Max der Simulationsläufe.

In Tabelle A.2 werden für die durch das vollständige Backtracking zugestellten Nachrichten zusätzlich die Messwerte für den Backtracking-Aufwand in Nachrichten und der maximalen Backtracking-Tiefe aufgelistet.

Beim tiefenbeschränkten Backtracking wird für die Nachrichten zusätzlich ein TREELEVEL festgelegt. Dieser ist in Zeile zwei der Tabelle A.3 zusammen mit den Messwerten der Simulation aufgeführt.

Tabelle A.4 und A.5 bildet die Messwerte der futuristischen Nachrichten für die originäre Zustellung und die verteilte Auswertung ab. Die Zeile mit dem Titel „frEmpf.“ gibt die Anzahl an freigeschaltenen Empfängern in Relation zu den gesamten Empfängern in Dezimalbruchdarstellung an, „beEmpf.“ steht für die Anzahl an Bestätigungen des Nachrichtenerhalts durch die ContextNode-Empfänger.

Die Mengen an zwischengespeicherten futuristischen Nachrichten in jedem Zyklus der Simulation sind in Tabelle A.6 angegeben.

LokBreite	Nachr.	Min	Max	Empf.	Min	Max	TNs	Min	Max
0.05	4404	3028	5711	18	13	22	2522	1673	3383
0.1	18421	15339	21934	71	65	76	10577	8755	12650
0.15	40797	36005	45903	157	137	175	23395	20570	26198
0.2	74671	69208	85994	287	266	316	42807	38620	49689
0.25	107818	83796	121825	442	411	464	61276	45886	69775

Tabelle A.1: Originäre Zustellung historischer Nachrichten

LokBreite	Nachr.	Min	Max	Empf.	Min	Max	TNs	Min	Max
0.05	508	501	515	18	13	22	300	299	300
0.1	569	560	582	71	65	76	300	300	300
0.15	665	642	686	157	137	175	300	300	300
0.2	802	778	824	287	266	316	300	300	300
0.25	964	926	987	442	411	464	300	300	300

BacktrNachr.	Min	Max	MaxTiefe	Min	Max
20	13	25	2	1	4
83	76	91	3	2	5
178	159	197	4	2	5
316	297	338	5	3	6
478	441	498	5	4	7

Tabelle A.2: Vollständiges Backtracking der historischen Nachrichten

LokBreite	TREELEVEL	Nachr.	Min	Max	Empf.	Min	Max
0.05	1	2000	751	4427	18	13	22
0.05	2	4174	1984	5711	18	13	22
0.05	3	4379	3028	5711	18	13	22
0.1	1	3007	770	19152	71	65	76
0.1	2	8398	2242	19357	71	65	76
0.1	3	16538	5544	21934	71	65	76
0.15	1	4297	1113	27734	157	137	175
0.15	2	12062	2099	40158	157	137	175
0.15	3	26818	11917	40333	157	137	175
0.2	1	4699	1211	30324	287	266	316
0.2	2	26607	2782	74124	287	266	316
0.2	3	55455	24197	79652	287	266	316
0.25	1	3772	1014	16461	442	411	464
0.25	2	22796	5271	39129	442	411	464
0.25	3	65124	32871	101722	442	411	464

TNs	Min	Max	BacktrNachr.	Min	Max	MaxTiefe	Min	Max
1154	420	2496	13	0	21	1	0	3
2396	1177	3383	2	0	15	0	0	2
2510	1673	3383	0	0	2	0	0	1
1728	404	10990	72	2	89	2	1	4
4883	1275	11086	47	0	80	1	0	3
9498	3258	12650	8	0	52	1	0	2
2445	588	16067	165	75	194	3	1	4
6961	1190	22816	129	0	186	2	0	3
15508	7009	23630	66	0	147	1	0	2
2618	558	17639	302	228	333	4	2	5
15454	1466	42732	220	37	296	3	1	4
31890	13908	45789	87	0	215	2	0	3
1896	316	8739	464	395	493	4	3	6
12971	2745	22939	386	275	458	3	2	5
37136	18556	58941	200	57	353	2	1	4

Tabelle A.3: Tiefenbeschränktes Backtracking der historischen Nachrichten

LokBreite	Nachr.	Min	Max	Empf.	Min	Max
0.05	22	12	28	15	9	21
0.1	84	68	106	69	56	82
0.15	182	166	198	160	147	183
0.2	306	264	344	275	229	312
0.25	476	441	515	437	396	475

Tabelle A.4: Originäre Zustellung futuristischer Nachrichten

LokBreite	frEmpf.	Nachr.	Min	Max	Empf.	Min	Max	beEmpf.	Min	Max
0.05	0.4	13	6	18	15	9	21	7	5	10
0.05	0.5	15	10	18	15	9	21	8	5	12
0.05	0.6	17	10	23	15	9	21	10	6	15
0.05	0.7	19	12	23	15	9	21	11	7	15
0.05	0.8	21	14	26	15	9	21	13	8	18
0.1	0.4	45	36	57	69	56	82	30	24	38
0.1	0.5	52	40	72	69	56	82	37	31	48
0.1	0.6	59	45	74	69	56	82	43	34	55
0.1	0.7	67	53	82	69	56	82	50	40	62
0.1	0.8	72	58	92	69	56	82	56	45	66
0.15	0.4	89	77	102	160	147	183	68	59	79
0.15	0.5	108	97	119	160	147	183	86	76	103
0.15	0.6	122	113	133	160	147	183	100	89	116
0.15	0.7	136	126	149	160	147	183	113	103	130
0.15	0.8	152	141	167	160	147	183	130	118	147
0.2	0.4	150	120	170	275	229	312	118	99	134
0.2	0.5	178	149	199	275	229	312	145	122	165
0.2	0.6	203	173	226	275	229	312	171	140	193
0.2	0.7	231	195	259	275	229	312	198	165	225
0.2	0.8	257	219	285	275	229	312	225	185	253
0.25	0.4	225	204	242	437	396	475	186	160	205
0.25	0.5	271	244	300	437	396	475	231	199	262
0.25	0.6	316	289	337	437	396	475	274	245	292
0.25	0.7	354	332	385	437	396	475	313	288	342
0.25	0.8	396	369	426	437	396	475	354	318	385

Tabelle A.5: Verteilte Auswertung futuristischer Nachrichten

Zyklus	LokBreite	frEmpf.	Nachr.	Min	Max
0	0.05	0.4	1	1	1
0	0.05	0.5	1	1	1
0	0.05	0.6	1	1	1
0	0.05	0.7	1	1	1
0	0.05	0.8	1	1	1
0	0.1	0.4	1	1	1
0	0.1	0.5	1	1	1
0	0.1	0.6	1	1	1
0	0.1	0.7	1	1	1
0	0.1	0.8	1	1	1
0	0.15	0.4	1	1	1
0	0.15	0.5	1	1	1
0	0.15	0.6	1	1	1
0	0.15	0.7	1	1	1
0	0.15	0.8	1	1	1
0	0.2	0.4	1	1	1
0	0.2	0.5	1	1	1
0	0.2	0.6	1	1	1
0	0.2	0.7	1	1	1
0	0.2	0.8	1	1	1
0	0.25	0.4	1	1	1
0	0.25	0.5	1	1	1
0	0.25	0.6	1	1	1
0	0.25	0.7	1	1	1
0	0.25	0.8	1	1	1
1	0.05	0.4	0	0	1
1	0.05	0.5	0	0	1
1	0.05	0.6	0	0	1
1	0.05	0.7	0	0	1
1	0.05	0.8	0	0	1
1	0.1	0.4	0	0	0
1	0.1	0.5	0	0	0
1	0.1	0.6	0	0	0
1	0.1	0.7	0	0	0
1	0.1	0.8	0	0	0
1	0.15	0.4	0	0	0
1	0.15	0.5	0	0	0
1	0.15	0.6	0	0	0
1	0.15	0.7	0	0	0
1	0.15	0.8	0	0	0
1	0.2	0.4	0	0	0
1	0.2	0.5	0	0	0
1	0.2	0.6	0	0	0
1	0.2	0.7	0	0	0
1	0.2	0.8	0	0	0
1	0.25	0.4	0	0	0
1	0.25	0.5	0	0	0
1	0.25	0.6	0	0	0
1	0.25	0.7	0	0	0
1	0.25	0.8	0	0	0
2	0.05	0.4	0	0	0
2	0.05	0.5	0	0	0
2	0.05	0.6	0	0	0
2	0.05	0.7	0	0	0
2	0.05	0.8	0	0	0
2	0.1	0.4	0	0	1
2	0.1	0.5	0	0	0
2	0.1	0.6	0	0	0
2	0.1	0.7	0	0	1
2	0.1	0.8	0	0	1
2	0.15	0.4	0	0	0
2	0.15	0.5	0	0	0
2	0.15	0.6	0	0	1
2	0.15	0.7	0	0	2
2	0.15	0.8	0	0	1
2	0.2	0.4	0	0	0
2	0.2	0.5	0	0	0
2	0.2	0.6	0	0	0
2	0.2	0.7	0	0	1
2	0.2	0.8	0	0	0
2	0.25	0.4	0	0	1
2	0.25	0.5	0	0	1
2	0.25	0.6	0	0	1
2	0.25	0.7	0	0	2
2	0.25	0.8	0	0	1
3	0.05	0.4	0	0	1
3	0.05	0.5	0	0	0
3	0.05	0.6	0	0	2
3	0.05	0.7	0	0	0
3	0.05	0.8	0	0	0
3	0.1	0.4	1	0	3
3	0.1	0.5	0	0	0
3	0.1	0.6	0	0	0
3	0.1	0.7	1	0	4
3	0.1	0.8	1	0	4
3	0.15	0.4	0	0	0
3	0.15	0.5	0	0	0
3	0.15	0.6	1	0	2
3	0.15	0.7	0	0	0
3	0.15	0.8	1	0	6
3	0.2	0.4	0	0	4
3	0.2	0.5	0	0	0

Tabelle A.6: Zwischengespeicherte futuristische Nachrichten

3	0.2	0.6	0	0	2
3	0.2	0.7	0	0	1
3	0.2	0.8	0	0	2
3	0.25	0.4	3	0	23
3	0.25	0.5	1	0	6
3	0.25	0.6	0	0	2
3	0.25	0.7	1	0	10
3	0.25	0.8	0	0	1
4	0.05	0.4	0	0	1
4	0.05	0.5	0	0	1
4	0.05	0.6	0	0	0
4	0.05	0.7	0	0	0
4	0.05	0.8	0	0	2
4	0.1	0.4	1	0	7
4	0.1	0.5	0	0	1
4	0.1	0.6	0	0	0
4	0.1	0.7	2	0	9
4	0.1	0.8	1	0	4
4	0.15	0.4	0	0	2
4	0.15	0.5	0	0	2
4	0.15	0.6	1	0	8
4	0.15	0.7	0	0	1
4	0.15	0.8	1	0	6
4	0.2	0.4	1	0	12
4	0.2	0.5	0	0	0
4	0.2	0.6	1	0	4
4	0.2	0.7	1	0	6
4	0.2	0.8	2	0	12
4	0.25	0.4	7	0	52
4	0.25	0.5	4	0	18
4	0.25	0.6	1	0	3
4	0.25	0.7	2	0	18
4	0.25	0.8	1	0	3
5	0.05	0.4	0	0	1
5	0.05	0.5	1	0	3
5	0.05	0.6	0	0	2
5	0.05	0.7	0	0	0
5	0.05	0.8	0	0	2
5	0.1	0.4	2	0	8
5	0.1	0.5	1	0	8
5	0.1	0.6	1	0	3
5	0.1	0.7	2	0	12
5	0.1	0.8	1	0	6
5	0.15	0.4	1	0	6
5	0.15	0.5	1	0	4
5	0.15	0.6	1	0	7
5	0.15	0.7	1	0	4
5	0.15	0.8	2	0	11
5	0.2	0.4	3	0	20
5	0.2	0.5	1	0	4
5	0.2	0.6	3	0	19
5	0.2	0.7	4	0	25
5	0.2	0.8	4	0	23
5	0.25	0.4	10	0	36
5	0.25	0.5	7	0	30
5	0.25	0.6	2	0	9
5	0.25	0.7	5	0	27
5	0.25	0.8	3	0	18
6	0.05	0.4	1	0	2
6	0.05	0.5	1	0	5
6	0.05	0.6	1	0	2
6	0.05	0.7	0	0	0
6	0.05	0.8	0	0	2
6	0.1	0.4	2	0	12
6	0.1	0.5	4	0	17
6	0.1	0.6	3	0	12
6	0.1	0.7	2	0	9
6	0.1	0.8	2	0	7
6	0.15	0.4	3	0	16
6	0.15	0.5	4	0	17
6	0.15	0.6	4	0	18
6	0.15	0.7	2	0	12
6	0.15	0.8	5	0	16
6	0.2	0.4	3	0	10
6	0.2	0.5	3	0	21
6	0.2	0.6	9	0	32
6	0.2	0.7	5	0	23
6	0.2	0.8	5	0	28
6	0.25	0.4	15	0	45
6	0.25	0.5	13	0	38
6	0.25	0.6	3	0	8
6	0.25	0.7	13	0	36
6	0.25	0.8	9	0	39
7	0.05	0.4	1	0	4
7	0.05	0.5	2	0	4
7	0.05	0.6	1	0	5
7	0.05	0.7	0	0	2
7	0.05	0.8	0	0	0
7	0.1	0.4	1	0	5
7	0.1	0.5	4	0	9
7	0.1	0.6	4	0	12
7	0.1	0.7	4	0	13
7	0.1	0.8	4	0	13
7	0.15	0.4	2	0	13

Tabelle A.6: Zwischengespeicherte futuristische Nachrichten

7	0.15	0.5	6	0	27
7	0.15	0.6	5	0	12
7	0.15	0.7	4	0	18
7	0.15	0.8	7	0	20
7	0.2	0.4	12	0	35
7	0.2	0.5	7	0	28
7	0.2	0.6	10	0	30
7	0.2	0.7	7	0	32
7	0.2	0.8	7	0	20
7	0.25	0.4	13	5	31
7	0.25	0.5	20	0	38
7	0.25	0.6	13	0	34
7	0.25	0.7	15	0	55
7	0.25	0.8	15	0	50
8	0.05	0.4	1	0	2
8	0.05	0.5	2	0	3
8	0.05	0.6	1	0	4
8	0.05	0.7	1	0	2
8	0.05	0.8	0	0	1
8	0.1	0.4	2	0	9
8	0.1	0.5	6	0	12
8	0.1	0.6	5	0	11
8	0.1	0.7	6	0	17
8	0.1	0.8	3	0	7
8	0.15	0.4	2	0	11
8	0.15	0.5	8	0	21
8	0.15	0.6	10	0	19
8	0.15	0.7	6	0	16
8	0.15	0.8	7	0	13
8	0.2	0.4	14	0	30
8	0.2	0.5	10	0	25
8	0.2	0.6	15	0	34
8	0.2	0.7	8	0	23
8	0.2	0.8	13	0	24
8	0.25	0.4	24	6	43
8	0.25	0.5	23	0	45
8	0.25	0.6	22	0	50
8	0.25	0.7	13	0	35
8	0.25	0.8	18	0	40
9	0.05	0.4	2	0	4
9	0.05	0.5	2	0	4
9	0.05	0.6	2	0	5
9	0.05	0.7	1	0	5
9	0.05	0.8	1	0	2
9	0.1	0.4	5	0	14
9	0.1	0.5	6	0	15
9	0.1	0.6	6	1	11
9	0.1	0.7	6	0	12
9	0.1	0.8	4	0	11
9	0.15	0.4	4	0	11
9	0.15	0.5	8	0	21
9	0.15	0.6	10	0	23
9	0.15	0.7	8	0	20
9	0.15	0.8	12	2	19
9	0.2	0.4	15	0	29
9	0.2	0.5	14	0	25
9	0.2	0.6	17	0	35
9	0.2	0.7	12	0	36
9	0.2	0.8	19	2	37
9	0.25	0.4	29	15	40
9	0.25	0.5	20	0	41
9	0.25	0.6	27	2	50
9	0.25	0.7	17	0	39
9	0.25	0.8	23	0	55
10	0.05	0.4	2	0	3
10	0.05	0.5	2	0	6
10	0.05	0.6	1	0	4
10	0.05	0.7	2	0	4
10	0.05	0.8	1	0	4
10	0.1	0.4	6	0	11
10	0.1	0.5	7	2	12
10	0.1	0.6	8	3	16
10	0.1	0.7	6	0	12
10	0.1	0.8	6	2	13
10	0.15	0.4	6	0	14
10	0.15	0.5	12	1	22
10	0.15	0.6	11	2	21
10	0.15	0.7	8	0	17
10	0.15	0.8	11	5	17
10	0.2	0.4	15	4	28
10	0.2	0.5	15	0	26
10	0.2	0.6	21	0	40
10	0.2	0.7	18	0	37
10	0.2	0.8	19	4	34
10	0.25	0.4	30	22	41
10	0.25	0.5	27	0	36
10	0.25	0.6	30	6	43
10	0.25	0.7	20	3	42
10	0.25	0.8	25	0	39
11	0.05	0.4	3	0	5
11	0.05	0.5	2	0	4
11	0.05	0.6	2	0	4
11	0.05	0.7	1	0	5
11	0.05	0.8	2	0	4

Tabelle A.6: Zwischengespeicherte futuristische Nachrichten

11	0.1	0.4	6	0	13
11	0.1	0.5	8	4	13
11	0.1	0.6	8	2	16
11	0.1	0.7	10	3	15
11	0.1	0.8	9	5	19
11	0.15	0.4	11	2	19
11	0.15	0.5	12	1	21
11	0.15	0.6	9	1	15
11	0.15	0.7	10	4	17
11	0.15	0.8	11	3	18
11	0.2	0.4	23	10	31
11	0.2	0.5	16	0	28
11	0.2	0.6	17	1	23
11	0.2	0.7	18	0	34
11	0.2	0.8	20	8	28
11	0.25	0.4	31	25	38
11	0.25	0.5	26	7	39
11	0.25	0.6	27	13	39
11	0.25	0.7	35	24	48
11	0.25	0.8	28	2	49
12	0.05	0.4	2	1	4
12	0.05	0.5	2	0	4
12	0.05	0.6	2	0	5
12	0.05	0.7	2	0	3
12	0.05	0.8	2	0	3
12	0.1	0.4	6	0	11
12	0.1	0.5	8	3	13
12	0.1	0.6	8	5	11
12	0.1	0.7	9	1	19
12	0.1	0.8	8	1	12
12	0.15	0.4	16	9	24
12	0.15	0.5	13	6	22
12	0.15	0.6	11	4	19
12	0.15	0.7	11	5	20
12	0.15	0.8	11	6	17
12	0.2	0.4	26	16	35
12	0.2	0.5	18	0	28
12	0.2	0.6	21	9	30
12	0.2	0.7	21	6	37
12	0.2	0.8	18	5	25
12	0.25	0.4	33	26	38
12	0.25	0.5	34	19	56
12	0.25	0.6	34	23	39
12	0.25	0.7	32	20	42
12	0.25	0.8	30	3	44
13	0.05	0.4	2	1	4
13	0.05	0.5	2	0	5
13	0.05	0.6	3	1	5
13	0.05	0.7	2	0	2
13	0.05	0.8	2	1	2
13	0.1	0.4	8	2	14
13	0.1	0.5	9	6	12
13	0.1	0.6	8	4	11
13	0.1	0.7	8	5	13
13	0.1	0.8	8	4	12
13	0.15	0.4	12	8	22
13	0.15	0.5	14	7	23
13	0.15	0.6	12	7	20
13	0.15	0.7	12	5	22
13	0.15	0.8	12	6	21
13	0.2	0.4	24	11	33
13	0.2	0.5	22	10	28
13	0.2	0.6	23	19	26
13	0.2	0.7	21	5	32
13	0.2	0.8	21	13	29
13	0.25	0.4	35	26	39
13	0.25	0.5	36	26	48
13	0.25	0.6	32	22	41
13	0.25	0.7	28	16	39
13	0.25	0.8	33	22	45
14	0.05	0.4	2	1	4
14	0.05	0.5	3	1	5
14	0.05	0.6	2	1	3
14	0.05	0.7	2	1	3
14	0.05	0.8	1	0	2
14	0.1	0.4	8	3	13
14	0.1	0.5	9	6	13
14	0.1	0.6	8	6	11
14	0.1	0.7	8	5	12
14	0.1	0.8	8	5	12
14	0.15	0.4	14	9	24
14	0.15	0.5	15	11	23
14	0.15	0.6	13	8	19
14	0.15	0.7	12	7	19
14	0.15	0.8	12	7	20
14	0.2	0.4	26	18	33
14	0.2	0.5	21	7	28
14	0.2	0.6	24	18	30
14	0.2	0.7	22	8	34
14	0.2	0.8	24	13	38
14	0.25	0.4	36	26	43
14	0.25	0.5	36	29	44
14	0.25	0.6	35	23	42
14	0.25	0.7	29	22	37

Tabelle A.6: Zwischengespeicherte futuristische Nachrichten

14	0.25	0.8	34	24	44
15	0.05	0.4	2	1	4
15	0.05	0.5	2	1	5
15	0.05	0.6	2	1	3
15	0.05	0.7	2	1	4
15	0.05	0.8	2	1	3
15	0.1	0.4	9	5	12
15	0.1	0.5	9	6	13
15	0.1	0.6	8	6	11
15	0.1	0.7	8	5	12
15	0.1	0.8	7	4	11
15	0.15	0.4	15	10	23
15	0.15	0.5	14	10	23
15	0.15	0.6	14	8	20
15	0.15	0.7	13	8	19
15	0.15	0.8	12	7	19
15	0.2	0.4	26	17	33
15	0.2	0.5	27	19	41
15	0.2	0.6	26	19	35
15	0.2	0.7	21	10	26
15	0.2	0.8	20	14	25
15	0.25	0.4	36	26	42
15	0.25	0.5	36	30	44
15	0.25	0.6	35	29	42
15	0.25	0.7	30	23	38
15	0.25	0.8	31	22	43
16	0.05	0.4	3	1	4
16	0.05	0.5	2	1	5
16	0.05	0.6	2	1	3
16	0.05	0.7	2	1	3
16	0.05	0.8	1	1	2
16	0.1	0.4	9	5	11
16	0.1	0.5	9	6	12
16	0.1	0.6	8	6	11
16	0.1	0.7	8	5	11
16	0.1	0.8	7	4	12
16	0.15	0.4	15	10	22
16	0.15	0.5	14	10	23
16	0.15	0.6	14	8	20
16	0.15	0.7	13	8	20
16	0.15	0.8	11	8	15
16	0.2	0.4	26	18	33
16	0.2	0.5	24	17	30
16	0.2	0.6	26	19	32
16	0.2	0.7	23	17	30
16	0.2	0.8	19	14	29
16	0.25	0.4	36	26	41
16	0.25	0.5	36	30	44
16	0.25	0.6	35	29	42
16	0.25	0.7	31	24	38
16	0.25	0.8	30	22	37
17	0.05	0.4	2	0	4
17	0.05	0.5	2	1	5
17	0.05	0.6	2	1	3
17	0.05	0.7	2	1	3
17	0.05	0.8	1	1	2
17	0.1	0.4	9	5	12
17	0.1	0.5	9	6	13
17	0.1	0.6	8	6	11
17	0.1	0.7	9	5	12
17	0.1	0.8	7	4	9
17	0.15	0.4	15	10	24
17	0.15	0.5	15	11	23
17	0.15	0.6	14	8	20
17	0.15	0.7	13	8	20
17	0.15	0.8	11	8	15
17	0.2	0.4	26	18	33
17	0.2	0.5	25	19	31
17	0.2	0.6	26	19	31
17	0.2	0.7	23	17	30
17	0.2	0.8	20	14	25
17	0.25	0.4	37	26	43
17	0.25	0.5	37	30	44
17	0.25	0.6	36	29	42
17	0.25	0.7	31	24	38
17	0.25	0.8	28	22	37
18	0.05	0.4	3	1	4
18	0.05	0.5	2	1	5
18	0.05	0.6	2	1	3
18	0.05	0.7	2	1	3
18	0.05	0.8	1	1	2
18	0.1	0.4	9	5	12
18	0.1	0.5	9	6	13
18	0.1	0.6	8	6	11
18	0.1	0.7	9	5	12
18	0.1	0.8	7	4	10
18	0.15	0.4	15	10	24
18	0.15	0.5	15	11	23
18	0.15	0.6	14	8	20
18	0.15	0.7	13	8	20
18	0.15	0.8	11	8	15
18	0.2	0.4	27	18	33
18	0.2	0.5	26	19	32
18	0.2	0.6	26	19	33

Tabelle A.6: Zwischengespeicherte futuristische Nachrichten

18	0.2	0.7	23	17	29
18	0.2	0.8	18	14	25
18	0.25	0.4	37	26	43
18	0.25	0.5	37	30	44
18	0.25	0.6	36	29	42
18	0.25	0.7	31	25	39
18	0.25	0.8	28	22	36
19	0.05	0.4	2	1	4
19	0.05	0.5	2	2	3
19	0.05	0.6	2	1	3
19	0.05	0.7	2	1	2
19	0.05	0.8	1	1	2
19	0.1	0.4	9	5	11
19	0.1	0.5	9	6	12
19	0.1	0.6	8	6	11
19	0.1	0.7	9	5	12
19	0.1	0.8	7	4	9
19	0.15	0.4	16	10	24
19	0.15	0.5	15	11	23
19	0.15	0.6	15	8	20
19	0.15	0.7	13	8	20
19	0.15	0.8	11	8	15
19	0.2	0.4	27	18	33
19	0.2	0.5	25	19	32
19	0.2	0.6	26	19	33
19	0.2	0.7	23	17	30
19	0.2	0.8	18	14	21
19	0.25	0.4	38	26	43
19	0.25	0.5	38	30	44
19	0.25	0.6	37	29	42
19	0.25	0.7	33	25	39
19	0.25	0.8	29	22	36
20	0.05	0.4	2	1	4
20	0.05	0.5	2	2	3
20	0.05	0.6	2	1	3
20	0.05	0.7	2	1	2
20	0.05	0.8	2	1	2
20	0.1	0.4	9	5	11
20	0.1	0.5	9	7	12
20	0.1	0.6	8	6	11
20	0.1	0.7	9	5	12
20	0.1	0.8	7	4	10
20	0.15	0.4	16	10	24
20	0.15	0.5	15	11	23
20	0.15	0.6	14	8	20
20	0.15	0.7	14	8	20
20	0.15	0.8	11	9	15
20	0.2	0.4	26	18	33
20	0.2	0.5	26	19	32
20	0.2	0.6	25	19	33
20	0.2	0.7	24	17	30
20	0.2	0.8	18	14	20
20	0.25	0.4	38	26	43
20	0.25	0.5	38	30	44
20	0.25	0.6	37	29	42
20	0.25	0.7	33	25	39
20	0.25	0.8	29	22	36
21	0.05	0.4	3	1	4
21	0.05	0.5	2	2	2
21	0.05	0.6	2	1	3
21	0.05	0.7	2	1	2
21	0.05	0.8	2	1	2
21	0.1	0.4	9	5	11
21	0.1	0.5	10	5	12
21	0.1	0.6	9	7	11
21	0.1	0.7	10	5	12
21	0.1	0.8	8	4	10
21	0.15	0.4	15	10	19
21	0.15	0.5	14	11	19
21	0.15	0.6	13	8	16
21	0.15	0.7	13	8	17
21	0.15	0.8	11	9	12
21	0.2	0.4	29	26	33
21	0.2	0.5	27	24	32
21	0.2	0.6	27	24	33
21	0.2	0.7	25	21	30
21	0.2	0.8	18	14	20
21	0.25	0.4	35	26	40
21	0.25	0.5	36	30	41
21	0.25	0.6	35	29	41
21	0.25	0.7	31	25	36
21	0.25	0.8	29	22	36
22	0.05	0.4	2	1	4
22	0.05	0.5	2	2	2
22	0.05	0.6	2	1	3
22	0.05	0.7	2	1	2
22	0.05	0.8	1	1	2
22	0.1	0.4	9	5	11
22	0.1	0.5	10	7	12
22	0.1	0.6	9	7	10
22	0.1	0.7	9	5	12
22	0.1	0.8	7	3	10
22	0.15	0.4	13	10	17
22	0.15	0.5	14	11	19

Tabelle A.6: Zwischengespeicherte futuristische Nachrichten

22	0.15	0.6	13	8	16
22	0.15	0.7	12	8	17
22	0.15	0.8	10	9	11
22	0.2	0.4	29	26	33
22	0.2	0.5	26	24	28
22	0.2	0.6	28	24	33
22	0.2	0.7	26	24	30
22	0.2	0.8	17	14	20
22	0.25	0.4	34	26	40
22	0.25	0.5	36	30	41
22	0.25	0.6	33	29	38
22	0.25	0.7	29	25	34
22	0.25	0.8	27	22	33
23	0.05	0.4	2	2	2
23	0.05	0.5	2	2	2
23	0.05	0.6	2	2	2
23	0.05	0.7	2	2	2
23	0.05	0.8	1	1	1
23	0.1	0.4	10	10	10
23	0.1	0.5	10	10	10
23	0.1	0.6	9	9	9
23	0.1	0.7	10	10	10
23	0.1	0.8	8	8	8
23	0.15	0.4	17	17	17
23	0.15	0.5	19	19	19
23	0.15	0.6	16	16	16
23	0.15	0.7	17	17	17
23	0.15	0.8	9	9	9
23	0.2	0.4	26	26	26
23	0.2	0.5	24	24	24
23	0.2	0.6	26	26	26
23	0.2	0.7	24	24	24
23	0.2	0.8	14	14	14
23	0.25	0.4	26	26	26
23	0.25	0.5	30	30	30
23	0.25	0.6	29	29	29
23	0.25	0.7	25	25	25
23	0.25	0.8	22	22	22

Tabelle A.6: Zwischengespeicherte futuristische Nachrichten

Literaturverzeichnis

[All83] James F. Allen, *Maintaining knowledge about temporal intervals*, Commun. ACM **26** (1983), no. 11, 832–843. 49

[BCM+99] Guruduth Banavar, Tushar Deepak Chandra, Bodhi Mukherjee, Jay Nagarajarao, Robert E. Strom, and Daniel C. Sturman, *An efficient multicast protocol for content-based publish-subscribe systems*, Proceedings of the 19th International Conference on Distributed Computing Systems (ICDCS 1999) (Austin, Texas, USA), May 1999, pp. 262–272. 8

[BFI+07] R. Boivie, N. Feldman, IBM, Y. Imai, WIDE / Fujitsu, W. Livens, ESCAUX, D. Ooms, and OneSparrow, *Explicit multicast (Xcast) concepts and options*, RFC5058, November 2007, Category: Experimental. 5

[Cis06] Cisco Systems, Inc., *Internetworking technology handbook*, Cisco.com, 2006, Chapter 43, Internet Protocol Multicast. 5

[CNF01] Gianpaolo Cugola, Elisabetta Di Nitto, and Alfonso Fuggetta, *The JEDI event-based infrastructure and its application to the development of the OPSS WFMS*, IEEE Transactions on Software Engineering **27** (2001), no. 9, 827–850. 7

[CRW99] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf, *Interfaces and algorithms for a wide-area event notification service*, Tech. Report CU-CS-888-99, Department of Computer Science, University of Colorado, October 1999, revised May 2000. 8, 9, 10, 12, 14, 16

[CRW00] A. Carzaniga, D. Rosenblum, and A. Wolf, *Content-based addressing and routing: A general model and its application*, Tech. Report CU-CS-902-00, Department of Computer Science, University of Colorado, January 2000. 8, 9, 12, 16

[CS02] Edith Cohen and Scott Shenker, *Replication strategies in unstructured peer-to-peer networks*, SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications (New York, NY, USA), ACM, 2002, pp. 177–190. 111

[CW03] Antonio Carzaniga and Alexander L. Wolf, *Forwarding in a content-based network*, Proceedings of ACM SIGCOMM 2003 (Karlsruhe, Germany), Department of Computer Science, University of Colorado, August 2003, pp. 163–174. 16

[DA99] Anind K. Dey and Gregory D. Abowd, *Towards a better understanding of context and context-awareness*, Georgia Institute of Technology, Atlanta, GA, USA, 1999. 19, 29

[DC90] Stephen E. Deering and David R. Cheriton, *Multicast routing in datagram internetworks and extended LANs*, ACM Trans. Comput. Syst. **8** (1990), no. 2, 85–110. 5, 6

[Dee89] S. Deering, *Host extensions for IP multicasting*, RFC1112, Network Working Group, August 1989, Obsoletes: RFCs 988, 1054. 5

[Dep97] Department of Defense, *World geodetic system 1984, its definition and relationships with local geodetic systems*, Tech. Report TR8350.2, July 1997, Third Edition. 19

[DO03] D. Doval and D. O'Mahony, *Overlay networks: A scalable alternative for P2P*, Internet Computing, IEEE **7** (2003), no. 4, 79–82. 3, 4

[DR03] F. Dürr and K. Rothermel, *On a location model for fine-grained Geocast*, UbiComp2003: Ubiquitous Computing, 5th International Conference, 2003, pp. 18–35. 21

[Fen97] W. Fenner, *Internet Group Management Protocol, version 2*, RFC2236, Network Working Group, November 1997, Updates: 1112. 5

[Fid06] E. Fidler, *PADRES: A distributed content-based publish/subscribe system*, Master's thesis, University of Toronto, Graduate Department of Electrical & Computer Engineering, 2006. 8

[FJLM05] E. Fidler, Hans-Arno Jacobsen, Guoli Li, and Serge Mankovski, *The PADRES distributed publish/subscribe system*, FIW (Stephan Reiff-Marganiec and Mark Ryan, eds.), IOS Press, december 2005, pp. 12–30. 8

[FKP02] A. Fabrikant, E. Koutsoupias, and C. Papadimitriou, *Heuristically optimized tradeoffs: A new paradigm for power laws in the internet*, ICALP '02: Proceedings of the 29th International Colloquium on Automata, Languages and Programming (London, UK), 34th Symposium on Theory of Computing, Springer-Verlag, 2002, pp. 110–122. 93

[FM00] Ludger Fiege and Gero Muhl, *Rebeca Event-Based Electronic Commerce Architecture*, http://www.gkec.informatik.tu-darmstadt.de/rebeca, 2000. 8

[Gei07] Dilp. Inf. Lars Geiger, *Konzipierung und Realisierung eines kontextbezogenen Kommunikationsmechanismus auf der Basis von inhaltsbezogenen Vermittlungsalgorithmen*, Diplomarbeit, IPVS-VS, Universität Stuttgart, April 2007. 19, 20, 23, 25, 26, 28, 29, 44, 91

[HD98] R. Hinden and S. Deering, *IPv6 multicast address assignments*, RFC2375, Network Working Group, July 1998, Category: Informational. 9

[Jes05] Gian Paolo Jesi, *PeerSim HOWTO: Build a new protocol for the PeerSim 1.0 simulator*, December 2005. 91

[Jes06] _____, *PeerSim HOWTO: Build a topology generator for PeerSim 1.0*, April 2006. 93

[JMJV06] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris, *PeerSim: A peer-to-peer simulator*, November 2006. 91

[KE06] Alfons Kemper and Andre Eickler, *Datenbanksysteme. Eine Einführung*, 6 ed., Oldenbourg, March 2006. 41

[KN02] G. Klyne and C. Newman, *Date and time on the internet: Timestamps*, RFC3339, 2002. 51

[KR95] Balachander Krishnamurthy and David S. Rosenblum, *Yeast: A General Purpose Event-Action System*, IEEE Transactions on Software Engineering **21** (1995), no. 10, 845–857. 7

[LCH+07] G. Li, A. Cheung, Sh. Hou, S. Hu, V. Muthusamy, R. Sherafat, A. Wun, H.-A. Jacobsen, and S. Manovski, *Historic data access in publish/subscribe*, DEBS '07: Proceedings of the 2007 inaugural international conference on Distributed event-based systems (New York, NY, USA), Middleware System Research Group, University of Toronto, Canada, ACM, June 2007, pp. 80–84. 17, 18

[LCP+05] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim, *A survey and comparison of peer-to-peer overlay network schemes*, IEEE Communications Surveys & Tutorials **7** (2005), no. 2, 72–93. 4

[LHJ05] Guoli Li, Shuang Hou, and Hans-Arno Jacobsen, *A unified approach to routing, covering and merging in publish/subscribe systems based on Modified Binary Decision Diagrams*, ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (Washington, DC, USA), IEEE Computer Society, 2005, pp. 447–457. 16

[LJ05] Guoli Li and Hans-Arno Jacobsen, *Composite subscriptions in content-based publish/subscribe systems*, Middleware (Gustavo Alonso, ed.), vol. 3790, Springer, december 2005, pp. 249–269. 18

[MFB02] Gero Mühl, Ludger Fiege, and Alejandro P. Buchmann, *Filter similarities in content-based publish/subscribe systems*, ARCS '02: Proceedings of the International Conference on Architecture of Computing Systems (London, UK), Department of Computer Science, Darmstadt University of Technologies, Springer-Verlag, 2002, pp. 224–240. 16, 81

[Mil92] David L. Mills, *Network Time Protocol (version 3) - specification, implementation and analysis*, RFC1305, Network Working Group, March 1992, Obsoletes RFC-1119, RFC-1059, RFC-958. 27

[Mil96] D. Mills, *Simple Network Time Protocol (SNTP) version 4 for IPv4, IPv6 and OSI*, RFC2030, Network Working Group, October 1996, Category: Informational. 27

[MNR02] Dahlia Malkhi, Moni Naor, and David Ratajczak, *Viceroy: A scalable and dynamic emulation of the butterfly*, Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC 2002) (Monterey, California, USA), July 2002, pp. 183–192. 4

[NI97] Julio C. Navas and Tomasz Imielinski, *GeoCast – geographic addressing and routing*, Mobile Computing and Networking, Computer Science Department, Rutgers, The State University, 1997, pp. 66–76. 20, 21

[Obj04] Object Management Group, Inc., *Event Service Specification*, 1.2 ed., October 2004, formal/04-10-02. 8

[Rat02] Sylvia Ratnasamy, *A scalable content-addressable network*, Ph.D. thesis, University of California at Berkeley, 2002, 15–31. 4

[SAB+00] B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps, *Content based routing with Elvin4*, AUUG2K, June 2000. 8

[Sch07] Ronald Schertle, *Lastsensitive Modifikation von Peer-to-Peer Indexstrukturen*, Studienarbeit 2106, October 2007. 4

[SK03] Alexandra Schäfer and Lars Karg, *REBECA - an Event-Based Electronic Commerce Architecture*, Technische Universität Darmstadt, Seminararbeit, Juli 2003. 8

[SKPK01] Myung-Ki Shin, Yong-Jin Kim, Ki-Shik Park, and Sang-Ha Kim, *Explicit Multicast Extension (Xcast+) for efficient multicast packet delivery*, ETRI Journal **23** (2001), 202–204. 6

[SMK+01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, *Chord: A scalable peer-to-peer lookup service for internet applications*, Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2001) (San Diego, California, USA), August 2001. 4

[Sof08] Softwired Inc., *iBus: The Java Intranet Software Bus*, http://www.softwired.ch/products/message/messagebus.html, 2008. 8

[Sun98] Sun Microsystems, Inc., Mountain View CA, U.S.A., *Java Distributed Event Specification*, 1998. 8

[Tan03] Andrew S. Tanenbaum, *Computernetzwerke*, vol. 4, Pearson Studium, 2003. 3, 4, 7, 15, 24

[WBS+05] Benjamin Weyl, Pedro Brandão, Antonio F. Gómez Skarmeta, Rafael Marin Lopez, Parijat Mishra, Christian Hauser, and Holger Ziemek, *Protecting privacy of identities in federated operator environments*, Information Society Technologies: 14th IST Mobile & Wireless Communications Summit (Dresden, Germany), June 2005. 46

[WH98] Mike Wray and Rycharde Hawkes, *Distributed virtual environments and VRML: an event-based architecture*, WWW7: Proceedings of the seventh international conference on World Wide Web 7 (Amsterdam, The Netherlands), Elsevier Science Publishers B. V., 1998, pp. 43–51. 8

[Zei04] Andreas Zeidler, *A distributed publish/subscribe notification service for pervasive environments*, Ph.D. thesis, Technischen Universität Darmstadt, 2004, Chapter 7: Decoupling in Space and Time, pp. 115–130. 17

[Zim80] Hubert Zimmermann, *OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection*, IEEE Transactions on Communications, vol. 28, April 1980, pp. 425–432. 3

Erklärung

Hiermit versichere ich, diese Arbeit
selbständig verfasst und nur die
angegebenen Quellen benutzt zu haben.

Stuttgart, den 7. Juli 2008

(Ronald Schertle)