Freie Universität Berlin
Fachbereich Mathematik und Informatik
Institut für Informatik

Diplomarbeit

Entwurf und Analyse eines effizienten verteilten Benachrichtigungssystems

von

Sven Bittner,
bittner@inf.fu-berlin.de

Betreuung:

Prof. Dr. Heinz F. Schweppe und
Dr. Annika Hinze

Berlin, den 8. September 2003

Danksagung

Ich möchte an dieser Stelle den Personen danken, die durch ihre Unterstützung zu dieser Arbeit beigetragen haben:

Dr. Annika Hinze für ihre hervorragende Betreuung, die viele Zeit, die sie investiert hat, die Diskussionen und Ratschläge,

Prof. Dr. Heinz F. Schweppe, der es mir ermöglichte diese Arbeit in seiner Arbeitsgruppe anzufertigen,

meiner Freundin Doris für die Geduld und Nachsicht, die sie mit mir hatte, sowie für die konstruktive Kritik beim Lesen meiner Arbeit,

Karsten Heinrich für Einführung und Anmerkungen zum Thema Gebäudesteuerung,

der Arbeitsgruppe Datenbanken und Informationssysteme für hilfreiche Hinweise,

Margot Jung und Carsten Schultz für die Durchsicht dieser Arbeit

und meinen Eltern für die Unterstützung während meines Studiums.

Kurzfassung

Das in Benachrichtigungssystemen eingesetzte Abonnentenprinzip steht im Gegensatz zu der heutzutage im Internet angewandten Kommunikationsform, dem Anfrage-Antwort-Prinzip. Bei diesem Abonnentenprinzip wird eine einmalige Anfrage eines jeden Abonnenten, ein sog. Profil, an das Benachrichtigungssystem gesendet. Informationen der Anbieter, bezeichnet als Ereignisse, werden ebenfalls zum Benachrichtigungssystem geschickt. Dieses ordnet Ereignisse passenden Profilen zu und benachrichtigt die Abonnenten über alle Ereignisse, die ihre Profile erfüllen. Diese Vorgänge werden Filterung und Benachrichtigung genannt. Insbesondere bei Realzeitsystemen wie der Gebäudesteuerung ist die Effizienz der eingesetzten Filterkomponente eines der wichtigsten Entwurfskriterien.

Im Rahmen dieser Arbeit wird eine solche Filterkomponente entwickelt, analysiert und bewertet. Zur effizienten Bedienung vieler Informationsanbieter und Abonnenten wird diese Filterkomponente als verteiltes System realisiert. Dazu werden vorhandene Lösungsansätze aus der Literatur untersucht, miteinander verglichen und für die Umsetzung in einer Implementierung ausgewählt. Im praktischen Teil werden neue Verfahren zum Einsatz in der verteilten Filterung vorgeschlagen und der Prototyp DAS mit drei unterschiedlichen Filteralgorithmen beschrieben. Mit diesem Prototyp werden zahlreiche Experimente bzgl. der Effizienz, Netzlast und Speicherauslastung durchgeführt. Die Filteralgorithmen werden dadurch unter verschiedenen Einsatzbedingungen analysiert und bewertet sowie ihre optimalen Einsatz-Szenarien gefunden.

Abstract

Event Notification Services use a different principle than that which is applied to the Internet, whereas the former make use of the publish/subscribe method the latter works with the principle of request/reply. The publish/subscribe method entails that each subscriber subscribes to the Event Notification Service in order to repeatedly receive information about certain facts, which it defines in a so-called profile. This information, which is referred to by the notion of an event, is offered to the Event Notification Service by publishers. Such events are matched against the subscribed profiles by the Event Notification Service, which notifies subscribers when matching events have occurred. These methods are called filtering and notification. In real-time systems such as facility management the efficiency of the applied filter component is one of the most important design goals.

In this thesis a filter component, which accomplishes the above goal, is developed, analysed and evaluated. It is realized as a distributed system to efficiently manage a great number of publishers and subscribers. Existing approaches are analysed, compared with each other and some are chosen for an implementation. The practical part of this thesis consists of a proposal of new methods, which may be employed in distributed filtering. Also a description of the prototype DAS including three dissimilar filter algorithms is presented. This prototype is tested with respect to efficiency, network traffic and memory consumption and so the implemented filter algorithms are analysed and evaluated considering various system loads. It is thus optimal application scenarios can be found.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Inhalt

Mit der ständig wachsenden Popularität des Internets kann auf immer größere Informationsmengen zugegriffen werden. Ein Hauptproblem für den Anwender ist dabei das Finden relevanter Informationen aus den vorhandenen Wissensmengen. Das Internet beruht hauptsächlich auf dem Anfrage-Antwort-Prinzip (Request/Response): Nutzer fragen bei einer Informationsquelle gemäß ihrer Interessen an und erhalten daraufhin eine Antwort bezüglich der angefragten Informationen. Dieses Prinzip ist die Grundidee des WWW und des damit verbundenen HTTP-Protokolls [36]: Nach der Anfrage an einen Webserver erhält man die angeforderte Webseite zurück. Als Erleichterung für den Anwender und zur Verbesserung der Informationssuche existieren Suchmaschinen (z.B. Google [4]), welche eine Vielzahl von Webseiten durchsuchen und dem Nutzer die Möglichkeit geben nach Stichworten zu suchen. Auch solche Suchmaschinen [4] beruhen auf dem Anfrage-Antwort-Prinzip. Der Datenfluss ist in diesen Fällen durch den Anwender initiiert (Pull Mode): Auf jede Anfrage erhält der Nutzer genau eine Antwort. Neue Informationen seitens der Informationsanbieter gelangen jedoch nicht zu den Nutzern, es sei denn eine erneute Anfrage wird gestellt.

Ein anderer, den Nutzerwünschen zugewandterer Ansatz, ist das Abonnentenprinzip. Interessenten stellen einmalig eine Anfrage bei einem Informationsanbieter und werden danach ständig über Neuerungen und Änderungen informiert. Der Datenfluss wird als fremdbestimmt (Push Mode) bezeichnet, da eine Informationsquelle über den Versand von Informationen entscheidet (anhand der Vorgaben der Interessenten). In recht einfacher Form wird diese Technik in Newsgroups (umgesetzt durch das Protokoll NNTP [34]) genutzt. Dabei kann sich ein Nutzer für bestimmte Themenbereiche anmelden und erhält danach ständig Nachrichten der gewählten Bereiche. Eine Verfeinerung innerhalb der Bereiche, z.B. durch Angabe von Stichworten, ist jedoch nicht möglich.

Benachrichtigungssysteme können diese Probleme der Interessenbeschreibung seitens der Nutzer, der ständigen Information der Nutzer über Neuerungen und der Verbreitung von Informationen lösen. Sie treten als Zwischensystem zwischen Informationsanbietern und Interessenten auf. Informationen der Anbieter werden vom Benachrichtigungssystem einer inhaltsbasierten Filterung unterzogen. Entsprechend den Nutzerspezifikationen werden diese Informationen an interessierte Nutzer weitergeleitet.

Der Entwurf und die Analyse der Filterkomponente eines solchen Benachrichtigungssystems sind Inhalt dieser Diplomarbeit. Zur Bedienung vieler Informationsanbieter und Interessenten muss die Filterkomponente als verteiltes System umgesetzt werden [24]. Es existieren einige Ansätze zur Realisierung einer verteilten Filterung [2, 5, 27, 30, 38]. Diese werden im theoretischen Teil der Arbeit analysiert und bewertet. Aufbauend auf diesen speziellen Filterstrategien wird danach eine allgemeine Klassifikation verteilter Filteralgorithmen erarbeitet und vorgestellt. Im praktischen Teil der Arbeit werden die wichtigsten dieser verteilten Filteralgorithmen implementiert. Dabei entsteht der Prototyp DAS eines verteilten Benachrichtigungssystems. Mit diesem Prototyp werden zahlreiche Experimente bezüglich Filtereffizienz, Netzlast und Speicherverbrauch durchgeführt. Anhand der Ergebnisse der Experimente erfolgt eine genaue Bewertung der verschiedenen Filteralgorithmen und das Aufzeigen der jeweiligen Vor- und Nachteile. Im Folgenden wird zuerst eine Einführung in Benachrichtigungssysteme gegeben, unter Bezug auf das für diese Arbeit gewählte Anwendungsszenario der Gebäudesteuerung. Danach folgen allgemeine Definitionen der für diese Arbeit wichtigsten Begriffe aus dem Kontext von Benachrichtigungssystemen.

1.1 Einführung und Anwendung

Benachrichtigungssysteme werden heutzutage in einer Vielzahl von Anwendungsgebieten eingesetzt, wie beispielsweise Digitale Bibliotheken, Verkehrssteuerung/-überwachung, Medizin und Informationsdienste im Internet. Ihre Aufgaben reichen dabei von der Verteilung elektronischer Veröffentlichungen und Artikel (Hermes [7]) über die medizinische Therapieplanung (PLAN [37]) und die Steuerung des Verkehrsflusses beim Luftverkehr [19] bis hin zur Überwachung von Webdokumenten (Conquer [20]). Im Rahmen dieser Diplomarbeit wird ein anderes Einsatzgebiet betrachtet, das immer mehr an Bedeutung gewinnt: die Gebäudesteuerung und -verwaltung.

Aufgaben in Gebäuden, zu deren Verwaltung ein Benachrichtigungsdienst eingesetzt werden kann, sind: die Steuerung von Jalousien und Klimaanlagen bzw. Heizungen, die Verständigung von Dienstleistern bei Störungen wie Lampenausfall oder Glasbruch und die automatische Anpassung der Helligkeit von Räumen an die optimalen Lichtverhältnisse für einen Bildschirmarbeitsplatz. Allerdings können durch das Benachrichtigungssystem auch Aufgaben wie das Einschalten der Beleuchtung auf Knopfdruck übernommen werden.

Informationsanbieter in Gebäuden sind beispielsweise Sensoren, Messfühler und Schalter (so genannte Aktoren) [18]. Sie senden periodisch oder bei Zustandsänderungen ihren aktuellen Zustand als *Ereignisse* an das Benachrichtigungssystem. Interessenten sind die Gewerke des Gebäudes, z.B. Jalousien, Leuchtmittel oder Heizungen, aber auch ein Hausmeister oder die Wachgesellschaft kann als Interessent auftreten [18]. Zur Konfiguration kann eine Installateurin *Profile* anlegen. Die typi-

sche Funktionalität einer Alarmanlage wird durch dieses Beispielprofil beschrieben:

> Informiere die Wachgesellschaft, falls Glasbruch auftritt und danach ein Bewegungssensor eine Person im Gebäude meldet.

Zur Steuerung einer Heizung H_1 ist folgendes vorstellbar:

> Schalte die Heizung H_1 ab, falls die Außentemperatur über 20 °C steigt.

Die Außentemperatur wird von einem Temperatursensor S_1 gemessen, der bei Temperaturänderungen die neue Temperatur an das Benachrichtigungssystem sendet. Die Heizung kann daraufhin bei Temperaturen über 20 °C abschalten. Dieses einfache Beispielprofil wird im weiteren Verlauf der Arbeit erneut aufgegriffen und weiter verfeinert.

Heutzutage wird in der Gebäudesteuerung ein dezentraler Ansatz verfolgt: Logikbausteine zur Gerätesteuerung werden hauptsächlich in Endgeräte integriert. Deren Eingabesignale sind in einer hierarchischen Struktur über ein Bussystem wie EIB (European Installation Bus) [6], Luxmate [22] oder LON (Local Operating Network) [21] vernetzt. Die Programmierung und Verknüpfung der Endgeräte findet dabei direkt in den Geräten statt, der Bus ist lediglich für die Übertragung der Signale verantwortlich. Teilweise werden Verknüpfungen nicht in Endgeräten durchgeführt, sondern in separierten Logikbausteinen. Dadurch können komplexere Verknüpfungen ausgeführt werden, auch wenn ein Endgerät diese nicht unterstützt.

Wichtige Eigenschaften zur Bewertung einer Gebäudeverwaltungsstrategie sind: die Änderbarkeit der aktuellen Systemkonfiguration, Möglichkeiten zur Geräteauswahl, Möglichkeiten zur Programmierung der Endgeräte (also die Ausdrucksstärke der Programmiersprache) und die Skalierbarkeit des Systems. Der heutige dezentrale Ansatz mit integrierter Logik birgt mehrere Nachteile:

- Änderbarkeit: Beim Umkonfigurieren der Endgeräte muss zum einen die Logik des Gerätes eine Umprogrammierung unterstützen, zum anderen müssen alle betroffenen Geräte direkt von einem Techniker umprogrammiert werden.

- Geräteauswahl: Die Auswahl von Endgeräten muss hauptsächlich aufgrund ihrer technischen Möglichkeiten geschehen. Das für einen Kunden ebenfalls wichtige Kriterium des Designs muss eine untergeordnete Rolle spielen. Weiterhin können teilweise nur Geräte bestimmter Anbieter miteinander gekoppelt werden, da sich einige Funktionalitäten nicht beliebig miteinander verbinden lassen.

- Ausdrucksstärke: Die Möglichkeiten zur Programmierung von Endgeräten sind durch deren integrierte oder separierte Logikbausteine begrenzt. Spezialanfertigungen können umfangreiche und komplizierte Lösungen bieten, sind dadurch jedoch teuer in der Herstellung und lediglich spezialisiert einsetzbar.

- Skalierbarkeit: Die heute eingesetzten Bussysteme können nur eine begrenzte Anzahl an Endgeräten und Aktoren miteinander koppeln. Der Einsatz in großen Gebäuden, beispielsweise den Treptowers in Berlin-Treptow-Köpenick, wird dadurch schwer möglich. Deshalb werden verschiedene parallele Busse benötigt und eingesetzt, um alle Geräte integrieren zu können.

Der Einsatz eines Benachrichtigungssystems zur Umsetzung sämtlicher Steuerungslogik, in Verbindung mit einem beliebigen Verbindungsnetz für Aktoren und Endgeräte, kann diese Probleme lösen:

- Änderbarkeit: Der Einsatz eines Benachrichtigungssystems macht es einfach, umfangreiche Konfigurationsänderungen durchzuführen, da über eine einzige Schnittstelle alle Endgeräte programmiert und angepasst werden können.
- Geräteauswahl: Endgeräte verfügen über keine Logikbausteine mehr. Dadurch können die Geräte aller Anbieter frei kombiniert werden. Eine Abwägung zwischen technischen Möglichkeiten und anderen Aspekten wie beispielsweise dem Design ist nicht mehr erforderlich.
- Ausdrucksstärke: Die Möglichkeiten zur Programmierung der Endgeräte eines Gebäudes sind lediglich durch das Benachrichtigungssystem selbst beschränkt. Dadurch können alle Geräte gleichartige Verknüpfungen abbilden.
- Skalierbarkeit: Die Skalierbarkeitseigenschaften des Benachrichtigungssystems können auf die mögliche Größe eines Gebäudes übertragen werden, d.h. wie viele Aktoren und Endgeräte das System unterstützt. Weiterhin sind das eingesetzte Verbindungsnetz und dessen genutztes Protokoll für die Skalierbarkeit relevant.

Neben diesen positiven Eigenschaften kann zusätzlich die Verkabelung eines Gebäudes sehr einfach geschehen, da sämtliche Leitungen der Endgeräte in einem Schacht zusammengeführt werden können. Die verschiedenen Teilbereiche wie Klimatechnik, Lichttechnik und Medientechnik sind so miteinander kombinierbar und verschränkt nutzbar, auch wenn deren Installationen von verschiedenen Unternehmen realisiert werden.

1.2 Begriffsdefinitionen

Dieser Abschnitt enthält eine Beschreibung und Erläuterung wichtiger Begriffe aus dem Kontext von Benachrichtigungsdiensten, die in dieser Arbeit verwendet werden. Da in der Literatur sowohl verschiedene Definitionen existieren als die Namensgebung englischsprachig erfolgt, ist die hier angegebene Bedeutung und Definition im Rahmen dieser Arbeit grundlegend. Neben diesen ersten Definitionen werden im Laufe der Arbeit weitere Termini eingeführt.

1.2.1 Benachrichtigungsdienste

Benachrichtigungsdienste sind Zwischensysteme zur Entkoppelung von Informationsquellen und Informationssenken. Informationsquellen werden in diesem Kontext als *Anbieter* oder Ereignisanbieter bezeichnet, Informationssenken als *Abonnenten* bzw. Ereignisabonnenten. Die Informationen werden Ereignisse genannt.

Benachrichtigungsdienste sind für die *Filterung* von Ereignissen der Anbieter zuständig. Weiterhin haben sie die Aufgabe Abonnenten über eingetroffene Ereignisse zu informieren, was auch als *Benachrichtigung* bezeichnet wird. Dazu spezifizieren Abonnenten ihre Interessen durch Profile.

Eine erste Übersicht eines Benachrichtigungssystems im Kontext der Gebäudesteuerung ist in Abb. 1.1 dargestellt. Rechts sind die Abonnenten, eine Heizung sowie

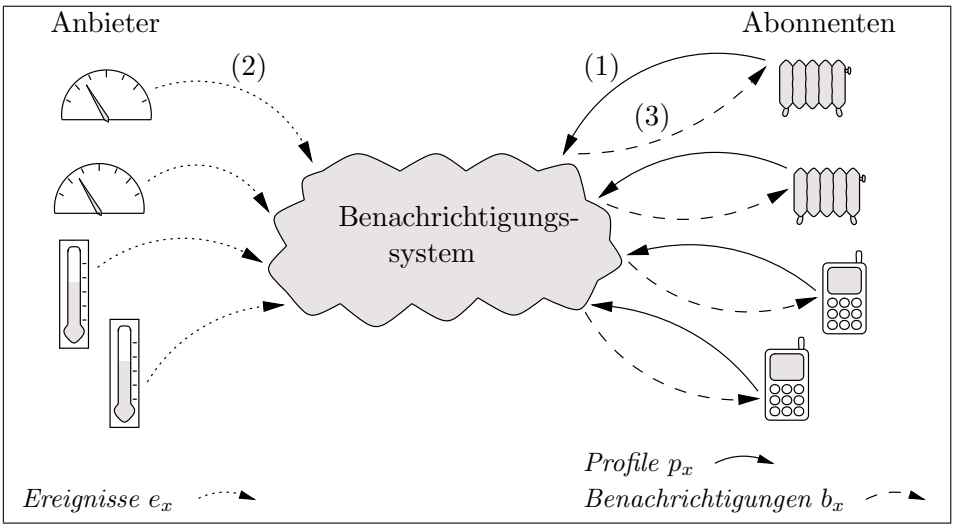


Abbildung 1.1: Übersicht eines Benachrichtigungssystems im Kontext der Gebäudesteuerung.

eine Sicherheitsfirma, mit einem Telefon als Mittel der Benachrichtigung, dargestellt. Links sind die Anbieter, Temperatursensoren und weitere Messgeräte, zu sehen. Die Abonnenten senden Profile p_x (1) mit der Definition ihrer Interessen an das System. Die Anbieter senden Ereignisse e_x (2) an das System, welche eine Zustandsänderung, den aktuellen Zustand zu einem bestimmten Zeitpunkt oder ein anderes real aufgetretenes Ereignis beschreiben. Das Benachrichtigungssystem filtert eintreffende Ereignisse und informiert interessierte Abonnenten mit Benachrichtigungen b_x (3).

Ein Ereignis e_x ist durch eine *Tupelmenge* $\mathcal{I}(e_x)$ von *Attribut-Wert-Paaren* und einen *Ereignistyp* $\mathcal{T}(e_x)$ beschrieben: $e_x = (\mathcal{I}(e_x), \mathcal{T}(e_x))$. Ein Ereignistyp T ist definiert durch eine Menge an *Attributen*: $T = \{a_1, \ldots, a_n\}$. Der Ereignistyp stellt eine Strukturbeschreibung von Ereignissen dar. Jedem Attribut a ist ein *Wertebereich* W zugeordnet. $\mathcal{W}(a)$ bezeichnet den Wertebereich des Attributes a.

Für den Wert w eines Attribut-Wert-Paares $(a, w) \in \mathcal{I}(e_x)$ gilt: $w \in \mathcal{W}(a)$. Weiterhin gelten für ein Ereignis e_x

$$\forall a_x \in \mathcal{T}(e_x) \exists w_x \in \mathcal{W}(a_x)((a_x, w_x) \in \mathcal{I}(e_x)) \text{ und}$$
$$\forall (a_x, w_x) \in \mathcal{I}(e_x) \forall (a_y, w_y) \in \mathcal{I}(e_x)(a_x = a_y \rightarrow w_x = w_y).$$

Das heißt, jedes Attribut des Typs von e_x muss in genau einem Attribut-Wert-Paar von e_x enthalten sein. Die Menge aller möglichen Ereignisse wird mit $\mathcal{E}$ bezeichnet. Der Anbieter eines Ereignisses e_x wird $\mathcal{A}(e_x)$ geschrieben. Ein Ereignis e_x des Typs $T_1 = \{a_1, a_2\}$ mit den Attributbelegungen $a_1 = w_1$ und $a_2 = w_2$ wird vereinfacht in dieser Form geschrieben

$$e_x : (a_1 = w_1, a_2 = w_2, T_1).$$

Das folgende Beispiel verdeutlicht die Darstellung von Ereignistypen und Ereignissen im in Abschnitt 1.1 eingeführten Heizungsbeispiel:

Beispiel 1.1 (Typdefinition und Ereignisdefinition) *Sei Ereignistyp* T_1 *durch* $T_1 = \{a_1, a_2\}$ *definiert. Attribut* a_1 *beschreibt den Mess-Sensor, Attribut* a_2 *die aktuell gemessene Temperatur dieses Sensors. Falls* 100 *Sensoren im Gebäude existieren, ist* $W_1 = \mathcal{W}(a_1) = [1, 100]$. *Für Temperaturen unserer Breiten wird* W_2 *durch* $W_2 = \mathcal{W}(a_2) = [-40, 60]$ *definiert. Ein Ereignis* e_1 *des Temperatursensors* S_1 *ist* $e_1 = (\{(a_1, 1), (a_2, 21)\}, T_1)$ *oder in Kurzform* $e_1 : (a_1 = 1, a_2 = 21, T_1)$. *Damit wird eine Temperatur von* 21 °*C beschrieben. Für den Anbieter* $\mathcal{A}(e_1)$ *gilt* $\mathcal{A}(e_1) = S_1$.

Abonnenten sind an Ereignissen interessiert und spezifizieren ihre Interessen als Profile mit Hilfe einer *Profildefinitionssprache*. Profile beschreiben dabei *Filteroperationen* auf Ereignissen, sie versetzen einen Abonnenten in die Lage Ereignisse aufgrund ihrer Attribut-Wert-Paare vom Benachrichtigungssystem filtern zu lassen. Die Menge aller am System angemeldeten Profile wird mit $\mathcal{P}$ bezeichnet. Der Abonnent eines Profils p_x wird auch $\mathcal{A}(p_x)$ geschrieben.

Der Benachrichtigungsdienst filtert die Ereignisse e_x der Anbieter und benachrichtigt alle Abonnenten mit passenden Profilen p_x durch eine Benachrichtigung $b_x = (p_x, e_x)$. Das Ereignis e_x erfüllt dann Profil p_x, geschrieben $e_x \succ p_x$.

Formal ist ein Profil p_x durch eine Menge an *Prädikaten* $\mathcal{PR}(p_x)$ und einen Ereignistyp $\mathcal{T}(p_x)$ beschrieben: $p_x = (\mathcal{PR}(p_x), \mathcal{T}(p_x))$. Ein Prädikat $pr \in \mathcal{PR}(p_x)$ ist ein Tripel $pr = (a, op, o)$, wobei a ein Attribut, op ein binärer Vergleichsoperator (z.B. $=, <, >$) und o rechter Operand des Vergleichsoperators op sind. Für die Menge der Prädikate $\mathcal{PR}(p_x)$ eines Profils p_x gilt

$$\forall (a_x, op_x, o_x) \in \mathcal{PR}(p_x)(a_x \in \mathcal{T}(p_x) \wedge$$
$$\nexists (a_x, op_y, o_y) \in \mathcal{PR}(p_x)(op_x \neq op_y \vee o_x \neq o_y)).$$

Ein Profil darf somit für jedes Attribut seines Typs maximal ein Prädikat enthalten. Ein Attribut-Wert-Paar $t_y = (a_y, w_y)$ *erfüllt* ein Prädikat $pr_x = (a_x, op_x, o_x)$, kurz $t_y \succ pr_x$, falls

$$a_x = a_y \wedge w_y \; op_x \; o_x.$$

Ein Ereignis e_x erfüllt ein Profil p_x, kurz $e_x \succ p_x$, wenn

$$\mathcal{T}(p_x) = \mathcal{T}(e_x) \wedge \forall pr \in \mathcal{PR}(p_x) \exists t \in \mathcal{I}(e_x)(t \succ pr).$$

Anders ausgedrückt erfüllt ein Ereignis ein Profil, wenn deren Typen gleich sind und alle Prädikate des Profiles auf den Attribut-Wert-Paaren des Ereignisses mit *Wahr* evaluieren. Die Menge der ein Profil p_x erfüllenden Ereignisse wird $\mathcal{E}(p_x)$ bezeichnet und ist definiert durch

$$\mathcal{E}(p_x) = \{e_y \in \mathcal{E} | e_y \succ p_x\}.$$

Ein Profil p_x des Typs $T_1 = \{a_1, a_2\}$ mit den Prädikaten $(a_1, =, w_1)$ und $(a_2, >, w_2)$ wird vereinfacht in dieser Form geschrieben

$$p_x : (a_1 = w_1 \wedge a_2 > w_2, T_1).$$

Folgendes Beispiel verdeutlicht diese Definition von Profilen:

Beispiel 1.2 (Profildefinition) *Im Heizungsbeispiel ist Profil* p_1 *der Heizung* H_1 *beschrieben durch* $p_1 = (\{(a_2, >, 20)\}, T_1)$ *oder in Kurzform* $p_1 : (a_2 > 20, T_1)$. *Profil* p_1 *beschreibt dabei Temperaturen über* 20 °*C, unabhängig vom messenden Sensor. Weiterhin gilt* $\mathcal{A}(p_1) = H_1$.

1.2.2 Verteilte Benachrichtigungsdienste

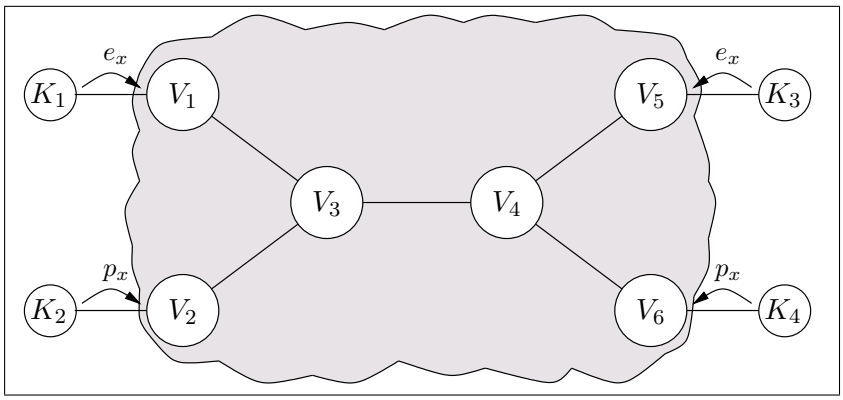


Abbildung 1.2: Übersicht eines verteilten Benachrichtigungssystems.

Ein verteilter Benachrichtigungsdienst (Abb. 1.2) ist ein Netz von mehreren Servern, auch *Vermittler* V genannt. Daher wird das Netz auch als *Vermittlungsnetz* $\mathcal{V}$ bezeichnet. Die Vermittler sind über eine beliebige Netztopologie miteinander verbunden. Ein Vermittler V muss nicht das gesamte Vermittlungsnetz $\mathcal{V}$ kennen. Die Menge der dem Vermittler V bekannten Vermittler wird seine *Nachbarvermittler* $\mathcal{N}(V)$ genannt.

Abonnenten und Anbieter von Informationen, allgemein *Klienten* K genannt, kommunizieren in einem verteilten Benachrichtigungsdienst mit einem Vermittler V ihrer Wahl, der als Schnittstelle zum Benachrichtigungssystem dient. Dieser Vermittler V wird *lokaler Vermittler* eines Klienten genannt. Die Abonnenten eines Vermittlers werden als dessen *lokale Abonnenten*, die Anbieter als *lokale Anbieter* (allg. lokale Klienten), bezeichnet.

In Abb. 1.2 existieren 6 Vermittler, V_1 bis V_6. Die Klienten K_1 und K_3 sind Anbieter von Ereignissen e_x. Sie kommunizieren mit ihren lokalen Vermittlern, V_1 bzw. V_5. K_2 und K_4 sind Abonnenten von Profilen p_x. Ihre Kommunikationspartner sind die Vermittler V_2 bzw. V_6. Die Nachbarvermittler des Vermittlers V_4 in Abb. 1.2 sind V_3, V_5 und V_6 ($\mathcal{N}(V_4) = \{V_3, V_5, V_6\}$).

Der Einsatz eines verteilten Benachrichtigungsdienstes ist notwendig, wenn eine zentralisierte Lösung die benötigte Filtereffizienz (wieviele Ereignisse können bei einer durch $|\mathcal{P}|$ charakterisierten Profilmenge pro Sekunde gefiltert werden) nicht erreicht. Das Verhalten bei Vorhandensein vieler Abonnenten (mit entsprechenden Profilmengen) und Anbieter, die Skalierbarkeit des Systems, macht ebenfalls eine verteilte Umsetzung notwendig, da ein zentraler Dienst den benötigten Kommunikationsaufwand nicht erbringen kann. Ein effizienter Filteralgorithmus eines Benachrichtigungssystems muss durch eine Hauptspeicherstruktur umgesetzt werden, da Zugriffe auf den Hintergrundspeicher bzw. Persistenzmechanismen wie in einer traditionellen Datenbank, zu einem zu starken Effizienzverlust führen [15, 31]. Unterschiede bei der Umsetzung eines verteilten Filteralgorithmus ergeben sich unter anderem in der Aufteilung des Filteraufwandes und der Netztopologie. So stellen sich beispielsweise die folgenden Fragen: Welche Vermittler führen die Filterung welcher Ereignisse durch? Wie wird

mit den Abonnenten kommuniziert? Werden Ereignisse mehrfach gefiltert? Auf solche Fragen wird diese Arbeit im Folgenden eine Antwort geben.

1.3 Zielstellung und Fokus der Arbeit

Der Fokus dieser Arbeit liegt auf dem Entwurf und der Bewertung eines verteilten Benachrichtigungsdienstes. Sie besteht aus einem theoretischen und einem praktischen Teil. Im theoretischen Teil werden vorhandene verteilte Filteralgorithmen analysiert und eine allgemeine Klassifikation verteilter Filteralgorithmen vorgestellt. Im praktischen Teil wird der Entwurf und die Implementierung des Prototypen DAS eines verteilten Benachrichtigungssystems beschrieben. Weiterhin wird eine umfangreiche Untersuchung der implementierten verteilten Filteralgorithmen anhand von Experimenten durchgeführt. Die Bestandteile der Arbeit sind im einzelnen:

Analyse verteilter Filteralgorithmen: Bisherige Arbeiten zur verteilten Filterung werden vorgestellt, analysiert und bewertet. Es handelt sich dabei um sehr spezielle Lösungen des Problems der verteilten Filterung, die als Grundlage für die weiteren Teile der Arbeit dienen.

Klassifikation verteilter Filteralgorithmen: Bisherige Arbeiten aus dem Gebiet verteilter Filteralgorithmen stellen spezielle Lösungen für die verteilte Filterung dar. Eine Einteilung solcher Algorithmen kann jedoch viel detaillierter als in den Arbeiten geschehen. Deshalb wird eine Klassifikation verteilter Filteralgorithmen vorgestellt. Diese gliedert sich in die Kriterien der Kommunikation mit den Abonnenten, der Aufteilung des Filteraufwandes, des Filterortes und der Speicherstrategie. Die bei einer Kombination der Kategorien entstehenden Filteralgorithmen werden bezüglich der wichtigen Bewertungskriterien Filtereffizienz, Skalierbarkeit, Netzlast und Speicherverbrauch analysiert.

Implementierung: Die Implementierung des Prototypen DAS eines verteilten Benachrichtigungssystems wird als Teil dieser Arbeit vorgestellt. Aufbauend auf einem zentralisierten Filteralgorithmus werden sowohl Erweiterungen der zentralen Filterstruktur als auch der Entwurf verteilter Filterkomponenten vorgestellt. DAS enthält nicht einen einzelnen verteilten Filteralgorithmus, sondern drei unterschiedliche Algorithmen. Dies sind die Algorithmen, die gemäß der o.g. Klassifikation die besten Resultate erwarten lassen.

Auswertung: Zum Vergleich der verteilten Filteralgorithmen des Systems DAS wird eine umfangreiche Analyse in Hinblick auf Effizienz, Netzlast und Speicherauslastung durchgeführt. Dabei werden Variationen zahlreicher Systemparameter untersucht: Anteil passender Profile bzw. erfüllender Ereignisse, Größe des Vermittlungsnetzes, Gleichartigkeit der Profile, Anzahl der Ereignistypen, Lokalitätsverhalten zwischen Ereignissen bzw. Profilen und Anzahl der Profile. Im Gegensatz zu den bisherigen Arbeiten wird ein reales System und keine Systemsimulation untersucht. Die Ergebnisse und deren Auswertung geben damit Aufschluss über die Vor- und Nachteile der unterschiedlichen Filteralgorithmen unter realen Bedingungen.

1.4 Gliederung der Arbeit

Kapitel 2 gibt einen ersten Überblick über Filterverfahren aus dem Gebiet der verteilten Benachrichtigungsdienste. Diese werden erläutert und bei gleichen Grundideen einer ersten Bewertung unterzogen. Kapitel 3 enthält eine ausführliche Klassifikation von verteilten Filteralgorithmen, als Verallgemeinerung und Erweiterung der in Kapitel 2 teilweise vorgestellten Arbeiten. Die Algorithmen mit den besten zu erwartenden Resultaten bzgl. Filtereffizienz, Skalierbarkeit, Netzlast und Speicherverbrauch werden für die prototypische Implementierung gewählt. Der Entwurf des Benachrichtigungssystems DAS wird in Kapitel 4 vorgestellt. Dazu wird die Systemarchitektur schematisch und mit Hilfe von UML-Diagrammen (Unified Modelling Language) erläutert. In Kapitel 5 werden die Filterprotokolle der entworfenen Algorithmen detailliert dargestellt. Kapitel 6 enthält die Ergebnisse und Auswertungen zahlreicher Tests und Experimente, welche zur Bewertung und Einordnung der Implementierung durchgeführt wurden. Es werden verschiedene Einflussparameter des Systems untersucht und die Effizienz, Netzlast und Speicherauslastung der Filteralgorithmen analysiert. Als Abschluss befindet sich in Kapitel 7 eine Zusammenfassung der Arbeit und ein Ausblick auf weitere Forschungsmöglichkeiten im Bereich verteilter Benachrichtigungsdienste.

Kapitel 2

Verwandte Arbeiten

Inhalt

In diesem Kapitel werden vorhandene Arbeiten aus dem Gebiet der verteilten Filteralgorithmen vorgestellt. Es wird ein erster Überblick über die eingesetzten Techniken und Verfahren gegeben. In der Literatur sind zwei unterschiedliche Ansätze zu finden: Rendezvousknoten (Abschnitt 2.1) und Verteilte Auswertung (Abschnitte 2.2 und 2.3). Abschnitt 2.4 stellt Verfahren zur Verminderung von Profilredundanzen vor, welche in den vorgestellten verteilten Filterstrategien genutzt werden können. Als Abschluss erfolgt jeweils eine kurze Bewertung der Verfahren. Eine Zusammenfassung der vorgestellten Ansätze ist in Abschnitt 2.5 zu finden.

2.1 Rendezvousknoten

Rendezvousknoten wurden von Rowstron und anderen [30] vorgestellt und von Pietzuch und Bacon [27] erweitert. Die Netztopologie beim Einsatz von Rendezvousknoten ist ein azyklischer ungerichteter Graph. Ein Rendezvousknoten V ist eine spezielle Art von Vermittlungsknoten, welcher für eine bestimmte Menge von Ereignistypen $\mathcal{T}(V)$ zuständig ist. Es gelten

$$\forall V_x, V_y \in \mathcal{V}, V_x \neq V_y(\mathcal{T}(V_x) \cap \mathcal{T}(V_y) = \emptyset) \text{ und}$$
$$\bigcup_{V \in \mathcal{V}} \mathcal{T}(V) \supseteq \{\mathcal{T}(p_x)|p_x \in \mathcal{P}\} \cup \{\mathcal{T}(e_x)|e_x \in \mathcal{E}\}.$$

Zwei Rendezvousknoten sind also nie für ein und denselben Ereignistyp zuständig und für jedes Profil der Profilmenge $\mathcal{P}$ und jedes mögliche Ereignis $e \in \mathcal{E}$ existiert genau ein Rendezvousknoten.

Rendezvousknoten fungieren als Treffpunkt für Ereignisse und Profile ihrer Typen. Dazu werden Ereignisse e_x und Profile p_x von lokalen Vermittlern auf dem kürzesten Weg zum Rendezvousknoten V ihres jeweiligen Typs ($\mathcal{T}(e_x) \in \mathcal{T}(V)$ bzw. $\mathcal{T}(p_x) \in \mathcal{T}(V)$) gesendet. Die Benachrichtigung über ein Ereignis erfolgt auf dem entgegengesetzten Pfad, auf welchem das zugehörige Profil auf dem Weg zum Rendezvousknoten im Vermittlungsnetz weitergeleitet wurde.

Rendezvousknoten müssen allen Vermittlern des Systems bekannt sein. Dazu sind die Vermittler mit einem Identifikator gekennzeichnet. Der Vermittler mit dem Hashwert des Ereignistypen eines Profils oder Ereignisses als Identifikator bzw. der numerisch nächstgelegene fungiert als Rendezvousknoten dieses Typs.

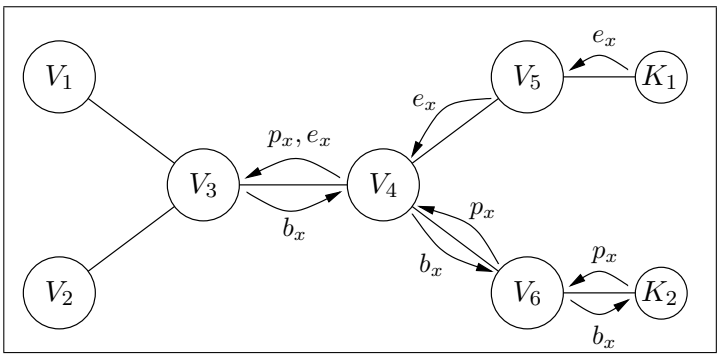


Abbildung 2.1: Konfiguration bei Nutzung von Rendezvousknoten.

In Abb. 2.1 ist eine Beispielkonfiguration des Ansatzes der Rendezvousknoten zu sehen. Das Vermittlungsnetz besteht aus den 6 Vermittlern V_1 bis V_6, es gilt $T_2 \in \mathcal{T}(V_3)$ (V_3 ist Rendezvousknoten des Typs T_2). Es sind 2 Klienten mit dem System verbunden. K_1 ist Anbieter von Ereignissen e_x mit $\mathcal{T}(e_x) = T_2$, K_2 ist Abonnent mit Profilen p_x mit $\mathcal{T}(p_x) = T_2$. Die Profile p_x werden über das Vermittlungsnetz zu Vermittler V_3 geleitet. Auch Ereignisse e_x werden bei Veröffentlichen zum Rendezvousknoten geleitet. Dort findet die Filterung statt und Benachrichtigungen b_x werden von V_3 über das Vermittlungsnetz zum Abonnenten K_2 gesendet.

2.1.1 Exklusive Filterung

Das System Scribe [30] nutzt den vorgestellten Ansatz der Rendezvousknoten. In Scribe ist keine attributbasierte Filterung von Ereignissen möglich. Abonnenten können sich für verschiedene Themengebiete (Realisierung über Ereignistypen) des Dienstes anmelden. Dann erhalten sie sämtliche Ereignisse bezüglich dieses Gebiets. Die Filterung eines Themengebietes findet ausschließlich im Rendezvousknoten statt. Deshalb speichern Vermittler, welche Profile zum Rendezvousknoten weiterleiten, keine Profile. Seien in Abb. 2.1 die Profile p_x bezüglich eines Themengebietes S_1 und die Ereignisse e_x bezüglich des Themengebietes S_1. Nur Vermittler V_3 speichert dann die Profile p_x. V_4 leitet bei der dargestellten Konfiguration lediglich Ereignisse, Benachrichtigungen und Profile weiter, Vermittler V_6 nur Profile und Benachrichtigungen. Vermittler V_5 leitet Ereignisse weiter.

2.1.2 Unterwegsfilterung

Bei der Unterwegsfilterung, vorgestellt im System Hermes [27], werden Informationen über Profile in allen Vermittlern gespeichert, die auf dem Weg zum Rendezvousknoten besucht werden. Dort finden auch die Filterungen der Ereignisse und bei Bedarf die Benachrichtigungen statt. In Abb. 2.1 würden die Vermittler V_3, V_4 und V_6 die Profile p_x speichern und filtern.

Ereignisse werden über die Vermittler zum Rendezvousknoten gesendet. Auf diesem Weg findet schon eine Filterung statt, da Vermittlern Profile ihrer Abonnenten und teilweise Profile der lokalen Abonnenten ihrer Nachbarn bekannt sind. Im Rendezvousknoten findet eine letzte Filterung statt, welche Benachrichtigungen an alle Abonnenten weiterleitet, die nicht über den Vermittler, der das Ereignis weitergeleitet hat, abonniert wurden. Damit werden Benachrichtigungen über gleiche Ereignisse vermieden.

2.1.3 Wertung

Rendezvousknoten sind die Kopplung eines verteilten Ansatzes mit einer zentralisierten Lösung. Es existiert jeweils ein Knoten im Netz, welcher für eine bestimmte Menge an Profilen zuständig ist. Ebenfalls werden alle Ereignisse eines Typs an diesen Knoten weitergeleitet. Damit entfällt der durch die Verteilung des Systems entstehende Vorteil der Aufteilung von Filter- und Speicheraufwand. Lediglich in dem Fall, dass die Menge der Ereignistypen größer oder gleich der Vermittlerzahl ist und diese Typen in den Profilen und Ereignissen zur gleichverteilten Belastung der Rendezvousknoten führen, ist eine Aufwandsverteilung im gesamten Vermittlungsnetz gegeben.

Das System Hermes [27] versucht die Aufwandsverteilung durch die Unterwegsfilterung zu kompensieren. Dadurch wird der Speicheraufwand erhöht, da mehrere Knoten die gleichen Profile speichern und filtern. Ein Gewinn an Effizienz ist nur unter sehr speziellen Umständen zu erwarten, denn im Rendezvousknoten müssen zwar nicht alle Benachrichtigungen ausgeliefert werden, eine Filterung muss aber dennoch stattfinden. Nur Profile, die auf dem Weg eines Ereignisses zum Rendezvousknoten gefiltert werden, erfahren dadurch eine frühzeitigere Filterung. Profile der Abonnenten lokaler Vermittler anderer Teilnetze erfahren diese schnellere Filterung jedoch nicht.

Die Weiterleitung der Ereignisse im Netz stellt ein weiteres Problem dar. Da mit

sehr hohen Ereignisfrequenzen zu rechnen ist, wird die Netzlast in der Nähe der Rendezvousknoten und deren Belastung selbst sehr hoch ausfallen.

2.2 Verteilte Auswertung: Hierarchische Netze

Bei der verteilten Auswertung findet die Filterung an verschiedenen Orten im Vermittlungsnetz statt. Im Folgenden werden Ansätze basierend auf baumartigen Netztopologien vorgestellt.

2.2.1 Link State Matching

Ein Ansatz zur Filterung in hierarchischen Netzen, Link State Matching, wird von Banavar und anderen in [2] vorgestellt. Dazu müssen jedem Vermittlungsknoten alle Profile bekannt sein.

Vor der Filterung wird aus den Profilen in jedem Vermittler eine baumartige Filterstruktur [1] erzeugt. Diese wertet in jeder Baumebene genau ein Attribut der Profile aus. Aus der Netzstruktur wird in jedem Knoten des Baumes ein Vektor von der Größe der Anzahl der Kindnetzknoten (das sind Vermittler) dieses Knotens integriert. Dieser enthält drei mögliche Werte: *Ja*, *Nein*, *Vielleicht*. Die ersten beiden beschreiben, ob im entsprechenden Netzpfad ein passendes Profil vorhanden ist oder nicht. Im dritten Fall kann eine Entscheidung in dieser Baumebene noch nicht getroffen werden. Bei der Filterung wird der Baum soweit abgearbeitet, bis alle Einträge des erreichten Vektors entweder *Ja* oder *Nein* enthalten. Dann wird das Ereignis an die entsprechenden Kindknoten (Vermittler) ausgeliefert bzw. nicht weitergeleitet.

2.2.2 Hierarchie von Vermittlungsknoten

Yu und andere teilen in [38] die Vermittler in mehrere Gruppen ein, welche jeweils in einer Baumstruktur verbunden sind. Die Wurzelelemente einer jeden Gruppe können wiederum an anderen Gruppen beteiligt sein. Die Gesamtstruktur ist damit ein Baum mit Bäumen als Baumknoten.

Für die Filterung der Ereignisse existiert ein Protokoll zwischen den Knoten, das folgendes Ziel verfolgt: Die Vermittler haben stets so viele Informationen über Profile, wie notwendig sind, um ein Ereignis an alle Nachbarn mit lokalen Klienten mit passenden Profilen (innerhalb ihrer Gruppe) oder solchen Nachbarvermittlern ausliefern zu können. Dazu existieren in jedem Vermittler Listen mit Informationen, von welchen Nachbarn Ereignisse eintreffen und welche Nachbarn Profile spezifiziert haben. Zum Informationsaustausch werden Profile entweder an Kindknoten oder soweit in Richtung Wurzel weitergeleitet, wie es nötig ist. Trifft ein Ereignis ein, so wird es an alle interessierten Nachbarn gesendet. Dieses erfolgt rekursiv, so dass die gesamte Hierarchie abgearbeitet wird. Abb. 2.2 zeigt eine Beispielkonfiguration mit 9 Vermittlern. Jede Gruppe besteht aus 3 Vermittlern. K_1 ist Anbieter von Ereignissen e_x, K_2 ist Abonnent von Profilen p_x. Diese Profile und Ereignisse werden jeweils zur Wurzel einer Gruppe weitergeleitet. Vermittler V_2 ist p_x bekannt und dieser leitet daraufhin e_x an V_6 weiter, welcher die Benachrichtigung b_x von K_2 ausführt.

Carzaniga und andere verfolgen in [5] einen ähnlichen Ansatz. Der Unterschied ist, dass eine reine Baumstruktur der Vermittler angenommen wird, also keine Gruppen

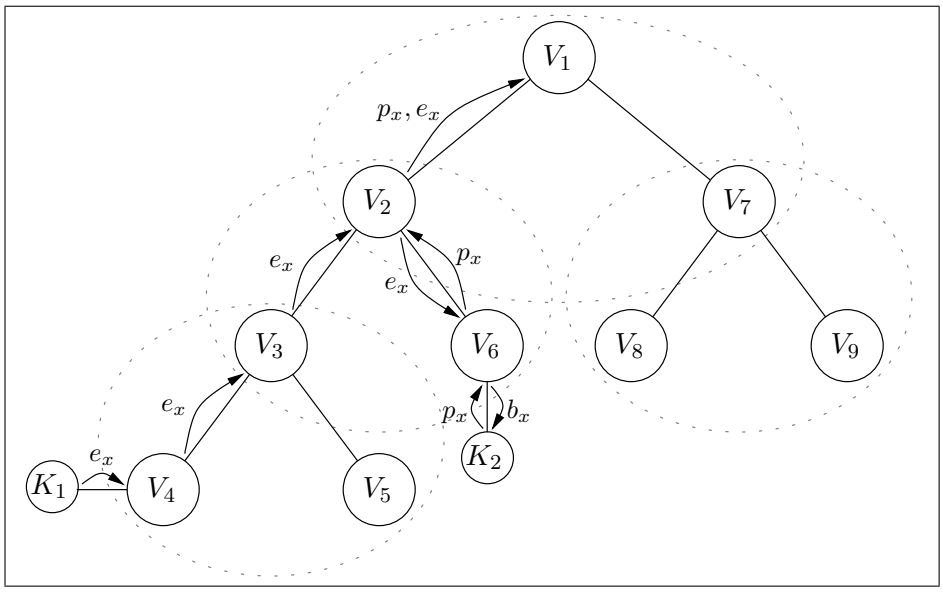


Abbildung 2.2: Konfiguration bei Nutzung einer Hierarchie von Vermittlungsknoten.

von Vermittlern einen Knoten des Baumes bilden. Ein Ereignis wird stets im aktuellen Knoten gefiltert und an den Mutterknoten gesendet. Dieser entscheidet, welche Kindknoten ebenfalls interessiert sind, filtert für seine Klienten und leitet das Ereignis wiederum an seinen Mutterknoten weiter. Zur Realisierung dessen muss ein Mutterknoten die Profile seiner Kinder kennen.

2.2.3 Wertung

In beiden baumbasierten Ansätzen existiert das Problem der starken Auslastung des Wurzelknotens. Beim Link State Matching muss die Baumwurzel jedes Ereignis auswerten, da die Ereignisverbreitung nur von der Wurzel zu den Blättern geschehen kann. Vorteilhaft ist, dass die Filterung eines Ereignisses u.U. nicht die Auswertung aller Attribute beinhaltet. Die Hierarchie von Vermittlungsknoten zeigt das gleiche Problem, dass alle Ereignisse zur Wurzel weitergeleitet werden müssen. Durch die Realisierung eines Baumes aus Bäumen müssen jedoch nicht alle Ebenen eine Filterung durchführen. Mit steigender Höhe der Teilbäume in den Knoten werden die Filterungen in verschiedenen Ebenen vermindert. Beide Ansätze versuchen also das strukturelle Problem der Filterung in verschiedenen Baumebenen mit unterschiedlichen Strategien zu verkleinern.

2.3 Verteilte Auswertung: Punkt-zu-Punkt-Netze

Carzaniga und andere beschreiben in [5] Algorithmen zur Filterung in Punkt-zu-Punkt-Netzen. Dabei werden Netztopologien in Form azyklischer Graphen genutzt.

2.3.1 Profilweiterleitung

Bei der Profilweiterleitung werden Profile im gesamten Vermittlungsnetz verteilt. Tritt ein Ereignis auf, so ist der Vermittlungsknoten des Anbieters für die Filterung verantwortlich. Jeder Vermittler führt die Filteroperationen durch, um Klienten und alle Nachbarn mit lokalen Abonnenten mit passenden Profilen, oder solchen Nachbarn ihrerseits, benachrichtigen zu können (analog zur Hierarchie von Vermittlungsknoten in Abschnitt 2.2.2). Abonnenten passender Profile werden dann auf dem entgegengesetzten Weg des Pfades, den das Profil zum filternden Vermittlungsknoten zurückgelegt hat, über ein Ereignis benachrichtigt. In [5] formuliert Carzaniga zur Erläuterung die Eigenschaft, dass die Filterung bei Profilweiterleitung so nah bei den Anbietern wie möglich stattfindet.

2.3.2 Ereignisweiterleitung

Die Ereignisweiterleitung verfolgt die entgegengesetzte Strategie zur Profilweiterleitung. Ereignisse werden im gesamten Netz verbreitet, Profile werden von lokalen Vermittlern nicht weitergeleitet. Abonnenten passender Profile werden von ihren lokalen Vermittlern benachrichtigt, eine Weiterleitung von Benachrichtigungen im Vermittlungsnetz erfolgt nicht. Die Filterung findet also so nah bei den Abonnenten wie möglich statt [5].

2.3.3 Wertung

Punkt-zu-Punkt-Netze sind eine Verallgemeinerung der hierarchischen Ansätze aus Abschnitt 2.2, denn azyklische Graphen (auch freie Bäume genannt) sind eine Obermenge der Bäume mit fester Wurzel.

Die Profilweiterleitung leitet keine Ereignisse im Vermittlungsnetz weiter. Dies ist eine gute Strategie, wenn wenige Ereignisse Profile erfüllen. Erfüllen jedoch viele Ereignisse Profile, sollte die Ereignisweiterleitung bevorzugt werden, da die Ereignisse sowieso im Vermittlungsnetz verbreitet werden müssen. Vorteile der jeweiligen Strategien sind somit von der Verteilung und Art der Ereignisse bzw. Profile abhängig.

2.4 Verminderung von Profilredundanzen

Im System Hermes [27] sowie bei der verteilten Auswertung [2, 5, 38] werden die im Folgenden vorgestellten Optimierungsstrategien genutzt. Diese sollen eine Verminderung von Profilredundanzen und damit des Filteraufwandes erreichen. Eine Variante versucht an Nachbarvermittler nur solche Profile weiterzuleiten, welche für diese zur Filterung relevant sind (Abschnitte 2.4.1 und 2.4.2). Dabei treten die Vermittler als Abonnenten auf und führen bei Benachrichtigungen eine Nachfilterung der Ereignisse durch, um die tatsächlichen Abonnenten zu benachrichtigen. Eine andere Variante ist, mehrere Profile zusammenzufassen (Abschnitt 2.4.3). Es wird im Weiteren von folgender Vereinfachung des Heizungsbeispiels aus Kapitel 1 ausgegangen:

Beispiel 2.1 (Vereinfachung des Heizungsbeispiels) *Sei der Ereignistyp für Temperaturereignisse* $T_3 = \{a_2\}$ *und Wertebereich* $\mathcal{W}(a_2)$ *ganzzahlig und gegeben durch* $\mathcal{W}(a_2) = [-40, 60]$. *Der messende Sensor wird somit nicht als Attribut eines Temperaturereignisses des Typs* T_3 *genutzt.*

2.4.1 Äquivalenz

Zwei Profile p_x und p_y können gemäß Mühl und anderen als äquivalent [26] angesehen werden, geschrieben $p_x \equiv p_y$, wenn sie den gleichen Ereignistyp beschreiben und die Menge der die Profile erfüllenden Ereignisse gleich ist ($\mathcal{T}(p_x) = \mathcal{T}(py) \wedge \mathcal{E}(p_x) = \mathcal{E}(p_y)$).

Eine Optimierung gestaltet sich dann wie folgt: Ein Profil p_x muss nur dann von Vermittler V_x an einen Nachbarvermittler weitergeleitet werden, wenn an diesen noch kein Profil p_y mit $p_x \equiv p_y$ gesendet wurde. Dieses liefert korrekte Ergebnisse, da zu Profil p_y die gleichen Ereignisse wie zu Profil p_x passen. Trifft bei Vermittler V_x eine Benachrichtigung über p_y ein, führt V_x eine Nachfilterung durch und benachrichtigt sowohl den Abonnenten von p_x als auch von p_y. Äquivalenz zwischen Profilen ist nicht nur bei gleichen Profildefinitionen der Fall, wie folgendes Beispiel zeigt:

Beispiel 2.2 (Äquivalenz von Profilen) *Sei Profil p_1 durch $p_1 : (a_2 > 58, T_3)$ und Profil p_2 durch $p_2 : (a_2 \in \{59, 60\}, T_3)$ gegeben. Die Profile p_1 und p_2 haben unterschiedliche Definitionen, jedoch die gleiche Menge erfüllender Ereignisse: $\mathcal{E}(p_1) = \mathcal{E}(p_2) = \{(a_2 = 59, T_3), (a_2 = 60, T_3)\}$.*

2.4.2 Bedecken

Ein Profil p_x bedeckt gemäß Mühl und anderen [25, 26] ein Profil p_y (geschrieben $p_x \sqsupseteq p_y$), wenn $\mathcal{E}(p_x) \supseteq \mathcal{E}(p_y)$. Gilt $\mathcal{E}(p_x) \supset \mathcal{E}(p_y)$, so spricht man von echter Bedeckung ($p_x \sqsupset p_y$). Als Optimierung braucht ein Profil p_x von Vermittler V_x nur dann an einen Nachbarvermittler weitergeleitet werden, wenn noch kein Profil p_y mit $p_y \sqsupset p_x$ an diesen gesendet wurde. Wurde schon ein solches Profil p_y gesendet, wird $\mathcal{A}(p_x)$ auch ohne Weiterleitung von p_x korrekt benachrichtigt, da die Benachrichtigungen des Profils p_y eine Obermenge der Benachrichtigungen des Profils p_x sind. Durch die Nachfilterung in V_x werden die Abonnenten der Profile p_x und p_y benachrichtigt.

Eine weitere Optimierung ist folgende: Erhält man von einem Nachbarvermittler V_x ein Profil p_x, dann brauchen die Profile $\{p_y \in \mathcal{P} | \mathcal{A}(p_y) = V_x, p_y \sqsubseteq p_x\}$ (von p_x bedeckte Profile des Nachbarn V_x) nicht mehr gefiltert zu werden, sondern lediglich p_x. Auch hier ist es ausreichend die Benachrichtigungen des bedeckenden Profils weiterzuleiten, da diese eine Obermenge der Benachrichtigungen der bedeckten Profile darstellen. Folgendes Beispiel macht die Eigenschaft der Bedeckung deutlich:

Beispiel 2.3 (Bedeckung zwischen Profilen) *Profil p_3 sei durch $p_3 : (a_2 > 50, T_3)$ und Profil p_4 durch $p_4 : (a_2 > 40, T_3)$ definiert. Es gilt $p_4 \sqsupseteq p_3$, da $\mathcal{E}(p_4) = \{(a_2 = 41, T_3), (a_2 = 42, T_3), \ldots, (a_2 = 60, T_3)\} \supseteq \mathcal{E}(p_3) = \{(a_2 = 51, T_3), (a_2 = 52, T_3), \ldots, (a_2 = 60, T_3)\}$.*

Berechnung der Bedeckungen

Das Berechnen der Bedeckungen zur Verhinderung der Weiterleitung von Profilen ist u.U. sehr aufwendig, bei entsprechender Gestaltung der Profildefinitionssprache sogar NP-vollständig [8]. Allerdings gestaltet sich bei Einschränkungen in den Operatoren der Prädikate der Profile die Umsetzung einfacher. Zur Berechnung der Bedeckungen kann allgemein folgender Grundsatz genutzt werden

$$p_1 \sqsupseteq p_2 \text{ gdw. } \forall pr_i \in \mathcal{PR}(p_1) \exists pr_j \in \mathcal{PR}(p_2)(pr_i \sqsupseteq pr_j)$$
$$\text{bzw. } p_1 \sqsupset p_2 \text{ gdw. } \forall pr_i \in \mathcal{PR}(p_1) \exists pr_j \in \mathcal{PR}(p_2)(pr_i \sqsupset pr_j).$$

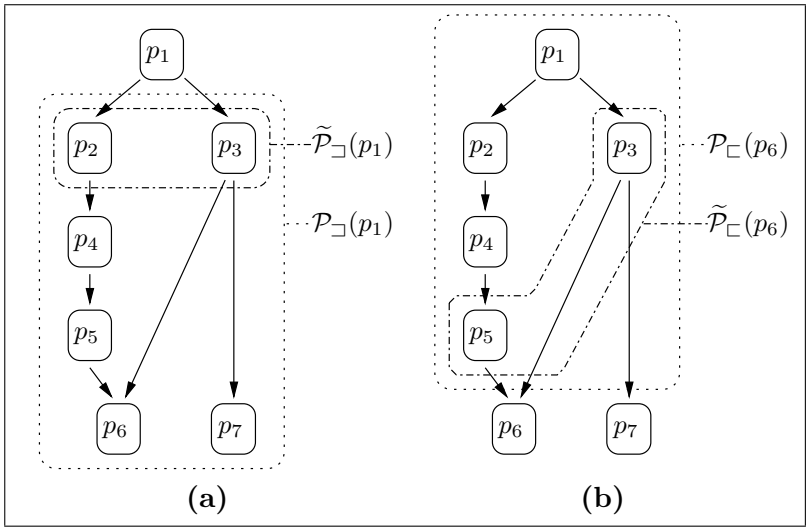


Abbildung 2.3: Beispiele der bedeckten und bedeckenden Profilmengen.

Die Berechnung der Bedeckungen kann entweder sofort beim An- bzw. Abmelden von Profilen oder jeweils auf Anfrage geschehen. Beide Möglichkeiten werden im Folgenden vorgestellt. Zuerst werden einige spezielle Profilmengen definiert. Abb. 2.3 zeigt eine mögliche Halbordnung auf den 7 Profilen p_1 bis p_7. Die Pfeile zeigen dabei jeweils von einem Profil p_x zu den direkt bedeckten Profilen. Es gelten also (nur direkte Bedeckungen): $p_1 \sqsupset p_2$, $p_1 \sqsupset p_3$, $p_2 \sqsupset p_4$, $p_3 \sqsupset p_6$, $p_3 \sqsupset p_7$, $p_4 \sqsupset p_5$, $p_5 \sqsupset p_6$. Die Menge der von einem Profil p_x echt bedeckten Profile $\mathcal{P}_{\sqsupset}(p_x)$ ist definiert durch

$$\mathcal{P}_{\sqsupset}(p_x) = \{p_y \in \mathcal{P} | p_x \sqsupset p_y\}.$$

In Abb. 2.3(a) bedeckt Profil p_1 die Profile p_2 bis p_7 echt. Analog wird die Menge $\mathcal{P}_{\sqsupseteq}(p_x)$ der bedeckten Profile definiert. Die Menge der ein Profil p_x echt bedeckenden Profile $\mathcal{P}_{\sqsubset}(p_x)$ ist definiert durch

$$\mathcal{P}_{\sqsubset}(p_x) = \{p_y \in \mathcal{P} | p_x \sqsubset p_y\}.$$

In Abb. 2.3(b) sind die Profile p_1 bis p_5 die Profil p_6 echt bedeckenden Profile. Analog wird die Menge $\mathcal{P}_{\sqsubseteq}(p_x)$ der bedeckenden Profile definiert. Die Menge der von einem Profil p_x direkt bedeckten Profile $\widetilde{\mathcal{P}}_{\sqsupset}(p_x)$ ist definiert durch

$$\widetilde{\mathcal{P}}_{\sqsupset}(p_x) = \{p_y \in \mathcal{P}_{\sqsupset}(p_x) | \nexists p_z \in \mathcal{P}_{\sqsupset}(p_x)(p_z \sqsupset p_y)\}.$$

Die Profile p_2 und p_3 werden in Abb. 2.3(a) von Profil p_1 direkt bedeckt. Die Menge der ein Profil p_x direkt bedeckenden Profile $\widetilde{\mathcal{P}}_{\sqsubset}(p_x)$ ist definiert durch

$$\widetilde{\mathcal{P}}_{\sqsubset}(p_x) = \{p_y \in \mathcal{P}_{\sqsubset}(p_x) | \nexists p_z \in \mathcal{P}_{\sqsubset}(p_x)(p_z \sqsubset p_y)\}.$$

Die Profile p_3 und p_5 sind in Abb. 2.3(b) die Profil p_6 direkt bedeckenden Profile. Und schließlich ist die Menge der zu einem Profil p_x äquivalenten Profile $\mathcal{P}_{\equiv}(p_x)$ definiert durch

$$\mathcal{P}_{\equiv}(p_x) = \{p_y \in \mathcal{P} | p_x \equiv p_y \wedge p_x \neq p_y\}.$$

Im Folgenden werden die beiden Methoden zur Berechnung von Bedeckungen angegeben.

Sofortige Berechnung. Carzaniga schlägt in [5] eine Struktur zur sofortigen Berechnung von Bedeckungen vor. Dazu werden alle Profile beim An- bzw. Abmelden in eine Halbordnung ein- bzw. ausgetragen und ständig gespeichert. Abb. 2.4 zeigt eine solche Halbordnung mit Profilen p_1 bis p_7 des Typs $T_1 = \{a_1, a_2\}$ des Heizungsbeispiels aus Kapitel 1. Die Pfeile zeigen jeweils von einem Profil p_x zu allen Profilen $p_y \in \widetilde{\mathcal{P}}_{\sqsupset}(p_x)$.

Das Nutzen einer solchen Halbordnung führt jedoch zu einem starken Effizienzverlust, da der Aufbau der vorgeschlagenen halbgeordneten Menge nicht effizient möglich ist. Die Halbordnung wird zwar beim Anmelden bzw. Abmelden der Profile aufgebaut. Dabei ist ein Effizienzverlust nicht von so großer Bedeutung wie beim Filtern, jedoch zeigt sich auch hier zu schlechtes Laufzeitverhalten. Der Aufwand für eine Einfüge- bzw. Löschoperation von p_x beträgt $O(k)$ bei k Elementen in der Menge (angenommen $\nexists p_y \in \mathcal{P}(p_y \equiv p_x)$):

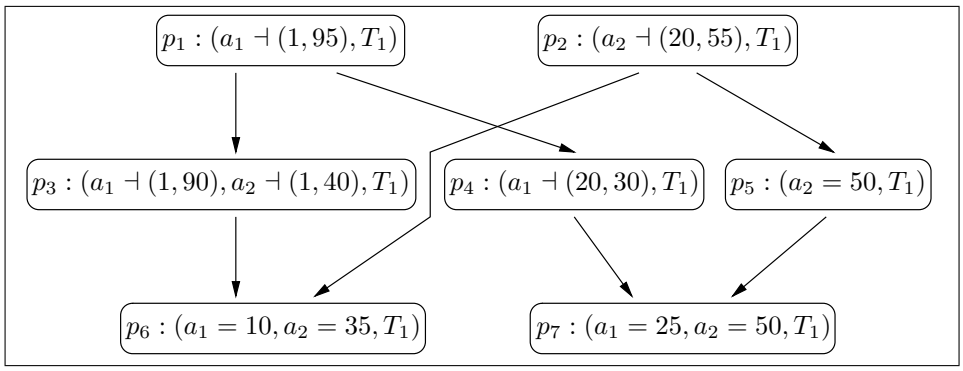


Abbildung 2.4: Beispiel einer Halbordnung.

- Beim Einfügen müssen alle Pfade von Profilen p_y mit $p_y \sqsupset p_x$ verfolgt werden, bis ein Profil p_z mit $p_z \sqsubset p_x$ gefunden wird. Zwischen p_z und seinem Elternknoten kann dann p_x eingetragen werden. Bedecken alle Profile $p_y \in \mathcal{P}$ Profil p_x, müssen alle Knoten der Struktur abgearbeitet werden.

- Beim Entfernen von p_x muss die Struktur solange nach unten abgearbeitet werden, wie Profile p_y mit $p_y \sqsupset p_x$ existieren. Gibt es nur bedeckende Profile, müssen alle Knoten bearbeitet werden.

Neben diesem schlechten Laufzeitverhalten wird für das Speichern der Kanten der Halbordnung viel Arbeitsspeicher benötigt.

Berechnung auf Anfrage. Mühl schlägt in [23] vor, die Bedeckungen nicht beim Anmelden und Abmelden der Profile zu berechnen und diese zu speichern, sondern die Berechnungen einzeln auf Anfrage durchzuführen. Dann wird kein zusätzlicher Speicherplatz benötigt, da die Bedeckungen nach der Berechnung nicht gespeichert werden.

Allerdings kann die Berechnung sehr zeitintensiv ausfallen. Deshalb wird für Profile p_x nicht die Menge der direkt bedeckten Profile $\widetilde{\mathcal{P}}_{\sqsupset}(p_x)$ berechnet, sondern lediglich die Menge der bedeckten Profile $\mathcal{P}_{\sqsupset}(p_x)$. Dadurch wird beim Abmelden von Profilen u.U. mehr Netzverkehr erzeugt (die von Profil p_x bedeckten Profile sind den Nachbarvermittlern unbekannt), da $\mathcal{P}_{\sqsupset}(p_x)$ anstatt $\widetilde{\mathcal{P}}_{\sqsupset}(p_x)$ an Nachbarvermittler gesendet wird und $\mathcal{P}_{\sqsupset}(p_x) \supset \widetilde{\mathcal{P}}_{\sqsupset}(p_x)$.

2.4.3 Verschmelzen

Ein Profil p_x ist eine Verschmelzung [25, 26] zweier Profile p_y und p_z (geschrieben $p_x \sqsupset \{p_y, p_z\}$), wenn $\mathcal{E}(p_x) \supset \mathcal{E}(p_y) \cup \mathcal{E}(p_z)$. Gilt $\mathcal{E}(p_x) = \mathcal{E}(p_y) \cup \mathcal{E}(p_z)$, spricht man von perfektem Verschmelzen (geschrieben $p_x \sqsupseteq \{p_y, p_z\}$). Folgendes Beispiel erläutert dieses am vereinfachten Heizungsbeispiel:

Beispiel 2.4 (Verschmelzung zwischen Profilen) *Sei* $p_5 : (a_2 = 100, T_3)$ *und* $p_6 : (a_2 > 90, T_3)$. *Profil* $p_7 : (a_2 > 90, T_3)$ *ist eine perfekte Verschmelzung von* p_5 *und* p_6 *(*$p_7 \sqsupseteq \{p_5, p_6\}$*). Profil* $p_8 : (a_2 > 88, T_3)$ *ist eine Verschmelzung, allerdings keine perfekte (*$p_8 \sqsupset \{p_5, p_6\}$*).*

Als Optimierung können zwei Profile eines nicht lokalen Abonnenten zu einem einzigen Profil verschmolzen werden, um die Profilanzahl zu verringern. Bei perfektem Verschmelzen erhält man auf jeden Fall eine Aufwandsverringerung. Bei imperfektem Verschmelzen werden u.U. mehr Ereignisse weitergeleitet, was eine Erhöhung des Netzverkehrs zur Folge hat. Jedoch findet aufgrund der Verschmelzung in den Vermittlern die Filterung über weniger Profilen statt.

2.4.4 Wertung

Das Ausnutzen der Äquivalenz zwischen Profilen stellt einen Spezialfall der Bedeckungen dar. Eine Anwendung des Bedeckens ist bei gleichartigen und ähnlichen Profilen verschiedener Abonnenten gut vorstellbar (Verschmelzen weniger):

Beispiel 2.5 (Anwendung des Bedeckens) *Sei im Heizungsbeispiel aus Kapitel 1 jeder einzelne Heizkörper für ein Abschalten bei hohen Außentemperaturen verantwortlich. Temperatursensor* S_1 *messe erneut die Außentemperaturen.*

Sollen alle Heizkörper bei Temperaturen über $20\ ^\circ C$ *abschalten, so sind sämtliche Profile der Heizkörper gleich mit* $p_x : (a_1 = 1, a_2 > 20, T_1)$. *Die Berechnung von Bedeckungen ist in diesem Fall nützlich, da Vermittler, anstatt über den Profilen aller Heizkörper zu filtern, teilweise nur über einem filtern (teilweise, weil Vermittler keine Bedeckungen der Profile lokaler Abonnenten ausnutzen).*

Sind den Raumnutzern kleine Variationen gestattet (Abschalten des Heizkörpers ab einem beliebigen Wert w_x, *also die Profile* $p_x : (a_1 = 1, a_2 > w_x, T_1)$*), bedeckt das Profil des Nutzers mit Abschalten des Heizköpers bei den niedrigsten Temperaturen die anderen Profile. Es ergibt sich der gleiche Vorteil der Filterung über weniger Profilen.*

Das Nutzen von Verschmelzungen ist bei wenigen Bedeckungen zwischen Profilen sehr gut einsetzbar. Beispielsweise ergeben sich Vorteile, wenn verschiedene Bereiche von Attributwerten abgefragt werden (Bedecken kann hier nicht angewendet werden):

Beispiel 2.6 (Anwendung des Verschmelzens) *Sei erneut jeder Heizkörper für das Abschalten verantwortlich. In jedem Raum existiert ein Sensor S_x zur Messung der aktuellen Temperatur. Die verschiedenen Profile der Heizkörper gestalten sich wie folgt: $p_x : (a_1 = S_x, a_2 > 20, T_1)$. Diese Profile können in einigen Vermittlern (einige, da Profile lokaler Abonnenten nicht verschmolzen werden) zu einem Profil $p_x : (a_2 > 20, T_1)$ verschmolzen werden. Damit ist die Temperaturmessung über 20 °C eines beliebigen Sensors beschrieben. Durch die Nachfilterung werden die Abonnenten dann korrekt benachrichtigt.*

Ob das Ausnutzen von Bedeckungen oder die Verschmelzung von Profilen bessere Resultate liefert, ist also stark vom Anwendungsfall abhängig. Deshalb werden erst im Verlauf der Arbeit weitere Argumente für bzw. gegen diese Verfahren erarbeitet.

2.5 Zusammenfassung

In diesem Kapitel wurden ausgewählte verteilte Filterstrategien vorgestellt und einer ersten Bewertung bzgl. der Verteilung von Filter- und Speicheraufwand unterzogen. In Abschnitt 2.1 wurden Rendezvousknoten beschrieben, welche spezialisierte Vermittlungsknoten darstellen. Diese Rendezvousknoten sind für jeweils verschiedene Ereignistypen zuständig und filtern alle Ereignisse dieses Typs. Dazu kennen sie Profile aller Abonnenten bezüglich des entsprechenden Ereignistyps.

Abschnitt 2.2 betrachtete Strategien der verteilten Auswertung in hierarchischen Netzen. Die verteilte Auswertung in azyklischen Punkt-zu-Punkt-Netzen war Gegenstand des Abschnitts 2.3. Bei der verteilten Auswertung wird der Filteraufwand im gesamten Vermittlungsnetz verteilt. Verschiedene Vermittler sind für die Filterung von Ereignissen zuständig. Entweder werden die Vermittler der Anbieter der Ereignisse genutzt (Profilweiterleitung), die der Abonnenten (Ereignisweiterleitung), oder die gegebene Netzstruktur legt die filternden Vermittler fest (Hierarchische Ansätze).

Die vorgestellten Filterstrategien nutzen verschiedene Ansätze zur Verminderung von Profilredundanzen in den Vermittlern: Äquivalenz nutzt Profile mit der gleichen Menge erfüllender Ereignisse. Eine Erweiterung der Äquivalenz ist das Bedecken, welches allgemein die Mengen erfüllender Ereignisse betrachtet und darauf aufbauend weniger Filteroperationen durchführt. Verschmelzen fügt zur Aufwandsverringerung Filteroperationen mehrerer Profile (bzw. Prädikate) zusammen.

Die vorgestellten Filterstrategien sind sehr spezielle Lösungen für die verteilte Filterung, die als erste Einordnung derartiger Lösungsansätze herangezogen werden können. Eine Einteilung und Unterscheidung der Filterstrategien kann jedoch in weit detaillierterer Art und Weise geschehen. Dadurch können die Ideen der verschiedenen Algorithmen miteinander kombiniert, Konzepte unterschiedlicher Arbeiten zusammen genutzt und so Verbesserungen (z.B. bzgl. der Effizienz oder Netzlast) erzielt werden. Eine genauere Bewertung und der Vergleich verschiedener Algorithmen wird ebenfalls möglich, da nicht vollkommen voneinander unabhängige Lösungen für die verteilte Filterung betrachtet werden. Der Vergleich erstreckt sich vielmehr auf verschiedene Varianten der gleichen Grundidee bzw. auf Ähnlichkeiten zwischen verschiedenen Grundideen. Diese Klassifikation verteilter Filteralgorithmen sowie ausführliche Vergleiche und Bewertungen werden im nächsten Kapitel vorgestellt.

Kapitel 3

Klassifikation verteilter Filteralgorithmen

Inhalt

Nachdem im letzten Kapitel bisherige Ansätze der verteilten Filterung vorgestellt wurden, werden diese jetzt verallgemeinert und miteinander kombiniert. Dazu werden die in unterschiedlichen Arbeiten entwickelten Filterstrategien in verschiedene Kategorien eingeteilt, um einen allgemeineren Blick auf die verteilte Filterung zu erhalten. Insbesondere können dadurch optimierende Ansätze bestimmter Arbeiten mit den Ideen anderer Arbeiten kombiniert werden und zu Verbesserungen führen. Die Gemeinsamkeiten und Unterschiede der verteilten Filteralgorithmen werden ebenfalls deutlich. Eine Bewertung und ein Vergleich der Algorithmen ist so möglich und erfolgt im Verlauf dieses Kapitels.

Aus dem Anwendungsszenario der Gebäudesteuerung ergeben sich als Bewertungskriterien verteilter Filteralgorithmen deren Effizienz und Skalierbarkeit (Effizienz wird benötigt, um eintreffende Ereignisse schnell verarbeiten zu können und Skalierbarkeit, um Filtereffizienz auch bei großen Profilmengen gewährleisten zu können). Spezieller heißt das:

Netzlast: Die Netzlast ist ein wichtiges Bewertungskriterium, da das Verbindungsnetz zwischen den Vermittlern und auch zu den Klienten nur eine begrenzte Übertragungsgeschwindigkeit aufweist. Niedrigere Netzlasten sind somit vorteilhafter für einen verteilten Filteralgorithmus, insbesondere, da die Datenübertragung über ein Netzwerk viel Zeit im Vergleich zu netzknoteninterner Berechnung benötigt. Die Netzlast bei der Filterung der Ereignisse ist kritischer als diejenige beim An- und Abmelden von Profilen einzuschätzen, welche nur einmalig bei Systemstart und Konfigurationsänderungen auftritt.

Speicherverbrauch: Der Speicherverbrauch bei steigender Anzahl der Profile ist ein wichtiges Bewertungskriterium. Auch hier ist eine Beschränkung der Ressourcen gegeben, so dass Algorithmen mit weniger intensiver Speichernutzung zu bevorzugen sind.

Effizienz: Die Effizienz besagt, wie viele Ereignisse in einer Sekunde (oder allg. einer Zeiteinheit) von einem Filteralgorithmus gefiltert werden können (in Bezug auf eine bestimmte Profilmenge). Sie wird unter anderem durch die Netzlast und den Speicherverbrauch beeinflusst. Aber auch andere Kriterien, wie die Anzahl der Filterschritte bis zur Auslieferung einer Benachrichtigung, sind für die Effizienz entscheidend.

Skalierbarkeit: Die Skalierbarkeit ist ein Maß für das Verhalten eines verteilten Filteralgorithmus bei steigender Profilanzahl je Vermittler. Allgemein wird ein Filteralgorithmus bei steigender Profilzahl je Vermittler weniger Ereignisse je Zeiteinheit filtern können. In welchen Größenordnungen sich dieser Effizienzverlust auswirkt ist wichtig für die Einordnung eines Filteralgorithmus. In der anderen Richtung sollte bei gleicher Gesamtprofilanzahl und einer Erhöhung der Vermittlerzahl die Anzahl filterbarer Ereignisse je Zeiteinheit steigen. Die Skalierbarkeit wird teilweise von Netzlast und Speicherverbrauch beeinflusst.

In den folgenden Abschnitten wird die Kategorisierung von Filteralgorithmen vorgenommen. Abschnitt 3.1 untersucht die Kommunikation der Vermittler mit den Abonnenten. In Abschnitt 3.2 wird die Aufteilung des Filteraufwands untersucht. Der Filterort ist Gegenstand des Abschnitts 3.3. Schließlich wird die Speicherstrategie in Abschnitt 3.4 betrachtet. Abschnitt 3.5 kombiniert die Möglichkeiten der verschiedenen Kategorien zu vollständigen Filteralgorithmen und bewertet diese entsprechend der o.g. Kriterien. Abschließend wird in Abschnitt 3.6 die vorgenommene Kategorisierung zusammengefasst.

3.1 Kommunikation mit den Abonnenten

In diesem Abschnitt werden verschiedene Möglichkeiten der Kommunikation von filternden Vermittlern mit den Abonnenten aufgezeigt. Das heißt: Wie werden Benachrichtigungen an einen Abonnenten ausgeliefert? Es existieren drei Möglichkeiten für

einen filternden Vermittler: direkte Kommunikation, Weiterleitung von Benachrichtigungen über das Vermittlungsnetz und transparente Abonnements durch Stellvertreterfunktionen der Vermittler.

3.1.1 Direkte Kommunikation

Hierbei findet zwischen einem filternden Vermittler und einem Abonnenten direkte Kommunikation statt. Eine Benachrichtigung über ein erfüllendes Ereignis wird vom Vermittler an den Abonnenten über eine direkte Netzverbindung gesendet. Andere Vermittler sind nicht in die Benachrichtigung involviert.

Vorteile: Die Benachrichtigungen treffen sofort nach der Filterung beim Abonnenten ein. Damit wird die Verzögerung zwischen dem Auftreten eines Ereignisses und der Benachrichtigung möglichst klein gehalten, da keine Zwischenstationen in die Kommunikation involviert sind.

Nachteile: Damit jeder Vermittler Verbindungen zu allen potentiellen Abonnenten der von ihm gefilterten Ereignisse unterhalten kann, muss vor einer Benachrichtigung diese Verbindung erst aufgebaut werden (offene Verbindungen zu allen Abonnenten sind aus Skalierbarkeitsgründen nicht möglich). Dieser Verbindungsaufbau benötigt Zeit und Ressourcen, so dass die Vorteile der direkten Kommunikation teilweise entfallen. Weiterhin müssen Abonnenten diese Verbindungswünsche von Vermittlern akzeptieren. Dadurch werden sie stärker belastet und fungieren als Server für diesen Dienst der Benachrichtigungsübermittlung. Ebenfalls müssen bei einer Vielzahl von möglichen Verbindungen gute Fehlerbehandlungsmechanismen integriert werden. Falls ein verbindungsloses Kommunikationsprotokoll verwendet wird, ergibt sich das Problem der Zuverlässigkeit des Eintreffens der Benachrichtigungen bei den Abonnenten.

Die direkte Kommunikation findet bei der Ereignisweiterleitung [5] und dem Link State Matching [2] statt.

3.1.2 Weiterleitung über das Vermittlungsnetz

In diesem Ansatz findet die Kommunikation mit den Abonnenten indirekt durch das Netzwerk der Vermittler statt. Eine Benachrichtigung wird an den Nachbarvermittler gesendet, welcher diese direkt an einen lokalen Abonnenten oder indirekt durch einen seiner Nachbarn weiterleiten kann.

Vorteile: Es ist keine Verbindung von filternden Vermittlern zu den nicht lokalen Abonnenten nötig. Dadurch werden sowohl die Abonnenten, als auch die Vermittler entlastet: Abonnenten müssen keine Verbindungen akzeptieren, Vermittler diese nicht aufbauen.

Nachteile: Durch die Weiterleitung über das Vermittlungsnetz wird die Latenzzeit zwischen Filterung eines Ereignisses und Auslieferung einer Benachrichtigung erhöht. Fraglich ist, wie die Bestimmung des Nachbarn erfolgen soll, der eine Benachrichtigung eines Abonnenten ausführen kann. Entweder wird an mehrere Nachbarn gleichzeitig weitergeleitet - dies erhöht den Netzverkehr, oder an die Nachbarn wird nacheinander gesendet - dieses vergrößert die Latenzzeit

erneut. Weiterhin können für alle Ausgangsknoten auch Indizes über den Abonnenten angelegt werden - ein höherer Speicherverbrauch ist die Folge. Letztlich kann auch beim Anlegen eines Profils in einem Vermittler dessen Weg im Netz protokolliert werden, bei der Benachrichtigung wird der entgegengesetzte Pfad verfolgt.

Die Weiterleitung über das Vermittlungsnetz kann bei der Profilweiterleitung [5], in den Systemen Hermes [27] und Scribe [30] sowie der Hierarchie von Vermittlungsknoten [38] genutzt werden.

3.1.3 Transparente Abonnements

In diesem Ansatz übernehmen die Vermittler eine Stellvertreterfunktion für die Abonnenten. Bei der Weiterleitung eines Profils an Nachbarvermittler tritt nicht dessen tatsächlicher Abonnent als Sender auf, sondern der jeweilig weiterleitende Vermittler. Dieser entscheidet dann bei einer Benachrichtigung über deren weitere Versendung, welche an alle Nachbarknoten mit passenden Profilen weitergeleitet werden sollte.

Vorteile: Es muss keine direkte Kommunikation mit Abonnenten stattfinden (außer mit lokalen Abonnenten und den Nachbarvermittlern in ihrer Funktion als lokale Abonnenten). Weiterhin können die Optimierungen zur Verminderung von Profilredundanzen (Abschnitt 2.4) zwischen verschiedenen Abonnenten angewendet werden.

Nachteile: Der erwartete Speicherverbrauch ist hoch, denn es müssen zusätzliche Informationen über den tatsächlichen Abonnenten (u.U. ist dies erneut lediglich ein Stellvertreter) gespeichert werden. Dieser Speicherverbrauch wird jedoch zwischen den Vermittlern aufgeteilt, so dass sich die Belastung im akzeptablen Rahmen bewegt. Ebenfalls vergrößert die Filterung bzw. Bearbeitung von Ereignissen in mehreren Netzknoten die Latenzzeit zwischen dem Auftreten des Ereignisses und einer Benachrichtigung über dieses.

Bei Profilweiterleitung [5] und im System Hermes [27] wird das Nutzen transparenter Abonnements vorgeschlagen.

3.2 Aufteilung des Filteraufwandes

Ein Unterschied zwischen verschiedenen verteilten Filteralgorithmen ist die Aufteilung des Filteraufwands zwischen den Vermittlern. Es existieren zwei Ansätze: exklusive Filterung und verteilte bedingte Filterung.

3.2.1 Exklusive Filterung

Bei der exklusiven Filterung übernehmen Vermittlerknoten die Filterung einer bestimmten Menge von Profilen. Eine Einteilung kann z.B. nach Ereignistypen oder lokalen Abonnenten erfolgen. Nach der Filterung werden die passenden Abonnenten direkt vom Filterknoten oder durch Weiterleitung der Benachrichtigung über das Vermittlungsnetz (Abschnitt 3.1) benachrichtigt.

Vorteile: Vorteil ist eine relativ einfache Realisierung dieses Ansatzes. Weiterhin werden in verschiedenen Vermittlern keine redundanten Profilinformationen gespeichert, was den Speicherverbrauch gering hält.

Nachteile: Problematisch ist das Verhalten bei Benachrichtigungen entfernter Abonnenten. Findet weiterleitende Kommunikation statt, werden für Ereignisse u.U. mehrere Benachrichtigungen zu den gleichen Nachbarvermittlern gesendet. Anderenfalls, bei direkter Kommunikation, müssen für Benachrichtigungen erst Netzverbindungen zu den Abonnenten aufgebaut werden.

Im System Scribe [30] und beim Link State Matching [2] wird exklusive Filterung genutzt. Bei der Ereignisweiterleitung [5] und der Profilweiterleitung [5] kann diese genutzt werden.

3.2.2 Verteilte bedingte Filterung

Bei der verteilten bedingten Filterung wird ein Ansatz der Arbeitsteilung verfolgt. Jeder Vermittler nimmt die Filterungen vor, die nötig sind, um alle Nachbarn mit passenden Profilen zu finden. An diese wird das Ereignis dann weitergeleitet. Ebenfalls werden alle lokalen Profile gefiltert und deren Abonnenten benachrichtigt. Eine Abschwächung stellt die Bestimmung aller Vermittler mit passenden lokalen Abonnenten und die Weiterleitung der Ereignisse an diese Vermittler dar.

Vorteile: Die Aufteilung der Filterlast im gesamten Netz ist ein speicherfreundlicher Ansatz. Ereignisse werden nur an solche Vermittler weitergeleitet, die tatsächlich mit passenden lokalen Abonnenten verbunden sind. So wird die Netzlast minimiert und eine Aufgabenverteilung unter den Vermittlern erreicht.

Nachteile: Bei der verteilten bedingten Filterung werden Ereignisse mehrfach gefiltert. Dadurch wird die Latenzzeit zwischen dem Auftreten eines Ereignisses und dem Eintreffen der Benachrichtigung darüber bei den Abonnenten vergrößert.

Die Profilweiterleitung [5], das System Hermes [27] und die Hierarchie von Vermittlungsknoten [38] nutzen die verteilte bedingte Filterung.

3.3 Filterort

Für den Ort der Filterung existieren verschiedene Möglichkeiten: Filterung bei den Abonnenten oder Anbietern. Weiterhin kann die Filterung bei festgelegten Vermittlern erfolgen. Schließlich können sich auch mehrere Vermittler den Filteraufwand teilen (verteilte bedingte Filterung, Abschnitt 3.2.2).

3.3.1 Filterung bei Abonnenten

Dieser Ansatz verfolgt die Strategie, die Filterung so nahe bei den Abonnenten wie möglich durchzuführen. Dazu werden Ereignisse an alle Vermittler des Vermittlungsnetzes ausgeliefert. Diese filtern dann für ihre lokalen Abonnenten.

Vorteile: Da jeder Vermittler die Filterung für seine lokalen Abonnenten durchführt, gestaltet sich der erzeugte Speicheraufwand entsprechend der Profilanzahl je

Vermittler. Durch eine Beschränkung der Abonnentenzahl je Vermittler ist eine gute Skalierbarkeit gegeben.

Nachteile: Die Ereignisse müssen im gesamten Netz der Vermittler verteilt werden. Da mit hohen Ereignisfrequenzen zu rechnen ist, ist eine starke Belastung des Netzes gegeben.

Beispiele der Filterung bei den Abonnenten sind die Ereignisweiterleitung [5], das Link State Matching [2] und die Hierarchie von Vermittlungsknoten [38].

3.3.2 Filterung bei Anbietern

Hierbei wird die Filterung so nah wie möglich bei den Anbietern durchgeführt. Dazu werden die Profile an alle Vermittler mit entsprechenden Anbietern gesendet. Eine Filterung findet so schon beim lokalen Vermittler eines Anbieters statt.

Vorteile: Die zu erwartenden hohen Ereignisfrequenzen stellen in diesem Ansatz kein Problem dar, da Ereignisse nicht im Netz verteilt werden. Die Filterung wird sofort ausgeführt, wenn ein Ereignis den Benachrichtigungsdienst erreicht.

Nachteile: Problematisch ist die Verteilung der Profile im Netz. Für den Netzverkehr ist jedoch keine große Belastung zu erwarten, da in der Gebäudesteuerung Profiländerungen selten auftreten. Allerdings müssen in allen Vermittlern viele Profile gespeichert und gefiltert werden, was den Platzbedarf in den Vermittlern extrem erhöht.

Ein Beispiel der Filterung bei den Anbietern ist die Profilweiterleitung [5].

3.3.3 Filterung bei willkürlichen Vermittlern

Dieser Ansatz benutzt für die Filterung spezielle Knoten, welche im Netz bekannt sind und von allen lokalen Vermittlern zur Filterung genutzt werden. Beispielsweise wird der filternde Vermittler anhand des Typs eines Ereignisses bzw. Profils beschrieben.

Vorteile: Die Möglichkeit zur Auswahl der filternden Knoten kann leistungsfähigere Rechner mit der Filterung betrauen. Bei zu großer Belastung eines Filterknotens kann dieser selbstständig Unterstützerknoten wählen, transparent für den Rest des Vermittlungsnetzes.

Nachteile: Sowohl Ereignisse als auch Profile müssen an den filternden Knoten gesendet werden. Je nach Struktur des Netzes müssen längere bzw. kürzere Strecken zurückgelegt werden. Damit wird die Netzlast erhöht. Weiterhin gestaltet sich die Speichernutzung u.U. schwierig: Bei Wahl des Filterorts entsprechend dem Typ ist bei starker Nutzung bestimmter Typen mit großen Belastungen der jeweiligen Vermittler zu rechnen.

Die Systeme Scribe [30] und Hermes [27] führen die Filterung abhängig vom Ereignistyp durch.

3.4 Speicherstrategie

Bei verteilter bedingter Filterung und den Maßnahmen zur Verminderung von Profilredundanzen (Bedecken, Äquivalenz und Verschmelzen, Abschnitt 2.4) ergeben sich zwei Strategien hinsichtlich der Speicherung von Profilen. Bei der vorbeugenden Speicherung versuchen Vermittler möglichst viele Profildefinitionen zu speichern, auch wenn die Filterung dieser Profile nicht erforderlich ist, um zu benachrichtigende Nachbarn zu finden. Bei optimistischer Speicherung behalten Vermittler lediglich Informationen über die tatsächlich zu filtern nötigen Profile.

3.4.1 Vorbeugende Speicherung

Dieser Ansatz versucht in Vermittlern Definitionen möglichst vieler Profile zu speichern. Erreicht z.B. ein durch ein Profil des gleichen Abonnenten bedecktes Profil eines nicht lokalen Abonnenten einen Vermittler, so wird auch dieses bedeckte gespeichert, jedoch nur das bedeckende Profil gefiltert. Beim Abmelden des bedeckenden Profils kann die Filterung der ehemals direkt bedeckten dann automatisch erfolgen. Bei Nutzung des Verschmelzens werden die verschmolzenen Profile ebenfalls gespeichert.

Vorteile: Das Abmelden von Profilen hat keine großen Auswirkungen auf den Netzverkehr. Eine alleinige Information über das Abmelden ist ausreichend, da die bedeckten Profile bzw. die Ausgangsprofile des Verschmelzens ebenfalls bekannt sind.

Nachteile: Für die Speicherung vieler Profile ist viel Hauptspeicher notwendig. Temporär redundante Profilinformationen werden in mehreren Vermittlern gespeichert.

3.4.2 Optimistische Speicherung

Dieser Ansatz versucht mit sehr wenig Profildefinitionen eine korrekte Filterung zu erreichen. Es werden nur solche Profile gespeichert, die tatsächlich gefiltert werden. In nicht lokalen Vermittlern werden nur die bedeckenden Profile eines Abonnenten gespeichert bzw. die Verschmelzungen.

Vorteile: Speicherfreundlich gestaltet sich das Verwerfen temporär nicht zu filternder Profile.

Nachteile: Das Abmelden von Profilen hat starken Einfluss auf die Netzlast. Da der Abonnent gleichzeitig mit dem Abmelden eines bedeckenden Profils die ehemals direkt bedeckten Profile verbreiten muss, kann der Netzverkehr dabei stark ansteigen. Nutzt man Verschmelzungen, muss der Vermittler die Ausgangsprofile der Verschmelzung anfordern, oder die Verschmelzung wird imperfekt.

Bei Profilweiterleitung [5] und dem System Hermes [27] sind beide Varianten möglich. Angaben über die verwendete Speicherstrategie wurden in der Literatur jedoch nicht gefunden.

3.5 Verbindung der Kategorien

Aus der in den letzten vier Abschnitten vorgenommen Kategorisierung werden nun verschiedene verteilte Filteralgorithmen kombiniert. Diese werden dann entsprechend der vorher erarbeiteten Kriterien Netzlast, Speicherverbrauch, Effizienz und Skalierbarkeit bewertet.

Tabelle 3.1 zeigt die verschiedenen Kombinationsmöglichkeiten der vier vorgestellten Kategorien. Spalte 1 enthält die Bezeichnung der Algorithmen. Spalte 2 zeigt die drei Möglichkeiten des Filterortes. Für die Verteilung des Filteraufwandes existieren in Spalte 3 zwei Möglichkeiten. Bei verteilter bedingter Filterung erfolgt eine weitere Unterteilung der beiden vorgestellten Speicherstrategien. Spalte 4 zeigt schließlich die drei Möglichkeiten der Kommunikation mit den Abonnenten. Die Bewertungen der verschiedenen Algorithmen sind in Spalte 5 dargestellt.

In Tabelle 3.1 sind drei Hauptarten von Algorithmen zu erkennen: Ereignisweiterleitung (EW), Profilweiterleitung (PW_x) und Rendezvousknoten (RK_x). Diese ergeben sich aufgrund des Filterortes in Spalte 2. In diesen Hauptarten existieren, außer bei der Ereignisweiterleitung, verschiedene Variationen. In den folgenden drei Abschnitten werden die drei Hauptarten von Algorithmen und deren Variationen skizziert und bewertet.

3.5.1 Ereignisweiterleitung

Bei der Ereignisweiterleitung (EW) findet die Filterung so nah bei den Abonnenten wie möglich statt. Profile werden beim lokalen Vermittler angemeldet und nicht weiter verbreitet. Stattdessen werden die Ereignisse an alle Vermittler des Netzes weitergeleitet. Da jeder Vermittler für seine lokalen Abonnenten filtert, ist nur exklusive Filterung möglich. Somit folgt als Kommunikationsstrategie direkte Kommunikation mit den Abonnenten. Bei der Ereignisweiterleitung sind keine Maßnahmen zur Verminderung von Profilredundanzen möglich, da lediglich lokale Vermittler für die Filterung verantwortlich sind. Die Optimierungen können und dürfen für lokale Abonnenten nicht durchgeführt werden (Benachrichtigungen müssen an die tatsächlichen Abonnenten ausgeliefert werden, also muss über deren Profilen gefiltert werden). Der Ansatz der Ereignisweiterleitung lässt sich wie folgt bewerten:

Netzlast: Besonders bei großen Ereignisfrequenzen muss mit sehr hohem Netzverkehr gerechnet werden, da jedes Ereignis an alle Vermittler gesendet wird. Allgemein wird durch das vorgenommene Fluten von Ereignissen die maximal mögliche Netzlast erzeugt.

Speicherverbrauch: Jeder Vermittler filtert nur für seine lokalen Abonnenten, der Speicherverbrauch ist somit relativ gering. Durch eine Beschränkung der Abonnentenmenge (und deren Profilanzahl) je Vermittler kann der Speicherverbrauch entsprechend den vorhanden Ressourcen direkt gesteuert werden.

Effizienz: Alle Ereignisse werden an jeden Vermittler gesendet. Dieser Kommunikationsaufwand stellt eine erste Effizienzminderung dar. Eine zweite Minderung erzeugt das Filtern jedes Ereignisses in jedem Vermittler. Die Verteilung des Dienstes führt somit in Hinblick auf die Entlastung der Vermittler zu keinem

Algorithmus	Filterort			Verteilung Filteraufwand (Speicherstrategie)			Kommunikation			Bewertung			
	Abonnenten	Anbieter	willkürlich	exklusiv	verteilt bedingt: vorbeugende Speicherung	verteilt bedingt: optimistische Speicherung	direkt	Weiterleitung	transparent	Netzlast	Speicherverbrauch	Effizienz	Skalierbarkeit
EW	×			×			×			−−	++	+−	−
PW_1		×		×			×			+	−−	+−	−−
PW_2		×		×				×		+−	−−	+−	−−
PW_3		×			×		×			+	−	+	−
PW_4		×			×			×		+−	−	+	−
PW_5		×			×				×	+	+−	++	+−
PW_6		×				×	×			+	+−	+	+−
PW_7		×				×		×		+−	+−	+	+−
PW_8		×				×			×	+	+	++	+
RK_1			×	×			×			−	−−	−	−−
RK_2			×	×				×		−	−−	−	−−
RK_3			×		×		×			+−	−−	+−	−
RK_4			×		×			×		−	−−	+−	−
RK_5			×		×				×	+−	−	+	+−
RK_6			×			×	×			+−	−	+−	−
RK_7			×			×		×		−	−	−	−
RK_8			×			×			×	+−	+	+	+−

Tabelle 3.1: Aufstellung möglicher verteilter Filteralgorithmen und deren Bewertung (EW - Ereignisweiterleitung, PW_x - Profilweiterleitung, RK_x - Rendezvousknoten).

Vorteil. Die Einfachheit des Algorithmus (keine aufwendigen Berechnungen) ist jedoch als Pluspunkt anzusehen.

Skalierbarkeit: Die Hinzunahme von Vermittlern bei gleicher Gesamtprofilanzahl führt zu keiner Steigerung der filterbaren Ereignisse je Zeiteinheit. Eine starke Verminderung ist jedoch auch nicht zu erwarten. Grund ist, dass jeder Vermittler alle Ereignisse filtern muss. Weiterhin steigt die Netzlast in diesem Fall weiter an, so dass das Vermittlungsnetz stärker belastet wird. Dieses ist nur bis zu einer bestimmten Last möglich.

3.5.2 Profilweiterleitung

Bei den verschiedenen Varianten der Profilweiterleitung (PW_x) findet die Filterung der Ereignisse so nah bei den Anbietern wie möglich statt. Die Profile werden zu mehreren Vermittlern weitergeleitet, welche eine Filterung durchführen können.

In den Ansätzen PW_1 und PW_2 kennen alle Vermittler alle Profile und filtern eintreffende Ereignisse einmalig. In PW_1 wird zur Benachrichtigung mit den Abonnenten Kontakt aufgenommen, in PW_2 leitet das Vermittlungsnetz die Benachrichtigungen weiter. Transparente Kommunikation steht im Widerspruch zur vollzogenen exklusiven Filterung.

PW_3 bis PW_8 verfolgen eine Strategie der Arbeitsteilung bei der Filterung. Dazu werden die Profile im Netz durch die Vermittler an die Nachbarvermittler verteilt. Daher können die Optimierungen zur Verminderung von Profilredundanzen genutzt werden. PW_3 und PW_6 verteilen die Profile mit den tatsächlichen Abonnenten als Quelle. Zur Benachrichtigung nimmt der filternde Vermittler mit dem lokalen Vermittler der Abonnenten direkten Kontakt auf. Dieser muss eine Nachfilterung durchführen (aufgrund der Optimierungen). Die Optimierungen zur Verminderung von Profilredundanzen geschehen auf Abonnentenbasis - verschmolzen werden Profile eines Abonnenten, ebenso werden Bedeckungen und Äquivalenzen ausgenutzt. Der Unterschied in PW_4 und PW_7 ist, dass zur Benachrichtigung kein Kontakt mit den lokalen Vermittlern aufgenommen wird, sondern das Vermittlungsnetz die Benachrichtigungen weiterleitet.

In PW_5 und PW_8 treten die Vermittler als Abonnenten auf und verbreiten die Profile als ihre Filteroperationen an ihre Nachbarvermittler. Somit filtert jeder Vermittler nur für lokale Abonnenten (Klienten bzw. Vermittler). Optimierungen werden für gleiche Abonnenten eingesetzt, in diesem Fall also auch für unterschiedliche tatsächliche Abonnenten.

PW_3 bis PW_5 speichern u.U. Profile, welche nicht gefiltert werden - vorteilhaft beim Abmelden von Profilen. PW_6 bis PW_8 speichern nur zu filternde Profile, was beim Abmelden mehr Netzverkehr verursacht (durch Weiterleiten von äquivalenten und bedeckten Profilen bzw. Ausgangsprofilen von Verschmelzungen). Die verschiedenen Varianten der Profilweiterleitung werden wie folgt bewertet:

Netzlast: In PW_2 werden u.U. mehrere Benachrichtigungen über ein Ereignis über die gleichen Vermittler gesendet, was die Netzlast erhöht. Bei PW_1, PW_3 und PW_6 entfällt dies, die Folge ist weniger Netzlast. Bei PW_4 und PW_7 tritt das gleiche Problem der doppelten Benachrichtigungen auf. PW_5 und PW_8 benachrichtigen Nachbarvermittler genau dann (und zwar einmalig), wenn einer von

deren Nachbarn lokaler Vermittler für passende Profile oder Abonnent ist. Es handelt sich um eine Variante mit niedriger Netzlast.

Speicherverbrauch: In PW_1 und PW_2 ist der Speicherverbrauch sehr hoch, da die Vermittler alle Profile aller Abonnenten filtern. PW_3 bis PW_5 haben etwas höheren Speicherbedarf als PW_6 bis PW_8, da mehr Profile gespeichert werden. PW_5 und PW_8 vermindern den Speicherverbrauch, da Profile mehrerer tatsächlicher Abonnenten verschmolzen bzw. Überdeckungen ausgenutzt werden können. PW_3, PW_4, PW_6 und PW_7 nutzen dies hingegen nur bei jeweils gleichen Abonnenten.

Effizienz: In PW_1 und PW_2 ist die Effizienz nur mäßig, da in einem Vermittler alle Profile gefiltert werden. PW_3, PW_4, PW_6 und PW_7 nutzen Optimierungen auf Abonnentenebene aus und verbessern damit die Effizienz (Filterung über weniger Profilen in den Vermittlern). PW_5 und PW_8 filtern genau die Profile, die nötig sind, um einen Nachbarvermittler zu benachrichtigen, gute Effizienz ist die Folge.

Skalierbarkeit: Die Skalierbarkeit ist bei PW_1 und PW_2 sehr schlecht, da eine Hinzunahme von Vermittlern den Filteraufwand nicht verringert. PW_6 bis PW_8 erreichen bei gleicher Kommunikationsstrategie bessere Skalierbarkeit als PW_3 bis PW_5, da weniger Vermittler nötig sind, um die kleinere Profilmenge zu verwalten. In PW_5 und PW_8 sind bei gleicher Verteilungsstrategie bessere Skalierbarkeiten gegeben, da Optimierungen über allen Profilen eines Teilnetzes durchgeführt werden und bei größerer Profilanzahl u.U. mehr Redundanzen vorhanden sind und ausgenutzt werden (und damit die Filterung über weniger Profilen stattfindet und weniger Speicher genutzt wird).

3.5.3 Rendezvousknoten

Bei den Varianten der Rendezvousknoten (RK_x) findet die Filterung in spezialisierten Knoten statt, beispielsweise abhängig vom Ereignistyp.

Bei RK_1 bis RK_3 kennen nur die spezialisierten Rendezvousknoten die entsprechenden Profile. Die Benachrichtigung kann dabei entweder direkt (RK_1) oder über das Vermittlungsnetz (RK_2) erfolgen.

Bei verteilter bedingter Filterung (RK_3 bis RK_8) sind beide Speicherstrategien und die Optimierungen zur Verminderung von Profilredundanzen möglich. Alle Vermittler, die auf dem Weg eines Profils zum Rendezvousknoten besucht werden, speichern dessen Definition und berücksichtigen es bei der Filterung. Bei RK_5 und RK_8 treten Vermittler als Abonnenten auf. Ereignisse werden zum Rendezvousknoten gesendet, alle Vermittler auf diesem Weg führen eine Filterung durch und benachrichtigen passende Vermittler (außer in Richtung des Rendezvousknotens) bzw. lokale Abonnenten. Bei RK_3 und RK_6 treten die Vermittler nicht als Abonnenten auf, die Benachrichtigung erfolgt durch den filternden Vermittler zu den lokalen Vermittlern der Abonnenten. Bei RK_4 und RK_7 leitet das Vermittlungsnetz die Benachrichtigungen weiter. Die Bewertung der verschiedenen Varianten der Rendezvousknoten gestaltet sich wie folgt:

Netzlast: Bei RK_1 und RK_2 ist die Netzlast hoch, da alle Ereignisse zum Rendezvousknoten gesendet werden und danach die Benachrichtigungen, u.U. mehrere auf

dem gleichen Weg (RK_2), ausgeliefert werden. Dieses ist ebenfalls bei RK_4 und RK_7 der Fall. Bei den anderen Varianten entfällt der Aufwand des mehrmaligen Weiterleitens von Benachrichtigungen über das gleiche Ereignis. Bei der Weiterleitung der Ereignisse ist die Netzlast proportional zur Entfernung zwischen dem lokalen Vermittler des Anbieters und dem entsprechenden Rendezvousknoten.

Speicherverbrauch: Da bei RK_1 und RK_2 die spezialisierten Vermittlungsknoten alle Profile speichern, wird sehr viel Speicher benötigt. RK_3 bis RK_5 benötigen bei gleicher Verteilung des Filteraufwandes etwas mehr Speicher als RK_6 bis RK_8. RK_5 und RK_8 zeigen einen geringeren Speicherverbrauch, da Profile mehrerer Abonnenten verschmolzen bzw. Überdeckungen ausgenutzt werden können. RK_3, RK_4, RK_6 und RK_7 nutzen dies hingegen nur bei jeweils gleichen Abonnenten.

Effizienz: In RK_1 und RK_2 ist die Effizienz nur mäßig, da die Rendezvousknoten alle Profile eines Ereignistypen filtern. RK_3, RK_4, RK_6 und RK_7 nutzen Optimierungen auf Abonnentenebene aus und verbessern damit die Effizienz durch weniger Filteraufwand. In RK_5 und RK_8 filtern Rendezvousknoten die Profile, die nötig sind, um ihre Nachbarnvermittler zu benachrichtigen, weniger Filteraufwand je Vermittler (also bessere Effizienz) ist die Folge. Insgesamt ist die Effizienz jedoch schlechter als bei Profilweiterleitung, da der Rendezvousknoten ständig starke Belastung erfährt.

Skalierbarkeit: Bei RK_1 und RK_2 ist eine sehr schlechte Skalierbarkeit zu erwarten, da bei steigender Profilzahl die Rendezvousknoten mehr Profile verarbeiten müssen. RK_3, RK_4, RK_6 und RK_7 zeigen eine schlechte Skalierbarkeit, da mehr Profile die Rendezvousknoten stärker belasten und Optimierungen zur Verminderung von Profilredundanzen nur auf Abonnentenbasis durchgeführt werden. RK_5 und RK_8 zeigen eine etwas bessere Skalierbarkeit, da Optimierungen auf Ebene der Vermittler mehr Redundanzen unter den Profilen ausnutzen und damit die Filterung über weniger Profilen durchführen. Der Flaschenhals der Rendezvousknoten bleibt auch bei Hinzunahme von Vermittlern bestehen, da Rendezvousknoten alle Ereignisse eines Typs filtern und speichern müssen.

3.6 Zusammenfassung

In diesem Kapitel wurde eine umfangreiche Kategorisierung und Bewertung verteilter Filteralgorithmen vorgenommen. Zur Charakterisierung dieser Algorithmen wurden vier Kategorien eingeführt:

1. Die Kommunikation mit den Abonnenten kann direkt, durch das Vermittlungsnetz oder stellvertretend durch die Vermittler (transparente Abonnements) geschehen (Abschnitt 3.1).

2. Der Filteraufwand kann im gesamten Vermittlungsnetz verteilt werden oder exklusiv bestimmten Vermittlern zugeordnet werden (Abschnitt 3.2).

3. Für den Ort der Filterung ergeben sich die Möglichkeiten der Filterung bei den Abonnenten, bei den Anbietern oder bei willkürlich ausgewählten Vermittlern (Abschnitt 3.3).

4. Die Speicherstrategie bei der Verteilung des Filteraufwandes kann optimistisch oder vorbeugend gewählt werden (Abschnitt 3.4).

In Abschnitt 3.5 wurden die verschiedenen Umsetzungsmöglichkeiten der eingeführten Kategorien kombiniert. Es ergeben sich 17 verschiedene Varianten zur Implementierung von verteilten Filteralgorithmen, dabei existieren 3 Hauptarten dieser Algorithmen. Die Varianten der Algorithmen wurden danach einer Bewertung unterzogen. Als Kriterien wurden die erzeugte Netzlast, der benötigte Speicher, die Effizienz und die Skalierbarkeit herangezogen.

Aus den 3 Hauptarten von verteilten Filteralgorithmen wird nun der in Hinblick auf die vier Bewertungskriterien beste Algorithmus für eine praktische Umsetzung und Untersuchung ausgewählt. Durch die vorgenommene Kategorisierung und Bewertung ist diese Auswahl der Algorithmen mit den jeweils besten Eigenschaften leicht umsetzbar. Es ergeben sich die Algorithmen EW (im Folgenden Ereignisweiterleitung genannt), PW_8 (im Folgenden als Profilweiterleitung bezeichnet) und RK_8 (im Folgenden Rendezvousknoten genannt).

Durch eine zusätzliche praktische Untersuchung der verschiedenen Filteralgorithmen wird eine detailliertere Bewertung der drei Hauptarten verteilter Filteralgorithmen möglich. In den folgenden Kapiteln wird daher zuerst die Umsetzung aller drei Filteralgorithmen in der Implementierung des Prototypen DAS und danach dessen umfangreiche Evaluation beschrieben.

Kapitel 4

Systementwurf

Inhalt

Nachdem am Ende von Kapitel 3 drei verteilte Filteralgorithmen für eine praktische Umsetzung ausgewählt wurden, wird in den beiden folgenden Kapiteln deren Realisierung im Prototyp DAS dargestellt. Dieses Kapitel beschreibt den Systementwurf und die generelle Systemarchitektur von DAS. Im nächsten Kapitel werden dann die Filteralgorithmen und die Umsetzung der Optimierungen zur Verminderung von Profilredundanzen genauer erläutert.

Der Prototyp DAS baut auf der Implementierung PrimAS [3] eines zentralisierten Filteralgorithmus auf. Dessen Architektur und die an der zentralisierten Filterkomponente vorgenommenen Erweiterungen werden in Abschnitt 4.1 beleuchtet. Abschnitt 4.2 beschreibt die in DAS genutzte Profildefinitionssprache sowie die unterstützten Wertebereiche und Filteroperatoren. Annahmen und Einschränkungen des gewählten Netzwerks werden in Abschnitt 4.3 dargestellt. Abschnitt 4.4 schließlich widmet sich der Systemarchitektur von DAS. Dazu werden sowohl eine schematische Übersicht gegeben als auch vereinfachte UML-Diagramme dargestellt. Zunächst werden der zentralisierte Filteralgorithmus und dessen Erweiterung beschrieben.

4.1 Zentralisiertes System

Im Rahmen einer vorausgegangenen Studienarbeit [3] entstand der zentralisierte Filteralgorithmus PrimAS für die Nutzung in einem Benachrichtigungssystem. Dieser in Java implementierte Algorithmus wird als Grundlage für die Implementierung des Systems DAS (ebenfalls in Java implementiert) verwendet. Dadurch kann man sich zum einen auf die Implementierung der verteilten Filterung fokussieren und zum anderen ist mit PrimAS bereits ein effizienter zentralisierter Filteralgorithmus vorhanden. Eine Neuentwicklung oder lediglich exemplarische Implementierung der zentralisierten Filterkomponente in einem Vermittler wird damit nicht benötigt. Vielmehr kann die effiziente Umsetzung der Filterung von PrimAS auf ein verteiltes System erweitert werden. Im Folgenden wird zuerst die Filterung in PrimAS beschrieben. Abschnitt 4.1.2 zeigt dann Erweiterungen der bisherigen Implementierung von PrimAS, welche in einer Speicherplatzreduktion und einer dynamischen Filterstruktur (PrimAS ist ein statischer Filteralgorithmus) resultieren.

4.1.1 Bisheriges System

Grundlage des Filteralgorithmus PrimAS sind die Arbeiten von Gough und Smith [9], welche eine baumbasierte Filterung der Ereignisse vorschlagen. Dazu werden die Prädikate der Profile attributweise in einen ereignistypabhängigen Baum eingefügt. Jede Ebene des Baumes ist für die Filterung eines Attributes zuständig. Abb. 4.1(a) zeigt einen Filterbaum für fünf Profile mit den drei Attributen a_3, a_4 und a_5 des Typs T_4. Ebene 1 des Baumes filtert Attribut a_3, Ebene 2 das Attribut a_4 und Ebene 3 Attribut a_5. Die 4. Ebene bilden die Blätter des Baumes, in denen die Profile eingetragen werden. Zum Filtern von Ereignissen wird, ausgehend von der Baumwurzel, die Kante des aktuellen Knotens verfolgt, die zum Attributwert des Ereignisses passt. Danach wird das Attribut des Folgeknotens auf dem Ereignis ausgewertet. Erreicht man ein Blatt, sind in diesem die passenden Profile eingetragen. Anderenfalls existieren keine passenden Profile. Da Profile nicht alle Attribute spezifizieren müssen, existieren für diesen Fall *-Kanten im Baum. Diese werden beim Filtern verfolgt, falls keiner der Werte der Kanten zum Wert des Attribut-Wert-Paares eines Ereignisses passt.

Die Implementierung der Studienarbeit beschreibt die Kanten des Baumes nicht mit Attributwerten, sondern mit Bereichen von Attributwerten. Dadurch können problemlos neben der Gleichheit weitere Operatoren unterstützt werden (siehe Abschnitt 4.2.2). Die Werte der Kanten sind durch ein Feld Arr (der Größe $|Arr|$) von Elementen realisiert. Der Eintrag Arr_i des Index i beschreibt dabei das halboffene Intervall der Werte $]Arr_{i-1}, Arr_i]$ bzw. falls $i = 0$, dann $]-\infty, Arr_i]$. Die Verweise der Kanten werden in einem Feld $\widetilde{Arr}$ der gleichen Größe gespeichert. Der Verweis des Index i, $\widetilde{Arr}_i$, ist dabei dem Intervall, beschrieben durch Arr_i, zugeordnet. Eine genauere Beschreibung der Filterstruktur ist in der Studienarbeit [3] zu finden.

Das Problem eines baumbasierten Ansatzes ist sein großer Hauptspeicherbedarf. Deshalb wurde im Rahmen der Studienarbeit ein erweiterter Algorithmus entwickelt, der weniger Hauptspeicher als der von Gough und Smith [9] vorgeschlagene Algorithmus benötigt. Im Kontrast zur besseren Speicherausnutzung wird jedoch die Filterzeit erhöht. Dieser Algorithmus baut keine Baumstruktur auf, sondern erzeugt einen Filterknoten für jedes Attribut. Abb. 4.1(b) zeigt diese Filterstruktur bei den drei Attributen des Typs T_4. Als Vereinfachung sind die Kanten lediglich mit Werten beschrieben, die

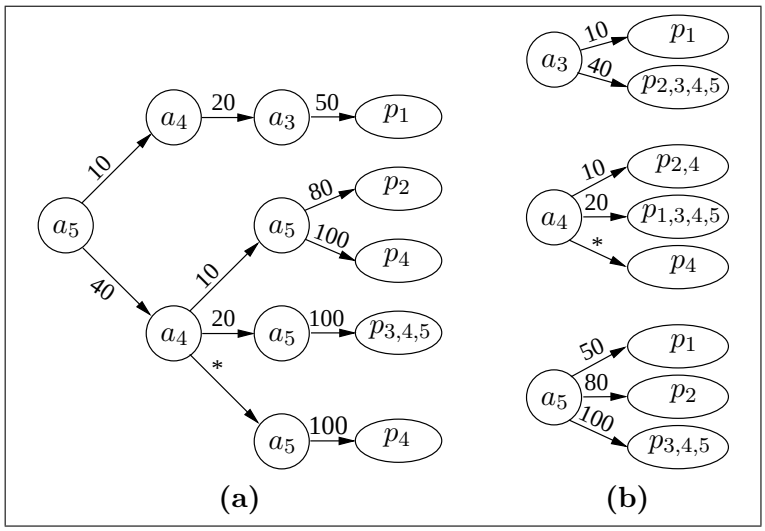


Abbildung 4.1: Filterbaum nach Gough und Smith mit den fünf Profilen $p_1 : (a_3 = 10 \wedge a_4 = 20 \wedge a_5 = 50, T_4)$, $p_2 : (a_3 = 40 \wedge a_4 = 10 \wedge a_5 = 80, T_4)$, $p_3 : (a_3 = 40 \wedge a_4 = 20 \wedge a_5 = 100, T_4)$, $p_4 : (a_3 = 40 \wedge a_5 = 100, T_4)$, $p_5 : (a_3 = 40 \wedge a_4 = 20 \wedge a_5 = 100, T_4)$.

Erweiterung auf Intervalle ist nicht dargestellt. Um ein Ereignis zu filtern, werden alle Filterknoten der Attribute des Ereignistyps abgearbeitet. Dazu wird der jeweils passenden Kante gefolgt (realisiert durch die Felder Arr und $\widetilde{Arr}$). Die dabei erhaltenen Profilmengen in den Blättern werden dann sukzessive geschnitten. Das Resultat ist die Menge der zu allen Attributen passenden Profile. In Abb. 4.1(b) erhält man beim Filtern des Ereignisses $e_1 : (a_3 = 40, a_4 = 10, a_5 = 80, T_4)$ folgende Profilmenge: $\{p_2, p_3, p_4, p_5\} \cap \{p_2, p_4\} \cap \{p_2\} = \{p_2\}$.

4.1.2 Systemerweiterungen

Im Rahmen der Studienarbeit zum Filteralgorithmus PrimAS wurde ein Verfahren zur weiteren Minimierung des Speicherbedarfs vorgeschlagen. Dieser Ansatz und seine Umsetzung in DAS wird im Folgenden beschrieben. Danach wird die Erweiterung des statischen Filteralgorithmus PrimAS zu einer dynamischen Filterstruktur erläutert. Damit wird das einzelne, nicht gleichzeitige An- und Abmelden aller Profile möglich. Zuerst wird auf das Verfahren zur Verbesserung des Speicherbedarfs eingegangen.

Bitliste

Die in der Studienarbeit [3] vorgeschlagene Methode, um den Hauptspeicherverbrauch weiter zu senken, beruht auf der Verwaltung der Profile in den Blättern der Filterstruktur. Es sollen nicht Profile in Blättern gespeichert werden, sondern Indizes der Profile in Form einer Bitliste. Diese Bitliste wurde im System DAS umgesetzt. Dazu wird in der Filterstruktur je Ereignistyp ein Feld aller Profile dieses Typs angelegt. In den Blättern der Filterknoten wird eine Bitliste verwaltet (umgesetzt als Feld von

Integers), welche die entsprechenden Profile markiert, anstatt auf diese zu verweisen. Folgende Beispiele verdeutlichen den Speicherverbrauch bei beiden Varianten:

Beispiel 4.1 (Hoher Speicherbedarf bei Profilverweisen) *Es seien* 1.000 *Profile des Typs* T_5 *mit* $|T_5| = 10$ *zu verwalten. Bei gleichverteilten Operatoren und einer Wertebereichgröße von* 10.000 *Elementen ergeben sich durchschnittlich* 700 *Ausgangskanten je Attributknoten mit ca.* 700 *Profilverweisen je Kante (Mittelwerte bei zufälliger Erzeugung der Profile).*

Speicherverbrauch bei Profilverweisen: *Kantenzahl * Profile pro Kante * Speicher pro Profilverweis * Attributzahl* $= 700 * 700 * 4 * 10 = 18,69$ *MB.*

Speicherverbrauch mit Bitliste: *Speicher für Feld aller Profile + Anzahl der Kanten * Speicher je Bitliste * Attributzahl* $= (1.000*4)+700*\frac{1.000}{8}*10 = 858,40$ *KB.*

Bei anderen Profilkonfigurationen kann jedoch ein gegenteiliger Speicherverbrauch entstehen, wie dieses Beispiel zeigt:

Beispiel 4.2 (Hoher Speicherbedarf bei Nutzung einer Bitliste) *Es seien erneut* 1.000 *Profile des Typs* T_5 *mit* $|T_5| = 10$ *zu verwalten. Jedes Prädikat in jedem Profil ist eindeutig und benutzt den Gleichheitsoperator (wird also in keinem anderen Profil genutzt).*

Speicherverbrauch bei Profilverweisen: *Kantenzahl * Profile pro Kante * Speicher pro Profilverweis * Attributzahl* $= 1.000 * 1 * 4 * 10 = 39,06$ *KB.*

Speicherverbrauch mit Bitliste: *Speicher für Feld aller Profile + Anzahl der Kanten * Speicher je Bitliste * Attributzahl* $= (1.000*4)+1.000*\frac{1.000}{8}*10 = 1,20$ *MB.*

Allgemein ist bei geringer Profilanzahl je Ausgangskante eine Speicherung der direkten Verweise im Hinblick auf den Speicherverbrauch sinnvoller als die Bitliste. Sind $3,125\%$ der Gesamtprofilzahl in einem Blatt gespeichert, so ist das Nutzen der Bitliste besser, da ein Verweis 4 Byte benötigt, eine Markierung hingegen 1 Bit. Zusätzlich wird noch Speicher für die Verweise auf alle Profile benötigt.

Es zeigen sich ebenfalls Unterschiede beim Schneiden der Profilmengen in den Blättern der Filterstruktur. Bei großer Auslastung (Anzahl Profilverweise je Blatt) ist ein Schneiden der Bitlisten effizienter, da das Schneiden durch logische Operatoren möglich ist. Anderenfalls sind die direkten Verweise zu bevorzugen.

Abb. 4.2 zeigt den Aufwand für das Schneiden von Profilmengen in Abhängigkeit von der Gesamtprofilanzahl. In den Schnittmengen sind jeweils 1.000, 3.000 bzw. 5.000 zufällige Profile enthalten. Je mehr Elemente pro Menge vorhanden sind, desto mehr Aufwand ist bei der Nutzung von Verweisen notwendig. Bei Nutzung der Bitliste ist die Größe der Profilmengen nicht relevant für die Zeit zum Schneiden (da die Länge der Bitliste von der Gesamtprofilanzahl abhängt). Bis zu einer von der Größe der Schnittmenge abhängenden Profilzahl ist die Nutzung von Verweisfeldern beim Schneiden langsamer. Bei mehr Profilen zeigt die Bitliste schlechtere Werte. Bei 5.000 Einträgen in den zu schneidenden Mengen ist die Bitliste bei den betrachteten Profilzahlen jedoch stets effizienter.

Aus dieser Abhängigkeit der Speicherauslastung und der Effizienz beim Schneiden der Profilmengen wurde der Einsatz einer Bitliste bzw. direkter Profilverweise konfigurierbar gestaltet. Der Nutzer kann angeben, ab welcher Auslastung in den Blättern eine

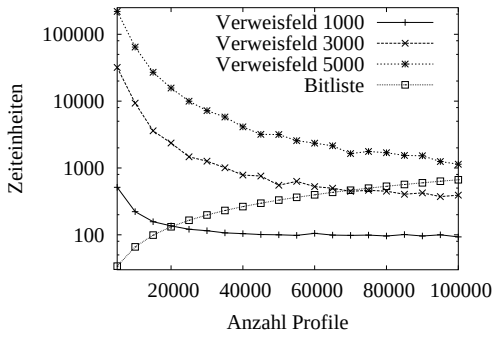


Abbildung 4.2: Zeit für das Schneiden.

Bitliste genutzt wird. Dadurch können die Vorteile beider Ansätze genutzt werden. Die Filterzeit und die Speicherauslastung sind dadurch geringer als beim ursprünglichen Ansatz im System PrimAS.

Dynamische Filterstruktur

Im System PrimAS [3] wird eine statische Filterstruktur erzeugt. Das bedeutet, dass die Nutzung des Systems nur in folgender Reihenfolge geschehen kann:

1. alle Profile erzeugen
2. aus allen Profilen statische Filterstruktur aufbauen
3. Ereignisse filtern

Sollen in PrimAS nach Schritt 2 weitere Profile hinzugefügt oder Profile aus der Filterstruktur entfernt werden, ist ein vollständiger Neuaufbau der Struktur notwendig. In einem verteilten Benachrichtigungssystem ist eine solche Nutzung nicht möglich, da Profile nicht gleichzeitig beim System angemeldet werden können. Im Anwendungsfall der Gebäudeverwaltung ist die Möglichkeit zum Entfernen und Hinzufügen von Profilen vorteilhaft, da Konfigurationsänderungen in den Gebäuden problemlos während des laufenden Betriebes möglich sind. Das System DAS unterstützt deshalb eine dynamische Änderung und Anpassung der Filterstruktur in den Vermittlern.

Im System PrimAS ist eine verteilungsabhängige Anpassung der Filterstruktur zur Zeit des Aufbaus möglich. Dabei werden die Profile entsprechend der in Zusammenarbeit mit Hinze [11] vorgeschlagenen Heuristiken analysiert und die Auswertungsreihenfolge der Knoten angepasst. Das Schneiden der Profilmengen kann dann u.U. früher abbrechen, ein Schnitt der Mengen aller Knoten wird verhindert. Im System DAS ist diese Optimierung der Auswertungsreihenfolge jederzeit möglich. So kann nach Hinzufügen bzw. Entfernen von Profilen eine Optimierung erfolgen.

Die dynamische Filterstruktur wird in DAS durch dynamische Felder umgesetzt. Diese werden zum einen für die Kanten der Filterknoten (mit den Bereichen von Attributwerten) genutzt und zum anderen in den Blättern, sowohl bei den Profilverweisen als auch in den Bitlisten. Zur Laufzeit können die Felder vergrößert bzw. verkleinert werden. Dieses kann durch den Nutzer frei konfiguriert werden:

Leerfelder: Anzahl der möglichen unbenutzten Einträge in einem Feld, ohne dass eine Umordnung der Elemente nötig ist. Ist diese Zahl überschritten, wird das Feld zusammengeschoben und verkürzt.

Änderung: Angabe der Vergrößerung bzw. Verkleinerung des Feldes bei Über- bzw. Unterlauf.

Entsprechend den Eigenschaften des Systems können diese Parameter so gewählt werden, dass sich ein minimaler Aufwand ergibt: Werden zahlreiche Änderungen der Profile vorgenommen, sollten die Zahl der Leerfelder und die Änderungen groß gewählt werden.

4.2 Profildefinitionssprache

Nachdem der zentralisierte Filteralgorithmus im vorigen Abschnitt beschrieben wurde, wird nun die in DAS genutzte Profildefinitionssprache, welche im Rahmen von Königs Diplomarbeit [17] entwickelt wird, mit den unterstützten Wertebereichen und Operatoren dargestellt. Die Profildefinition für ein Profil $p : (a_1 = 5 \wedge a_2 > 10, T_1)$ wird darin folgendermaßen formuliert:

```
PROFILE(PRI(a1=5,a2>10,TYPE=T1))
```

Weitere Möglichkeiten der Profildefinitionssprache werden im Rahmen dieser Arbeit bisher nicht benötigt. In Abschnitt 7.2 werden kurz zusammengesetzte Profile eingeführt, welche von dieser Profildefinitionssprache [17] in vielerlei Hinsicht unterstützt werden. Im Folgenden werden die Wertebereiche und die unterstützten Filteroperatoren beschrieben.

4.2.1 Wertebereiche

Im Anwendungsgebiet der Gebäudeverwaltung sind die verwendeten Wertebereiche für Attribute ausschließlich numerischen Typs [18]. Deshalb werden hauptsächlich solche Bereiche von DAS unterstützt. Zusätzlich besteht die Möglichkeit beliebige Aufzählungen zu definieren. Die Beschreibung von Wertebereichen in DAS geschieht wie folgt:

Ganze Zahlen: Ein Intervall $[a, b]$ ist für die Definition von ganzen Zahlen ausreichend.

Gebrochene Zahlen: Ein Intervall $[a, b]$ und die Anzahl der Nachkommastellen n muss zur Beschreibung angegeben werden.

Aufzählungen: Zur Beschreibung muss eine endliche Anzahl von Elementen (als Zeichenkette) definiert und eine totale Ordnung auf diesen angegeben werden.

Durch die o.g. Definitionen der Wertebereiche sind nur endliche, geordnete Wertebereiche möglich, welche eine Voraussetzung des in 4.1.1 beschriebenen Filteralgorithmus sind.

4.2.2 Operatoren

In den Prädikaten der Profile werden fünf verschiedene Vergleichsoperatoren zur Auswertung auf Attributwerten unterstützt. Diese sind wie folgt definiert:

Gleichheit $(=)$**:** Der Attributwert eines Ereignisses muss genau dem Vergleichswert des Prädikates eines Profils entsprechen. Formal: $(a, w_x) \succ (a, =, w_y)$ gdw. $w_x = w_y$.

Größer $(>)$**:** Der Attributwert eines Ereignisses muss größer als der Vergleichswert des Prädikates eines Profils sein. Durch die Endlichkeit der Wertebereiche kann dadurch auch der Operator Größer oder Gleich ausgedrückt werden. Formal: $(a, w_x) \succ (a, >, w_y)$ gdw. $w_x > w_y$.

Kleiner $(<)$**:** Der Attributwert eines Ereignisses muss kleiner als der Vergleichswert des Prädikates eines Profils sein. Durch die Endlichkeit der Wertebereiche kann dadurch auch der Operator Kleiner oder Gleich ausgedrückt werden. Formal: $(a, w_x) \succ (a, <, w_y)$ gdw. $w_x < w_y$.

Mengentest $(\in)$**:** Der Attributwert eines Ereignisses muss in der Vergleichsmenge des Prädikates eines Profils enthalten sein. Formal: $(a, w_x) \succ (a, \in, W_y)$ gdw. $w_x \in W_y$.

Bereichstest $(\dashv)$**:** Der Attributwert eines Ereignisses muss zwischen den beiden Vergleichswerten des Prädikates des Profils liegen. Formal: $(a, w_x) \succ (a, \dashv, (w_y, w_z))$ gdw. $w_z \geq w_x \geq w_y$.

Um die Operatoren Größer und Kleiner in allen Fällen einsetzen zu können, existieren virtuelle minimale w_{min} bzw. maximale Elemente w_{max} der Wertebereiche. Dadurch können die gesamten Wertebereiche mit diesen beiden Operatoren angefragt werden (die Prädikate $(a, >, w_{min})$ bzw. $(a, <, w_{max})$ werten auf allen Attribut-Wert-Paaren (a, w) mit *Wahr* aus).

4.3 Netzwerk

Nachdem Filteralgorithmus und Profildefinitionssprache beschrieben wurden, werden nun Annahmen und Entscheidungen bzgl. Netztopologie und verwendeter Protokolle der Netzwerk- und Transportschicht behandelt. Generell wird im Rahmen dieser Arbeit von einem stabilen und ausfallsicheren Verbindungsnetz ausgegangen. Deshalb brauchen in die Kommunikationsalgorithmen zwischen den Netzknoten keine umfangreichen Fehlerbehandlungsmechanismen integriert zu werden. Solche Betrachtungen können in eine spätere Systemerweiterung einbezogen werden, bilden jedoch nicht den Fokus dieser Arbeit. Im Folgenden werden zuerst die verwendete Netztopologie und danach die Netzprotokolle beschrieben.

4.3.1 Netztopologie

Die in Kapitel 3 gewählten verteilten Filteralgorithmen können für zwei Netztopologien genutzt werden: Hierarchische Netze und generelle Punkt-zu-Punkt-Netze. Diese Netzstrukturen sind nur sehr selten die tatsächliche physische Verbindung in Netzen.

Vielmehr sind unzählige lokale Netze über Ethernet [13], ein Bussystem, miteinander verbunden. Token Ring [14], ein logisches Ringsystem, wird teilweise ebenfalls genutzt. Ziel eines Benachrichtigungssystems sollte die Unabhängigkeit von der physischen Netzstruktur sein. Deshalb wird ein logisches Overlaynetz auf der vorhandenen physischen Netzstruktur angelegt. Dieses kann auf beliebige Netzzugriffsverfahren abbilden. Durch diese Verlagerung in die Anwendungsschicht ist es möglich komplexe und applikationsrelevante Filteralgorithmen (wie für ein Benachrichtigungssystem notwendig) zu entwickeln.

Für das logische Overlaynetz wird eine zusammenhängende Graphstruktur gewählt, da diese eine allgemeinere Variante im Vergleich zu einer hierarchischen Struktur darstellt. Als weitere Eigenschaft des Overlaynetzes wird die verwendete Graphstruktur azyklisch umgesetzt, so dass die Struktur eines freien Baumes entsteht. Da es sich um ein logisches Netz handelt, ist diese Forderung der Kreisfreiheit umsetzbar. Die auf azyklischen Graphen implementierbaren Filteralgorithmen können einfacher und effizienter gestaltet werden, da Probleme wie die Verhinderung von Kreisen und doppelten Sendungen auf die tiefer im Protokollstapel vorhandenen Netzwerkschichten verlagert werden. Der Nachteil der Fehleranfälligkeit der einzig vorhandenen Verbindung zwischen zwei Netzknoten, also Vermittlungsknoten, ist im azyklischen Overlaynetz nicht vorhanden. Unterliegende Netzwerkschichten finden einen Weg zwischen zwei Knoten, sofern dieser existiert. Mehrere mögliche Wege zwischen zwei Knoten bei Einsatz eines zyklischen Overlaynetzes verhalten sich gleichartig bei Netzfehlern. Lediglich eine tatsächliche Netzpartitionierung führt bei heute eingesetzten Routingalgorithmen zu diesem Problem, und zwar sowohl bei azyklischen als auch zyklischen Overlaynetzen.

4.3.2 Netzwerkprotokoll

Die Kommunikation in einem logischen Overlaynetz muss auf einem Netzwerk- und Transport- bzw. Vermittlungsprotokoll aufsetzen. Um vorhandene Standards und Möglichkeiten zu nutzen, wird die Implementierung des Prototypen DAS auf der populären TCP/IP-Protokollfamilie [32, 33, 35] aufsetzen. Die Kommunikation zwischen Vermittlern untereinander sowie von Klienten und Vermittlern wird durch TCP/IP-Sockets realisiert. Diese Referenzimplementierung kann jedoch durch andere Kommunikationsprotokolle ersetzt werden, sofern die beiden folgenden Voraussetzungen eingehalten werden:

- Fehlerfreiheit der Übertragung
- Beibehaltung der Reihenfolge der übertragenen Daten (FIFO)

4.4 Systemarchitektur

Nachdem die Filterkomponente in einem Vermittler und die Eigenschaften des Vermittlungsnetzes erläutert wurden, kann nun die Architektur der verteilten Filterung beschrieben werden. Zur Umsetzung der verteilten Filteralgorithmen sind drei Teilsysteme nötig: Die Vermittler realisieren die verteilten Filteralgorithmen und bilden die Serverkomponente des Benachrichtigungssystems. Abschnitt 4.4.1 beschreibt ihre Architektur genauer. Dazu wird zuerst eine komponentenbasierte Sicht gezeigt, gefolgt

von einem stark vereinfachten UML-Diagramm. Auf Klientenseite müssen Abonnenten und Anbieter umgesetzt werden. Dabei sollen vorhandene Schnittstellen des zentralisierten Filteralgorithmus PrimAS genutzt werden. Abschnitt 4.4.2 geht auf die Realisierung der Abonnenten ein, Abschnitt 4.4.3 auf die Anbieter. Es wird wieder eine komponentenbasierte und eine UML-Sicht dargestellt. Da in der Implementierung englischsprachige Ausdrücke verwendet werden, werden auch im Folgenden z.T. englischsprachige Bezeichnungen genutzt.

4.4.1 Vermittler

Die Vermittlerkomponente, dargestellt in Abb. 4.3, besteht aus:

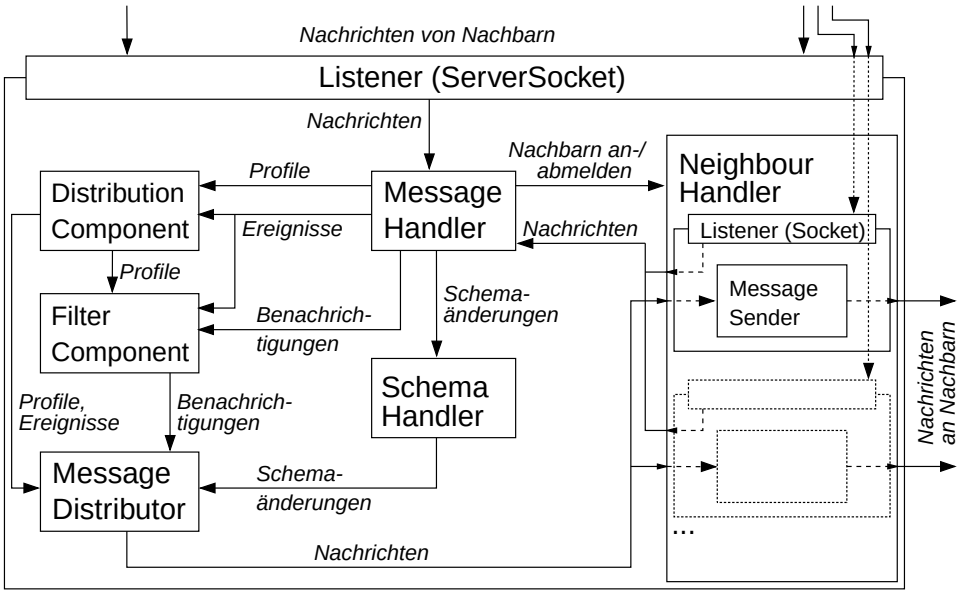


Abbildung 4.3: Architektur der Vermittlerkomponente.

Listener: Für die Entgegennahme von Verbindungsanfragen anderer Vermittler, Abonnenten und Anbieter ist in jedem Vermittler ein eigener Thread zuständig, der Listener. Realisiert mit Java ServerSockets werden die Nachrichten an den Message Handler zur Verarbeitung weitergeleitet.

Neighbour Handler: Die Nachbarn eines Vermittlers, Klienten und andere Vermittler, werden vom Neighbour Handler verwaltet. Durch diesen wird ein Zugriff auf die Nachbarn möglich.

Neighbour: Jeder Neighbour repräsentiert einen Nachbarn im Netz, entweder einen Vermittler oder einen Klienten. Ein Listener wartet auf Nachrichten vom Nachbarn; dazu wird ein eigener Thread genutzt in Verbindung mit Java Sockets. Treffen Nachrichten ein, werden diese an den Message Handler weitergeleitet und verarbeitet. Über einen Message Sender können Nachrichten an den Nachbarn verschickt werden.

Message Handler: Hier werden alle eintreffenden Nachrichten verarbeitet. Dazu gehören Verbindungsanfragen und -abmeldungen, Profilan- und abmeldungen, Ereignisse, Benachrichtigungen und Schemaänderungen.

Schema Handler: Er realisiert die Verwaltung der Ereignistypen und Wertebereiche. Schemaänderungen durch Nachbarn werden vom Message Handler an den Schema Handler weitergeleitet.

Distribution Component: In dieser Komponente ist die Verteilung von Profilen und Ereignissen im Netz realisiert. Je nach Algorithmus werden Profile und Ereignisse von den Nachbarn entgegengenommen und zur Filter Component hinzugefügt bzw. an weitere Vermittler verteilt.

Filter Component: Dieser Baustein realisiert die Filterung im Benachrichtigungssystem. Ereignisse, Benachrichtigungen und Profile der Nachbarn werden an ihn übergeben, gefiltert und u.U. passende Profile benachrichtigt. Grundlage dieser Komponente ist der erweiterte Filteralgorithmus PrimAS des zentralisierten Benachrichtigungssystems.

Message Distributor: Nachrichten werden durch ihn an die entsprechenden Nachbarn weitergeleitet.

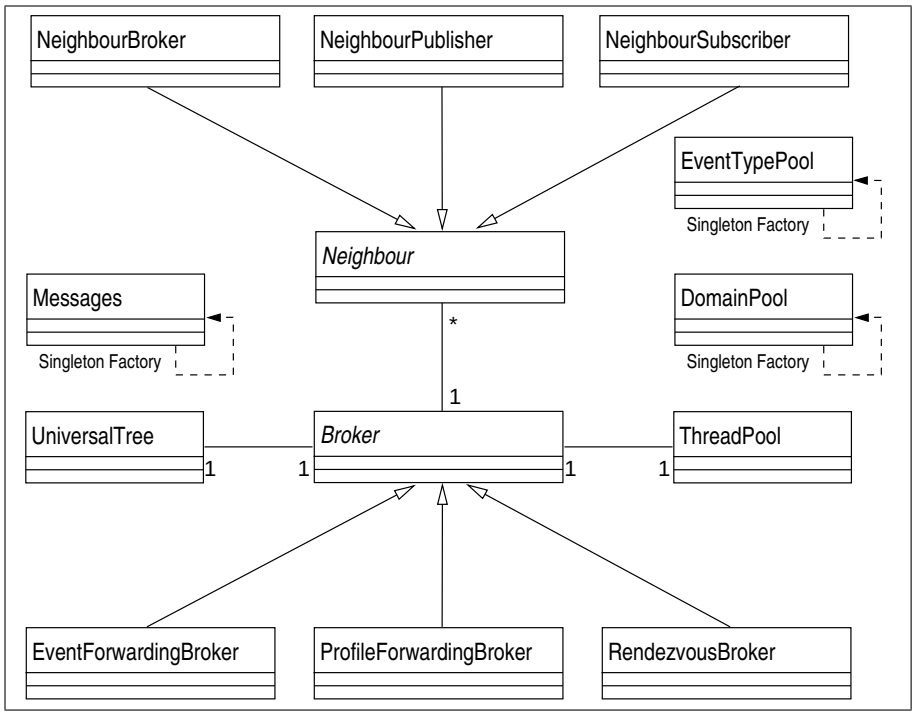


Abbildung 4.4: Klassendiagramm der Vermittlerkomponente.

Abb. 4.4 zeigt ein sehr vereinfachtes UML-Diagramm der Vermittlerkomponente. Im Mittelpunkt steht die abstrakte Klasse Broker (für einen Ausschnitt der Signatur siehe Anhang A.6), welche einen Vermittler implementiert. Die Komponenten Listener,

Neighbour Handler und Message Distributor werden von dieser Klasse realisiert. Für die verschiedenen Algorithmen existieren die Spezialisierungen EventForwardingBroker, ProfileForwardingBroker und RendezvousBroker. Diese kommunizieren untereinander jeweils mit anderen Filterprotokollen, die in Kapitel 5 genauer beschrieben werden. Jeder Broker ist mit einer abstrakten Klasse UniversalTree verbunden, welche die Filter Component implementiert (Ausschnitt der Signatur siehe Anhang A.3). Diese Klasse wird auch im zentralisierten Ansatz PrimAS zur Filterung genutzt und bietet unter anderem Methoden zum Anmelden von Profilen zur Filterung, zu deren Abmeldung und zum Filtern von Ereignissen und von entsprechenden Benachrichtigungen an.

Weiterhin ist jeder Broker mit einem ThreadPool verbunden, der aktive Threads verwaltet. Diese können zur Ausführung der Listener genutzt werden. Mehrere abstrakte Neighbour (Realisierung der Komponente Neighbour, Ausschnitt der Signatur siehe Anhang A.4) sind mit einem Broker verbunden. Diese repräsentieren einen Nachbarn im Vermittlungsnetz bzw. Klienten. Wichtigste Funktion ist das Empfangen, Versenden und Verarbeiten von Netznachrichten. Es existieren die Spezialisierungen NeighbourBroker (Ausschnitt der Signatur siehe Anhang A.5), NeighbourPublisher und NeighbourSubscriber. Diese fügen speziellen Code, beispielsweise zum Abmelden einer Verbindung, hinzu. Weiterhin existieren die Klassen EventTypePool und DomainPool als Realisierung der Komponente Schema Handler. Die Klasse Messages implementiert Message Handler und Distribution Component. Diese drei Klassen sind durch das Entwurfsmuster Singleton realisiert.

4.4.2 Abonnenten

Die Abonnentenkomponente ist in Abb. 4.5 dargestellt und besteht aus:

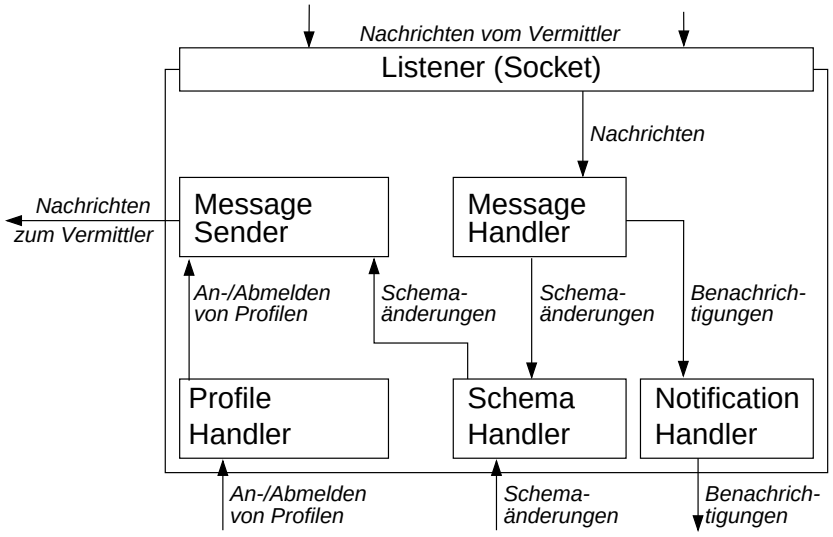


Abbildung 4.5: Architektur der Abonnentenkomponente.

Listener: Das Entgegennehmen von Nachrichten des lokalen Vermittlers geschieht im Listener, welcher in einem eigenen Thread verwaltet wird. Realisiert wird dieser

Listener über Java Sockets. Eintreffende Nachrichten werden an den Message Handler weitergeleitet.

Message Handler: Er ist für das Verarbeiten von Nachrichten zuständig. Solche Nachrichten sind beispielsweise die Ereignistypen und Wertebereiche des Systems sowie Benachrichtigungen. Treffen Benachrichtigungen ein, werden sie an den Notification Handler weitergeleitet, Schemainformationen werden an den Schema Handler geleitet.

Notification Handler: Diese Komponente leistet die eigentliche Benachrichtigung der Profile. Dazu wird die Benachrichtigung vom lokalen Vermittler den eigentlich vom Abonnenten spezifizierten Benachrichtigungen zugeordnet. Diese werden dann ausgeführt.

Message Sender: Die Übermittlung von Nachrichten an den lokalen Vermittler wird hier realisiert. Dazu werden Nachrichten im entsprechenden Format codiert und über ein Socket an den Vermittler gesendet.

Profile Handler: Dieser Baustein nimmt anzumeldende bzw. abzumeldende Profile entgegen und leitet diese zum Versand an den Message Sender. Analog zur zentralisierten Version werden Profile als Objekte entgegengenommen, daraus wird dann eine Darstellung zur Übertragung erzeugt.

Schema Handler: Die Verwaltung der Ereignistypen und Wertebereiche ist im Schema Handler realisiert. Änderungen durch den Vermittler werden vom Message Handler an diese Komponente weitergeleitet. Änderungen vom Anwender werden über den Message Sender ins System eingebracht.

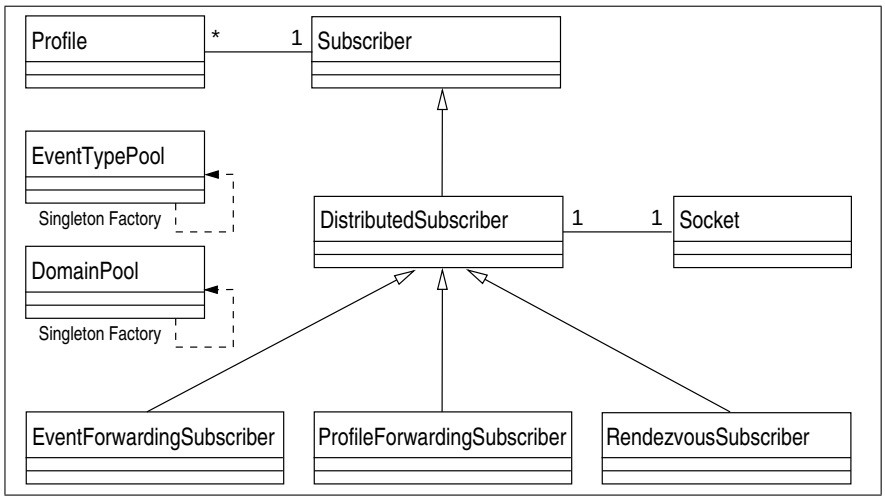


Abbildung 4.6: Klassendiagramm der Abonnentenkomponente.

Abb. 4.6 zeigt ein vereinfachtes UML-Diagramm der Abonnentenkomponente. Oberklasse der Implementierung des verteilten Abonnenten (DistributedSubscriber) ist die Klasse Subscriber aus der zentralisierten Version des Filteralgorithmus. Dadurch ist

eine transparente Nutzung der verteilten Version des Filteralgorithmus gewährleistet. Profile (Profile) sind mit genau einem Abonnenten verbunden, welchem beliebig viele Profile zugeordnet sind. Die verschiedenen Implementierungen des verteilten Filteralgorithmus sind durch die Spezialisierungen EventForwardingSubscriber, ProfileForwardingSubscriber und RendezvousSubscriber implementiert. Die Kommunikation erfolgt über ein Socket, das eine Verbindung mit dem lokalen Vermittler unterhält und die Komponente Listener realisiert. Die anderen Komponenten sind derzeit durch die Klasse DistributedSubscriber implementiert (Ausschnitt der Signatur siehe Anhang A.1). Der Schema Handler wird durch die Klassen EventTypePool und DomainPool implementiert.

4.4.3 Anbieter

Die Anbieterkomponente ist in Abb. 4.7 dargestellt und besteht aus:

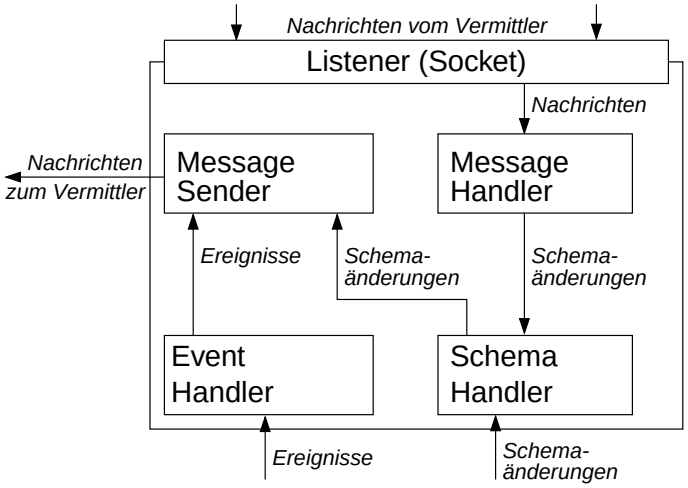


Abbildung 4.7: Architektur der Anbieterkomponente.

Listener: Das Entgegennehmen von Nachrichten des lokalen Vermittlers geschieht im Listener, der in einem eigenen Thread verwaltet wird. Realisiert wird dieser Listener über Java Sockets. Eintreffende Nachrichten werden an den Message Handler weitergeleitet.

Message Handler: Für das Verarbeiten von Nachrichten ist dieser Baustein zuständig. Solche Nachrichten sind beispielsweise die Ereignistypen und Wertebereiche des Systems.

Message Sender: Die Übermittlung von Nachrichten an den lokalen Vermittler wird hier realisiert. Dazu werden Nachrichten im entsprechenden Format codiert und über ein Socket an den Vermittler gesendet.

Event Handler: Er nimmt zu veröffentlichende Ereignisse entgegen und leitet diese zum Versand an den Message Sender. Analog zur zentralisierten Version werden

Ereignisse als Objekte entgegengenommen, daraus wird dann eine Darstellung zur Übertragung erzeugt.

Schema Handler: Die Verwaltung der Ereignistypen und Wertebereiche ist in dieser Komponente realisiert. Änderungen durch den Vermittler werden vom Message Handler an den Schema Handler weitergeleitet. Änderungen vom Anwender werden über den Message Sender ins System eingebracht.

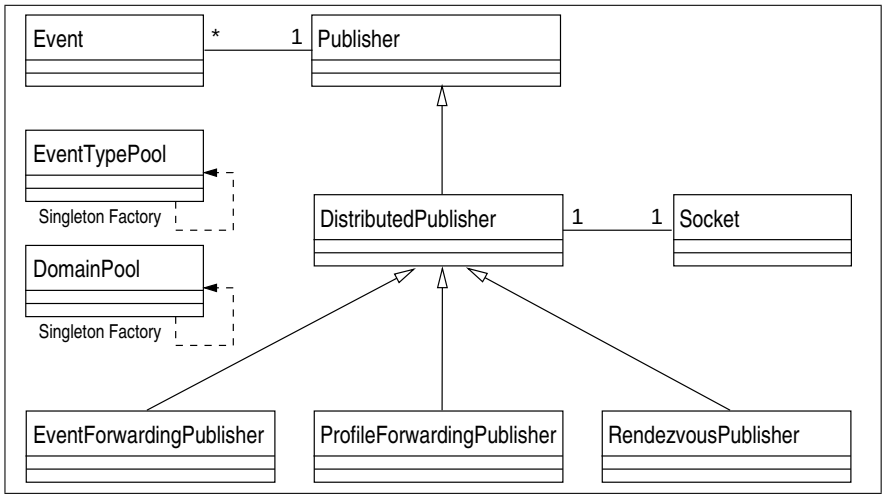


Abbildung 4.8: Klassendiagramm der Anbieterkomponente.

Abb. 4.8 zeigt ein vereinfachtes UML-Diagramm der Anbieterkomponente. Oberklasse der Implementierung des verteilten Anbieters (DistributedPublisher) ist die Klasse Publisher aus der zentralisierten Version des Filteralgorithmus. Dadurch ist eine transparente Nutzung der verteilten Version des Filteralgorithmus gewährleistet. Ereignisse (Event) sind genau einem Anbieter zugeordnet. Die verschiedenen Implementierungen der verteilten Filteralgorithmen sind durch die Spezialisierungen EventForwardingPublisher, ProfileForwardingPublisher und RendezvousPublisher implementiert. Die Kommunikation erfolgt über ein Socket, das eine Verbindung mit dem lokalen Vermittler unterhält und die Komponente Listener realisiert. Die anderen Komponenten sind derzeit durch die Klasse DistributedPublisher implementiert (Ausschnitt der Signatur siehe Anhang A.2). Der Schema Handler wird durch die Klassen EventTypePool und DomainPool implementiert.

4.4.4 Konfiguration

Die Konfiguration von DAS ist einfach und flexibel durch Dateien möglich. Es werden folgende Einstellungen unterstützt:

- Angabe des zu verwendenden verteilten Filteralgorithmus
- Angabe der IP-Adressen und der Ports der Vermittler
- Anlegen des virtuellen Vermittlungsnetzes durch Angabe der Nachbarn der Vermittler

- Angabe eines Zeitintervalls für Verbindungsversuche
- Angabe der Rendezvousknoten für die Ereignistypen

Zum Systemstart sind die gewünschten Vermittler durch Ausführen der Klasse Broker, mit einem Parameter zur Angabe der Nummer des Vermittlers, zu starten. Die Vermittler versuchen sich mit ihren Nachbarn zu verbinden. Ist das Zeitintervall für Verbindungsversuche überschritten, wird der Verbindungsversuch abgebrochen. Durch Veränderung dieses Parameters können Szenarien mit stark unterschiedlichen Startzeitpunkten der Vermittler gehandhabt werden. Ist das virtuelle Netz aufgebaut, ist das Benachrichtigungssystem mit dem eingestellten verteilten Filteralgorithmus nutzbar.

Zum Nutzen bzw. Testen des Systems ist auch eine einfache Konfiguration der Klienten durch Dateien möglich:

- Angabe des lokalen Vermittlers der Klienten
- Angabe der Dateien mit Profilen bzw. Ereignissen

Die Abonnenten werden durch Ausführen der Klasse DistributedSubscriber gestartet, verbinden sich mit dem lokalen Vermittler, melden Profile an und warten auf Benachrichtigungen. Die Anbieter werden durch Aufruf der Klasse DistributedPublisher gestartet. Der lokale Vermittler wird kontaktiert, die Ereignisse werden gesendet und die Ausführung terminiert.

4.5 Zusammenfassung

In diesem Kapitel wurden Systementwurf und -architektur von DAS dargestellt. In Abschnitt 4.1 wurden der zugrunde liegende zentralisierte Filteralgorithmus PrimAS [3] und dessen Erweiterungen und Anpassungen zur Unterstützung der verteilten Filterung beschrieben. Diese enthalten einen Ansatz zur Verminderung des Speicherbedarfs und die Erweiterung der bisherigen Filterstruktur zu einem dynamischen Filteralgorithmus zur Nutzung in den Vermittlern. Abschnitt 4.2 erläuterte die genutzte Profildefinitionssprache sowie die unterstützten Wertebereiche für Attribute und die möglichen Operatoren der Prädikate (Gleichheit, Größer, Kleiner, Mengentest und Bereichstest). Netztopologie und Transportprotokoll wurden in Abschnitt 4.3 betrachtet. Es wird ein azyklischer zusammenhängender Graph in Verbindung mit TCP/IP genutzt.

Abschließend wurde in Abschnitt 4.4 die Systemarchitektur der Vermittler, Abonnenten und Anbieter schematisch und mit Hilfe von UML-Diagrammen beschrieben. Konfiguration und Systemstart wurden ebenfalls dargestellt. Nach dieser allgemeinen Systemübersicht werden im folgenden Kapitel die Umsetzungen der Optimierungen zur Verminderung von Profilredundanzen und die implementierten Filteralgorithmen und -protokolle dargestellt.

Kapitel 5

Umsetzung der Filteralgorithmen und -protokolle

Inhalt

Nachdem im letzten Kapitel die allgemeine Architektur von DAS vorgestellt wurde, werden in diesem Kapitel die Umsetzungen der Optimierungen zur Verminderung von Profilredundanzen und die drei eingesetzten verteilten Filteralgorithmen beschrieben. In der Literatur [5, 23, 24, 25, 26, 27] werden zwar Grundideen sowohl der Optimierungen als auch der Protokolle der Filteralgorithmen beschrieben, eine exakte und genaue Beschreibung ist jedoch nicht zu finden. Um diese Lücke zu schließen werden in den folgenden Abschnitten mögliche Umsetzungen der Optimierungen beschrieben. Vorher werden das Bedecken und das Verschmelzen einer Aufwandsanalyse unterzogen, um eine dieser Optimierungen für die praktische Umsetzung auszuwählen. Danach werden die Filterprotokolle der Algorithmen Ereignisweiterleitung, Profilweiterleitung und Rendezvousknoten beschrieben.

5.1 Verminderung von Profilredundanzen

Als Optimierungen zur Verminderung von Profilredundanzen sind generell Verschmelzen und Bedecken (als Erweiterung der Äquivalenz) möglich. In Kapitel 3 wurde be-

reits aus Gründen der besseren Speicherausnutzung die optimistische Speicherung als Speicherstrategie gewählt. Im Folgenden wird nun versucht eine Aufwandsabschätzung beim Nutzen des Verschmelzens bzw. Bedeckens durchzuführen. Darauf aufbauend wird dann nur eines der beiden Verfahren in DAS umgesetzt. Im weiteren Verlauf dieses Abschnitts werden Implementierungsvarianten zur Umsetzung der Optimierungen entworfen und dargestellt.

5.1.1 Aufwandsabschätzung

Die Aufwandabschätzung soll bestimmen, welches der beiden Optimierungsverfahren mehr Nachrichtenverkehr erzeugt. Bei Hinzufügen von Profilen ist beim Bedecken und Verschmelzen mit ungefähr dem gleichen Aufwand zu rechnen: Bei Profilweiterleitung und Rendezvousknoten wird beim Verschmelzen teilweise anstelle von mehreren Profilen nur ein verschmolzenes an die Nachbarn weitergeleitet. Beim Bedecken werden nicht alle Profile verbreitet, da bedeckte Profile nicht weitergeleitet werden. Unterschiede ergeben sich jedoch beim Abmelden von Profilen, wie die beiden folgenden Abschnitte zeigen.

Abmelden beim Verschmelzen

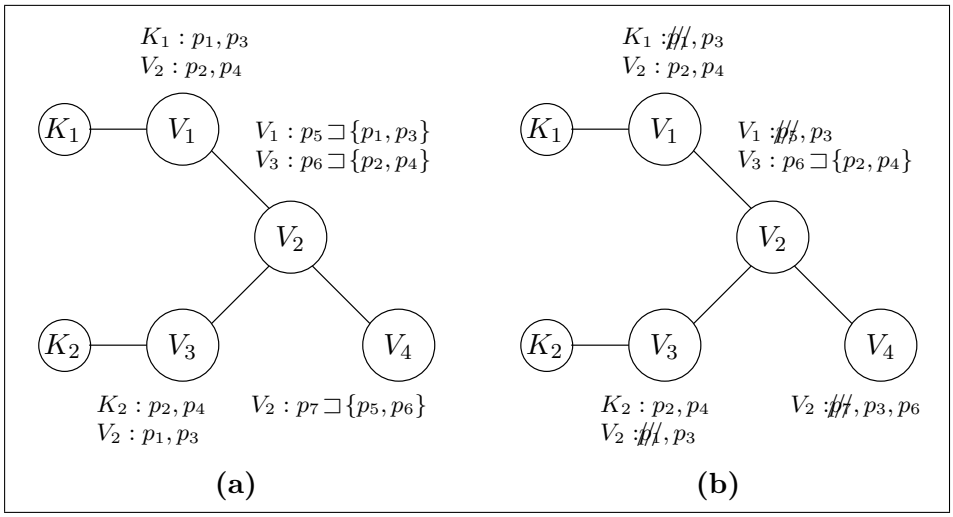


Abbildung 5.1: Konfiguration beim Verschmelzen.

Beim Verschmelzen in Verbindung mit optimistischer Speicherung ist mit erheblichem Netzverkehr zu rechnen. Grund ist, dass Vermittler beliebig Verschmelzungen durchführen. Die lokalen Vermittler der Abonnenten der verschmolzenen Profile kennen diese Verschmelzungen jedoch nicht. Beim Abmelden von Profilen müssen Vermittler deshalb die Ausgangsprofile vor der Verschmelzung anfordern. Diese können u.U. erneut Resultat einer Verschmelzung sein. Dadurch ergeben sich teilweise große Netzlasten, wie folgendes Beispiel zeigt:

Abb. 5.1 zeigt den Ausschnitt einer Konfiguration mit vier Vermittlungsservern V_1 bis V_4 beim Einsatz der Profilweiterleitung. Bei den Vermittlern sind jeweils die

zu filternden Profile eingetragen. Nach Anmelden der Profile p_1 bis p_4 der zwei Abonnenten K_1 und K_2 ergibt sich Abb. 5.1(a). Vermittler V_2 verschmilzt die Profile p_1 bis p_4 zu den Profilen $p_5 \sqsupset \{p_1, p_3\}$ und $p_6 \sqsupset \{p_2, p_4\}$. Vermittler V_4 verschmilzt erneut: $p_7 \sqsupset \{p_5, p_6\}$. Jetzt möchte K_1 Profil p_1 abmelden (Abb. 5.1(b)). Dazu sendet er diese Abmeldung an V_1. V_1 löscht p_1 aus seiner Filterstruktur und sendet die Abmeldung an V_2. V_2 bemerkt, dass $p_5 \sqsupset \{p_1, p_3\}$. Allerdings ist Profil p_3 nicht bekannt. Deshalb muss dieses Profil von V_1 angefordert werden. Erst dann kann p_5 gelöscht werden, wozu p_3 eingetragen werden muss. Daraufhin muss V_4 erfahren, dass p_5 verändert wurde und das an p_7 beteiligte Profil p_6 von V_2 anfordern. Profil p_5 existiert nicht mehr, also werden p_3 und p_6 von V_2 versendet und in V_4 gespeichert. Dann erst kann p_7 entfernt werden. Schließlich kann die Abmeldung von p_1 an V_3 und V_4 weitergeleitet werden. V_3 kann p_1 löschen, V_4 enthält kein p_1 mehr und beendet die Arbeit.

Abmelden beim Bedecken

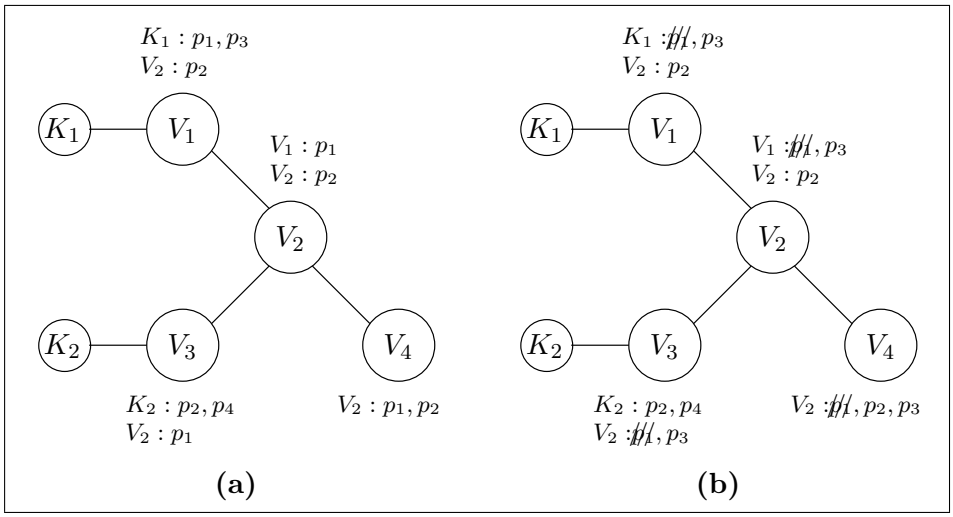


Abbildung 5.2: Konfiguration beim Bedecken.

Beim Bedecken ergibt sich weniger Netzverkehr, da Vermittlern bekannt ist, welche Bedeckungen existieren. Deshalb genügt beim Abmelden eines bedeckenden Profils die Weiterleitung der Abmeldung und das Versenden der bedeckten Profile. Folgendes Beispiel verdeutlicht diesen Umstand:

Abb. 5.2(a) zeigt beim Einsatz von Bedeckungen die gleiche Konfiguration wie beim Verschmelzen. Die direkten Bedeckungseigenschaften sind: $p_1 \sqsupset p_3$ und $p_2 \sqsupset p_4$. Die Abmeldung von p_1 (Abb. 5.2(b)) wird an Vermittler V_1 gesendet. Dieser löscht p_1 und sendet das ehemals bedeckte Profil p_3 und die Abmeldung von p_1 an V_2 weiter. V_2 fügt p_3 zur Filterstruktur hinzu, löscht p_1 und sendet es, in Verbindung mit p_3, an V_3 und V_4. Diese fügen ebenfalls p_3 zur ihrer Filterstruktur hinzu und löschen p_1.

Zusammenfassung

Beim Nutzen des Verschmelzens ist erheblich mehr Netzverkehr notwendig als beim Bedecken, da Ausgangsprofile der Verschmelzungen explizit angefordert werden müssen. Die Definitionen der Ausgangsprofile werden zwar nicht gespeichert, jedoch müssen deren Herkunft (von welchem Vermittler wurden sie gesendet) und ein Identifikator gespeichert werden. Somit ist ebenfalls erhöhter Speicherverbrauch gegeben. Aufgrund dieser beiden erheblichen Nachteile beim Verschmelzen wird in DAS das Bedecken angewendet.

Im folgenden Abschnitt werden Berechnungsmöglichkeiten für die Bedeckungen entworfen. Danach wird kurz auf mögliche Berechnungen der Verschmelzungen eingegangen. Dabei existieren jeweils zwei Implementierungstechniken: profilbasierte und bereichsbasierte Berechnung.

5.1.2 Berechnung des Bedeckens

Die profilbasierte Berechnung des Bedeckens vergleicht Profildefinitionen direkt miteinander. Der bereichsbasierte Ansatz testet die Ausnutzung der Wertebereiche durch die Profile, aufbauend auf der Filterstruktur in den Vermittlern (siehe Abschnitt 4.1.1).

Profilbasierte Berechnung

Durch die Beschränkung in den Vergleichsoperatoren ist es möglich, die Bestimmung des Bedeckens abhängig von diesen Operatoren durchzuführen. Dazu wird eine Fallunterscheidung für alle Kombinationen der Vergleichsoperatoren durchgeführt. Tabelle 5.1 enthält diese Fallunterscheidung. Dabei wird $pr_1 \sqsupseteq pr_2$ betrachtet. Die erste Spalte enthält den Vergleichsoperator op_1 des Prädikates pr_1, die zweite Spalte den Vergleichsoperator op_2 von pr_2. Das Attribut von pr_1 und pr_2 ist a_1. Spalte 3 beschreibt die Bedingung, die gelten muss, damit $pr_1 \sqsupseteq pr_2$ gilt. Die Bezeichner $x \in \mathcal{W}(a_1)$ und $y \in \mathcal{W}(a_1)$ beschreiben die Operanden der Prädikate pr_1 und pr_2. Ausnahmen sind Bereichstests ($\dashv$) und Mengentests ($\in$). Dabei sind die Operanden zwei Elemente, $x_{min}, y_{min} \in \mathcal{W}(a_1)$ und $x_{max}, y_{max} \in \mathcal{W}(a_1)$, bzw. Mengen $X, Y \subseteq \mathcal{W}(a_1)$.

Bereichsbasierte Berechnung

Die bereichsbasierte Berechnung setzt auf der vorhandenen Filterstruktur auf. Dazu werden die Profilverweise (bzw. die Bitlisten) in den Blättern der Filterstruktur der Vermittler analysiert. Abhängig davon welche Verweise ein Profil beinhalten oder nicht, kann damit eine Aussage über die Bedeckung getroffen werden. Zusätzlich muss dazu für jedes Profil die Anzahl der Gesamtverweise in den Blättern gespeichert werden (oder bei jeder Berechnung diese Zahl ermittelt werden). Die Berechnung der von einem Prädikat bedeckten Prädikate (falls dies für alle Prädikate zweier Profile zutrifft, bedecken sich die Profile), abhängig vom genutzten Vergleichsoperator, geschieht wie folgt. $Arr(w)$ bezeichnet dabei den Index des Elementes w im in Abschnitt 4.1 vorgestellten Feld mit Kantenbeschriftungen:

Gleichheit (=)**:** Sei das Prädikat $pr = (a, =, w)$ und $i = Arr(w)$ bzw. des nächstgrößeren Elements. Falls w in Arr enthalten ist ($i \neq -1$) und dessen Intervall

op_1	op_2	Bedingung
$=$	$=$	$x = y$
$=$	$>$	$x = \text{succ}(y) = \max(\mathcal{W}(a_1))$
$=$	$<$	$x = \text{pred}(y) = \min(\mathcal{W}(a_1))$
$=$	$\dashv$	$x = y_{min} = y_{max}$
$=$	$\in$	$x \in Y, \lvert Y\rvert = 1$
$>$	$=$	$x < y$
$>$	$>$	$x \leq y$
$>$	$<$	$x = \text{pred}(\min(\mathcal{W}(a_1)))$
$>$	$\dashv$	$x < y_{min}$
$>$	$\in$	$x < \min(Y)$
$<$	$=$	$x > y$
$<$	$>$	$x = \text{succ}(\max(\mathcal{W}(a_1)))$
$<$	$<$	$x \geq y$
$<$	$\dashv$	$x > y_{max}$
$<$	$\in$	$x > \max(Y)$
$\dashv$	$=$	$x_{min} \leq y \leq x_{max}$
$\dashv$	$>$	$x_{max} = \max(\mathcal{W}(a_1)), x_{min} \leq \text{succ}(y)$
$\dashv$	$<$	$x_{min} = \min(\mathcal{W}(a_1)), x_{max} \geq \text{pred}(y)$
$\dashv$	$\dashv$	$x_{min} \leq y_{min}, x_{max} \geq y_{max}$
$\dashv$	$\in$	$x_{min} \leq \min(Y), x_{max} \geq \max(Y)$
$\in$	$=$	$y \in X$
$\in$	$>$	$\forall z \in [\text{succ}(y), \max(\mathcal{W}(a_1))](z \in X)$
$\in$	$<$	$\forall z \in [\min(\mathcal{W}(a_1)), \text{pred}(y)](z \in X)$
$\in$	$\dashv$	$\forall z \in [y_{min}, y_{max}](z \in X)$
$\in$	$\in$	$\forall z \in Y(z \in X)$

Tabelle 5.1: Fälle bei profilbasierter Berechnung von Bedeckungen.

Arr_i genau ein Element beschreibt, werden alle Profile aus $\widetilde{Arr}_i$, die in keinem $\widetilde{Arr}_j$ mit $j \neq i$ enthalten sind, bedeckt. Anderenfalls werden keine Profile bedeckt.

Größer ($>$)**:** Sei das Prädikat $pr = (a, >, w)$ und $k = Arr(\text{succ}(w))$ bzw. des nächstgrößeren Elements. Falls $\text{succ}(w) = \min(\mathcal{W}(a))$, werden alle Profile bedeckt. Sonst werden alle Profile bedeckt, deren Zahl der Vorkommen in $\widetilde{Arr}_j$ mit $k \leq j < |\widetilde{Arr}|$ gleich der Zahl der Gesamtvorkommen in $\widetilde{Arr}$ ist.

Kleiner ($<$)**:** Sei das Prädikat $pr = (a, <, w)$ und $l = Arr(\text{pred}(w))$ bzw. des nächstkleineren Elements. Falls $\text{pred}(w) = \max(\mathcal{W}(a))$, werden alle Profile bedeckt. Sonst werden alle Profile bedeckt, deren Zahl der Vorkommen in $\widetilde{Arr}_j$ mit $0 \leq j \leq l$ gleich der Zahl der Gesamtvorkommen in $\widetilde{Arr}$ ist.

Zwischen ($\dashv$)**:** Sei das Prädikat $pr = (a, \dashv, w)$. Falls $w_{min} = \min(\mathcal{W}(a))$ und $w_{max} = \max(\mathcal{W}(a))$, werden alle Profile bedeckt. Sonst sei $m = Arr(w_{min})$,

falls w_{min} in Arr enthalten ist, sonst die Feldposition des nächstgrößeren Elements bzw. $m = 0$, falls $w_{min} = \min(\mathcal{W}(a))$. Ebenfalls sei $n = Arr(w_{max})$, falls w_{max} in Arr enthalten ist bzw. die Feldposition der nächstkleineren Elementes. Es werden alle Profile bedeckt, deren Zahl der Vorkommen in $\widetilde{Arr}_j$ mit $m \leq j \leq n$ gleich der Zahl der Gesamtvorkommen in $\widetilde{Arr}$ ist.

Enthalten ($\in$)**:** Sei das Prädikat $pr = (a, \in, W)$. Für alle Elemente $m \in W$ muss die Bedingung des Falles Gleichheit zutreffen.

5.1.3 Berechnung des Verschmelzens

Auch beim Verschmelzen existieren ein profilbasierter und ein bereichsbasierter Ansatz. Aufgrund der Auswahl des Bedeckens für die Implementierung wird von einer detaillierten Beschreibung abgesehen.

Profilbasierte Berechnung

op_1	op_2	op_3	Berechnung
$=$	$=$	$=$	falls $x = y : z = x$
$=$	$=$	$>$	$z = \text{pred}(\min(x, y))$
$=$	$=$	$<$	$z = \text{succ}(\max(x, y))$
$=$	$=$	$\in$	$Z = \{x\} \cup \{y\}$
$=$	$=$	$\dashv$	$z_{min} = \min(x, y), z_{max} = \max(x, y)$
$=$	$>$	$=$	falls $x = \max(\mathcal{W}(a_1)) \wedge y = \text{pred}(\max(\mathcal{W}(a_1))) : z = x$

Tabelle 5.2: Ausschnitt der Fälle bei einer profilbasierten Berechnung des Verschmelzens.

Bei der profilbasierten Berechnung werden erneut die Operatoren für eine Fallunterscheidung genutzt. Je nach genutzten Vergleichsoperatoren der beiden zu verschmelzenden Profile und des daraus erzeugten Profils ergeben sich zahlreiche Unterschiede. Tabelle 5.2 zeigt lediglich einen Ausschnitt der möglichen Fälle beim Verschmelzen von pr_1 und pr_2 zu pr_3. Das Attribut der Prädikate ist dabei a_1. Die erste Spalte enthält den Vergleichsoperator des Prädikates pr_1, die zweite Spalte den Operator von pr_2. Spalte 3 zeigt den Ergebnisoperator von pr_3, Spalte 4 beschreibt die Berechnung des Operanden z (bzw. $z_{min}, z_{max} \in \mathcal{W}(a_1)$ bei Mengentests und $Z \subseteq \mathcal{W}(a_1)$ bei Bereichstests) des Prädikates pr_3. Die Bezeichner x und y beschreiben die Operanden von pr_1 bzw. pr_2. Nur in seltenen Fällen ist perfektes Verschmelzen möglich, einige Ergebnisoperatoren sind überhaupt nicht möglich. Nur bei Nutzung des Mengenoperators im Ergebnisprädikat ist eine perfekte Verschmelzung stets erzeugbar.

Bereichsbasierte Berechnung

Die bereichsbasierte Berechnung setzt auf der Filterstruktur auf. Sollen zwei Profile miteinander verschmolzen werden, so werden ihre Vorkommen in den Profilverweisen der Blätter der Filterstruktur analysiert. Dann werden diese Profile aus der Struktur entfernt und ein neues Profil eingetragen, dessen Verweise dort vorkommen,

wo eines der beiden Ausgangsprofile oder beide einen Verweis hatten. Dadurch entsteht ausschließlich perfektes Verschmelzen. Allerdings ist es nur noch unter speziellen Umständen möglich die Beschreibung des Profils sinnvoll in der Profildefinitionssprache auszudrücken. Meistens ist allein die Spezifikation mit dem Mengenoperator möglich, was sehr große Profildefinitionen zur Folge hat.

5.1.4 Zusammenfassung

Aufgrund der in Abschnitt 5.1.1 vorgenommenen Aufwandsanalyse wird das Bedecken für die Implementierung in DAS gewählt. Für das Berechnen des Bedeckens und Verschmelzens wurden jeweils zwei Ansätze, profilbasiert und bereichsbasiert, erarbeitet (Abschnitte 5.1.2 und 5.1.3). Beide werden in DAS umgesetzt. Erste Experimente zum Vergleich der Effizienz beider Verfahren ergeben eine starke Abhängigkeit der Effizienz von Anzahl und Art der Profile. Eine genaue Analyse dieser Verfahren steht jedoch noch aus.

Im nächsten Abschnitt werden die Filterprotokolle der Algorithmen Ereignisweiterleitung, Profilweiterleitung und Rendezvousknoten im System DAS beschrieben.

5.2 Filterprotokolle

Nachdem bisher die generelle Architektur von DAS und das Berechnen von Bedeckungen und Verschmelzungen erläutert wurden, werden jetzt die Filterprotokolle der drei implementierten verteilten Filteralgorithmen dargestellt. Die im Folgenden beschriebenen Schritte laufen in der Implementierung in einer einzigen, atomaren Aktion in den Vermittlern ab. In den Beispielen gilt $e_y \succ p_x$, wenn $x = y$ und eine Benachrichtigung b_x benachrichtigt Profil p_x über Ereignis e_x.

5.2.1 Ereignisweiterleitung

Das Protokoll der Ereignisweiterleitung gestaltet sich recht einfach - die Ereignisse werden im Netz verteilt, Vermittler filtern nur Profile ihrer lokalen Abonnenten.

Profilbehandlung

Anmelden eines Profils p_x an einem Vermittler:

$\mathcal{P} = \mathcal{P} \cup \{p_x\}$ (füge Profil p_x zur Filterstruktur hinzu).

Abmelden eines Profils p_x von einem Vermittler:

$\mathcal{P} = \mathcal{P} \setminus \{p_x\}$ (entferne Profil p_x aus der Filterstruktur).

Ereignisbehandlung

Veröffentlichen eines Ereignisses e_x an einen Vermittler V_x durch einen Nachbar V_y:

1. Sende Ereignis e_x zu allen Nachbarn $\mathcal{N}(V_x) \setminus \{V_y\}$ (Nachbarvermittler außer V_y).
2. Filtere Ereignis e_x und benachrichtige die Abonnenten passender Profile.

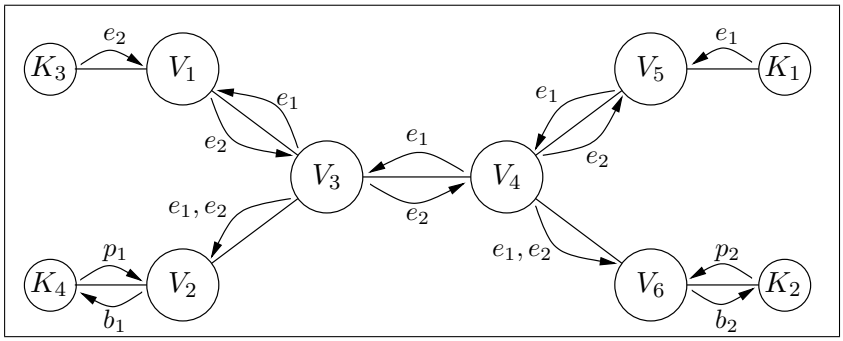


Abbildung 5.3: Konfiguration und Nachrichten bei der Ereignisweiterleitung.

Beispiele

Abb. 5.3 zeigt eine Beispielkonfiguration der Ereignisweiterleitung mit sechs Vermittlern V_1 bis V_6, zwei Abonnenten K_2 und K_4 sowie zwei Anbietern K_1 und K_3. Die Profile p_1 und p_2 werden an die lokalen Vermittler V_2 bzw. V_6 gesendet. Beim Abmelden ist die Kommunikation auch nur auf die lokalen Vermittler beschränkt. Anbieter K_1 sendet Ereignisse e_1 an seinen lokalen Vermittler V_5. Anbieter K_3 sendet Ereignisse e_2 an seinen lokalen Vermittler V_1. Diese Ereignisse werden nun im gesamten Vermittlungsnetz verbreitet. So sendet z.B. Vermittler V_4 Ereignis e_1, das von V_5 empfangen wurde, an $\mathcal{N}(V_4) \setminus \{V_5\} = \{V_3, V_5, V_6\} \setminus \{V_5\} = \{V_3, V_6\}$. Schließlich benachrichtigen die lokalen Vermittler V_2 und V_6 ihre Abonnenten K_4 und K_2 mit den Benachrichtigungen b_1 bzw. b_2.

5.2.2 Profilweiterleitung

Das Protokoll der Profilweiterleitung nutzt Bedeckungen zur Verminderung von Profilredundanzen und gestaltet sich wie folgt:

Profilbehandlung

Anmelden eines Profils p_x an einem Vermittler V_x durch Nachbar V_y:

1. Falls $V_y \in \mathcal{N}(V_x)$ (V_y ist ein Vermittler):

   $\mathcal{P} = \mathcal{P} \setminus \{p_y \in \mathcal{P}_{\sqsupseteq}(p_x) | \mathcal{A}(p_y) = V_y\}$ (entferne alle von Profil p_x bedeckten bzw. zu p_x äquivalenten Profile mit V_y als Abonnenten aus der Filterstruktur).

2. Falls $\{p_y \in \mathcal{P}_{\sqsubseteq}(p_x) | \mathcal{A}(p_y) = V_y\} = \emptyset$ (keine p_x bedeckenden oder äquivalenten Profile von Nachbar V_y existieren):

   (a) Falls $\mathcal{P}_{\sqsubseteq}(p_x) = \emptyset$ (überhaupt keine p_x bedeckenden oder äquivalenten Profile existieren):

   Melde Profil p_x bei allen Nachbarn $\mathcal{N}(V_x) \setminus \{V_y\}$ (Nachbarvermittler außer V_y) als eigenes Profil an.

(b) Sonst sei $N = \{\mathcal{A}(p_y) \in \mathcal{N}(V_x) | \mathcal{A}(p_y) \neq V_y, p_y \in \mathcal{P}_{\sqsubseteq}(p_x)\}$. Falls $|N| = 1$ (alle p_x bedeckenden oder äquivalenten Profile wurden vom gleichen Nachbarvermittler angemeldet):

Sende p_x als eigenes Profil an diesen Vermittler $V \in N$.

3. $\mathcal{P} = \mathcal{P} \cup \{p_x\}$ (füge Profil p_x zur Filterstruktur hinzu).

Abmelden eines Profils p_x von einem Vermittler V_x durch Nachbar V_y:

1. Sei $N = \{\mathcal{A}(p_y) \in \mathcal{N}(V_x) | p_y \in \mathcal{P}_{\sqsubseteq}(p_x)\}$.
   Falls $|N| = 1 \vee \mathcal{P}_{\sqsubseteq}(p_x) = \emptyset$ (alle p_x bedeckenden Profile sind vom gleichen Nachbarvermittler V oder keine p_x bedeckenden Profile existieren):
   (a) Sende allen Nachbarvermittlern $V_z \in \mathcal{N}(V_x) \setminus \{V_y\}$ ein Profil aus $S = \{p_y \in \mathcal{P}_{\equiv}(p_x) | \mathcal{A}(p_y) \neq V_z\}$, falls $S \neq \emptyset$, sonst die Profilmenge $\{p_y \in \widetilde{\mathcal{P}}_{\sqsupset}(p_x) | \mathcal{A}(p_y) \neq V_z\}$ (allen Nachbarvermittlern V_z, außer V_y, ein zu p_x äquivalentes Profil (falls ein solches existiert) bzw. die von p_x direkt bedeckten Profile, die nicht von V_z angemeldet wurden).
   (b) Sende allen Nachbarn $\mathcal{N}(V_x) \setminus \{V_y\}$ (Nachbarvermittler außer V_y) die Abmeldung von p_x.
2. $\mathcal{P} = \mathcal{P} \setminus \{p_x\}$ (entferne Profil p_x aus der Filterstruktur).

Ereignisbehandlung

Veröffentlichen eines Ereignisses e_x an Vermittler V_x durch einen Nachbar:

Filtere e_x und benachrichtige die Nachbarn $V_y \in \{\mathcal{A}(p_x) | p_x \in \mathcal{P}, e_x \succ p_x\}$ (Abonnenten zu e_x passender Profile) wie folgt:

Falls $V_y \in \mathcal{N}(V_x)$ (Nachbar V_y ist Vermittler), benachrichtige V_y nur genau einmal über Ereignis e_x, sonst benachrichtige V_y über jedes zu Ereignis e_x passende Profil.

Benachrichtigungsbehandlung

Veröffentlichen einer Benachrichtigung b_x an einem Vermittler V_x durch einen Nachbarn V_y:

Filtere das in b_x enthaltene Ereignis e_x und benachrichtige die Nachbarn $V_z \in \{\mathcal{A}(p_x) | \mathcal{A}(p_x) \neq V_y, p_x \in \mathcal{P}, e_x \succ p_x\}$ (Nachbarn V_z passender Profile außer V_y) wie folgt:

Falls $V_z \in \mathcal{N}(V_x)$ (Nachbar V_z ist Vermittler), benachrichtige diesen nur genau einmal über Ereignis e_x, sonst benachrichtige V_z über jedes zu Ereignis e_x passende Profil.

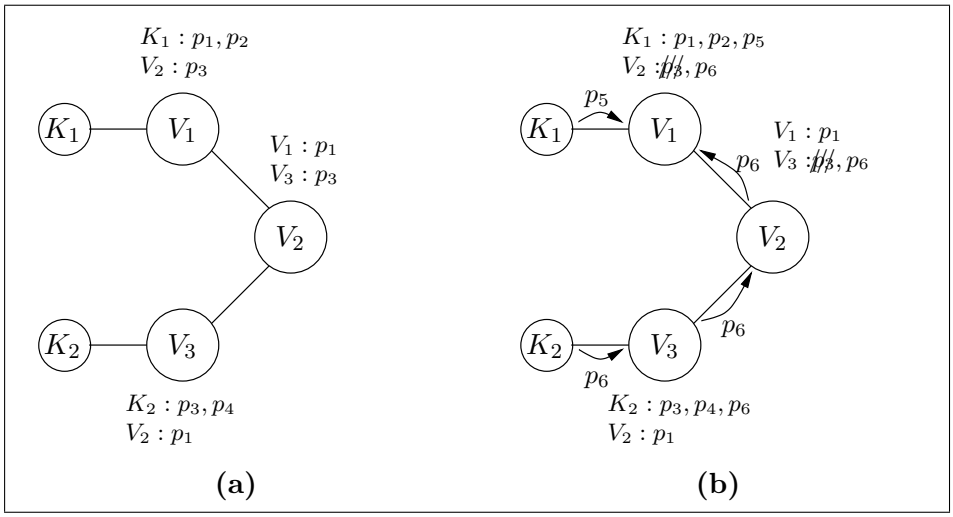


Abbildung 5.4: Nachrichten und Filterstruktur bei der Profilweiterleitung.

Beispiele

Abb. 5.4 zeigt eine Beispielkonfiguration der Profilweiterleitung mit drei Vermittlern V_1 bis V_3 und zwei Abonnenten K_1 und K_2. Die Profile p_1 bis p_6 besitzen folgende Bedeckungseigenschaften: $p_1 \sqsupseteq p_2 \sqsupseteq p_5$ und $p_1 \sqsupseteq p_6 \sqsupseteq p_3 \sqsupseteq p_4$. Zwischen anderen Profilen besteht Unabhängigkeit.

In Abb. 5.4(a) ist die Ausgangskonfiguration mit den Profilen p_1 und p_2 vom Abonnenten K_1 und den Profilen p_3 und p_4 vom Abonnenten K_2 dargestellt. Bei den Vermittlern sind jeweils die zu filternden Profile bei der Profilweiterleitung eingetragen. So filtert Vermittler V_2 unter anderem Profil p_1 für Vermittler V_1 und Profil p_3 für Vermittler V_3.

Abb. 5.4(b) zeigt die Nachrichten beim Anmelden der Profile p_5 und p_6, sowie die Änderungen in den zu filternden Profilmengen der Vermittler. Profil p_5 wird an V_1 ausgeliefert. Es existieren bereits bedeckende Profile p_1 und p_2 von K_1 (2.), somit wird p_5 lediglich in die Filterstruktur eingefügt (3.). Schwieriger ist die Situation bei Profil p_6, welches an V_3 ausgeliefert wird. In V_3 existieren keine bedeckenden oder äquivalenten Profile von K_2 (2.), sondern nur von V_2 (2(b)). Also wird p_6 and V_2 weitergeleitet. Danach wird Profil p_6 in die Filterstruktur von V_3 eingefügt (3.). Erreicht p_6 Vermittler V_2 von Vermittler V_3 (1.), werden alle bedecken und äquivalenten Profile von V_3 gelöscht (1.), also p_3. Es existieren keine bedeckenden bzw. äquivalenten Profile von V_3 (2.). Da nur ein bedeckendes Profil, p_1, existiert, wird p_6 an dessen Abonnenten V_1 gesendet (2(b)). Danach wird p_6 in die Filterstruktur von V_2 eingefügt (3.). Erreicht nun Profil p_6 Vermittler V_1, wird p_3 entfernt (1.). Das einzige bedeckende Profil p_1 ist nun von einem Klienten, also greift Fall 2(b) nicht. Schließlich wird p_6 in die Filterstruktur eingefügt (3.).

Werden jetzt die Profile p_5 und p_6 abgemeldet, erhält man wieder die Situation gemäß Abb. 5.4(a): Vermittler V_1 hat zwar bedeckende Profile, jedoch von einem Klienten, somit greift 1. nicht. Also wird p_5 aus der Filterstruktur von V_1 entfernt (2.). Profil p_6 erreicht V_3 zur Abmeldung, das bedeckende Profil p_1 ist von V_2 (1.).

Nun wird V_2 Profil p_3 gesendet (1(a)). Danach wird p_6 bei V_2 abgemeldet (1(b)) und schließlich aus der Filterstruktur entfernt (2.). Erreicht p_3 Vermittler V_2, wird es zur Filterstruktur hinzugefügt (Anmelden 3.). Die Abmeldung von Profil p_6 in Vermittler V_2 führt zum Senden von p_3 an V_1 (1(a)), dem Senden der Abmeldung von p_6 an V_1 (1(b)) und dem Entfernen von p_6 aus der Filterstruktur von V_2 (2.). In V_1 geschehen die Vorgänge aus V_2 analog (Anmelden 3., 2.) bis auf das Senden an Nachbarvermittler (1(a), 1(b)).

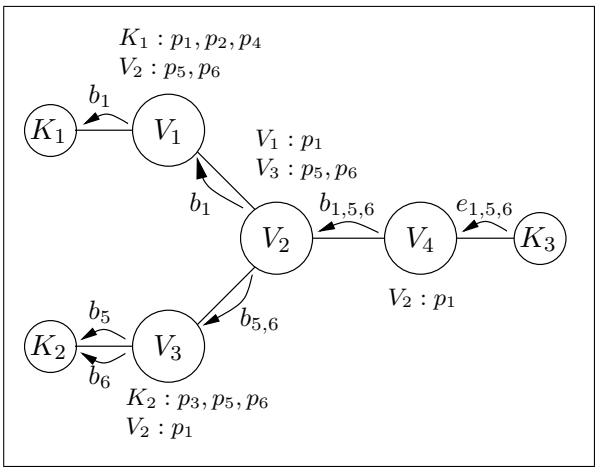


Abbildung 5.5: Filterung und Filterstruktur bei der Profilweiterleitung.

Abb. 5.5 zeigt eine Konfiguration mit vier Vermittlern V_1 bis V_4, zwei Abonnenten K_1 und K_2 sowie einem Anbieter K_3. K_1 hat die Profile p_1, p_2 und p_4 angemeldet, K_2 die Profile p_3, p_5 und p_6. Analog zum vorherigen Beispiel nehmen wir folgende Bedeckungseigenschaften an: $p_1 \sqsupseteq p_2 \sqsupseteq p_5$ und $p_1 \sqsupseteq p_6 \sqsupseteq p_3 \sqsupseteq p_4$. Anbieter K_3 sendet ein Ereignis $e_{1,5,6}$, welches die Profile p_1, p_5 und p_6 erfüllt. Dieses Ereignis wird zum lokalen Vermittler V_4 gesendet. Es passt zum Profil p_1, somit wird Vermittler V_2 benachrichtigt. V_2 filtert das in der Benachrichtigung $b_{1,5,6}$ enthaltene Ereignis $e_{1,5,6}$ und benachrichtigt V_1 und V_3. V_3 wird jedoch nur einmal benachrichtigt, obwohl sowohl p_5 als auch p_6 zum Ereignis passen. V_1 filtert und benachrichtigt K_1. Entsprechend geht die Filterung in V_3 vonstatten, allerdings werden hier zwei Benachrichtigungen an den Abonnenten K_2 versendet, jedoch keine an V_2, da dies die Quelle der Benachrichtigung ist.

5.2.3 Rendezvousknoten

Auch das Protokoll der Rendezvousknoten nutzt Bedeckungen zur Verminderung von Profilredundanzen. Es gestaltet sich wie folgt:

Profilbehandlung

Anmelden eines Profils p_x des Typs T_x an einem Vermittler V_x durch Nachbar V_y:

1. Falls $V_y \in \mathcal{N}(V_x)$ (V_y ist ein Vermittler):

$\mathcal{P} = \mathcal{P} \setminus \{p_y \in \mathcal{P}_{\sqsupseteq}(p_x) | \mathcal{A}(p_y) = V_y\}$ (entferne alle von Profil p_x bedeckten und zu p_x äquivalenten Profile mit V_y als Abonnenten aus der Filterstruktur).

2. Falls $T_x \notin \mathcal{T}(V_x)$ (V_x nicht für die Filterung von T_x zuständig ist) und $\mathcal{P}_{\sqsubseteq}(p_x) = \emptyset$ (kein p_x bedeckendes oder äquivalentes Profil existiert):

   Sende p_x in Richtung Rendezvousknoten des Typs T_x.

3. $\mathcal{P} = \mathcal{P} \cup \{p_x\}$ (füge Profil p_x zur Filterstruktur hinzu).

Abmelden eines Profils p_x des Typs T_x von einem Vermittler V_x:

1. Falls $T_x \notin \mathcal{T}(V_x)$ (V_x nicht Rendezvousknoten von T_x ist) und $\mathcal{P}_{\sqsubseteq}(p_x) = \emptyset$ (keine p_x bedeckenden oder äquivalenten Profile existieren):

   (a) Sende in Richtung des Rendezvousknotens von T_x ein Profil aus $\mathcal{P}_{\equiv}(p_x)$, falls $\mathcal{P}_{\equiv}(p_x) \neq \emptyset$, sonst die Profilmenge $\widetilde{\mathcal{P}}_{\sqsupset}(p_x)$ (ein zu p_x äquivalentes Profil (falls ein solches existiert) bzw. die von p_x direkt bedeckten Profile).

   (b) Sende die Abmeldung von p_x in Richtung des Rendezvousknotens von T_x.

2. $\mathcal{P} = \mathcal{P} \setminus \{p_x\}$ (entferne Profil p_x aus der Filterstruktur).

Ereignisbehandlung

Veröffentlichen eines Ereignisses e_x des Typs T_x an einen Vermittler V_x:

1. Filtere e_x und benachrichtige die Nachbarn $V_y \in \{\mathcal{A}(p_x) | p_x \in \mathcal{P}, e_x \succ p_x\}$ (Abonnenten zu e_x passender Profile) wie folgt:

   Falls $V_y \in \mathcal{N}(V_x)$ (Nachbar V_y ist Vermittler), benachrichtige V_y nur genau einmal über Ereignis e_x, sonst benachrichtige V_y über jedes zu Ereignis e_x passende Profil.

2. Falls $T_x \notin \mathcal{T}(V_x)$ (V_x nicht Rendezvousknoten von T_x ist):

   Sende e_x in Richtung des Rendezvousknotens von T_x.

Benachrichtigungsbehandlung

Veröffentlichen einer Benachrichtigung b_x an einem Vermittler V_x durch einen Nachbarn V_y:

Filtere das in b_x enthaltene Ereignis e_x und benachrichtige die Nachbarn $V_z \in \{\mathcal{A}(p_x) | \mathcal{A}(p_x) \neq V_y, p_x \in \mathcal{P}, e_x \succ p_x\}$ (Nachbarn V_z passender Profile außer V_y) wie folgt:

Falls $V_z \in \mathcal{N}(V_x)$ (Nachbar V_z ist Vermittler), benachrichtige diesen nur genau einmal über Ereignis e_x, sonst benachrichtige V_z über jedes zu Ereignis e_x passende Profil.

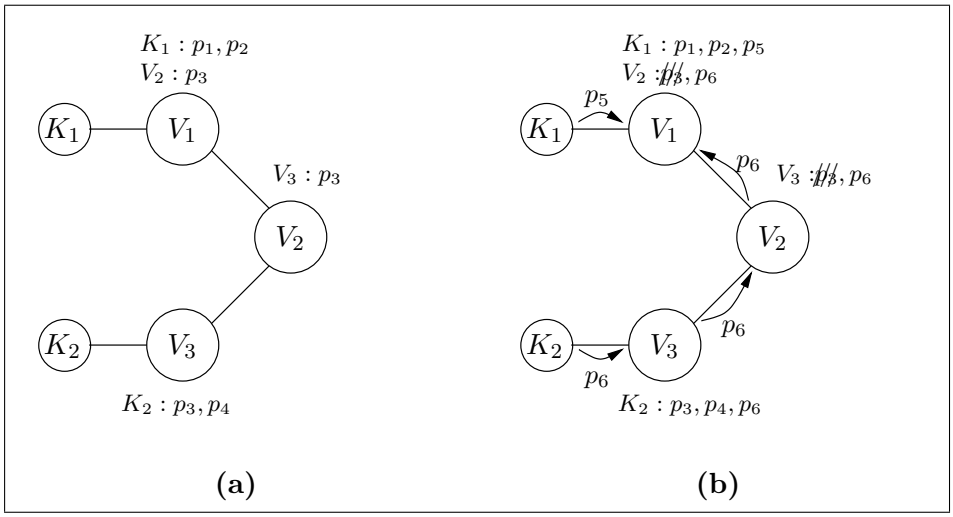


Abbildung 5.6: Nachrichten und Filterstruktur bei Rendezvousknoten.

Beispiele

Abb. 5.6 zeigt eine Beispielkonfiguration der Rendezvousknoten mit drei Vermittlern und zwei Abonnenten. Die Profile besitzen erneut folgende Bedeckungseigenschaften: $p_1 \sqsupseteq p_2 \sqsupseteq p_5$ und $p_1 \sqsupseteq p_6 \sqsupseteq p_3 \sqsupseteq p_4$. Zwischen nicht genannten Profilen besteht Unabhängigkeit. V_1 ist der Rendezvousknoten der Profile. In Abb. 5.6(a) ist die Ausgangskonfiguration mit den Profilen p_1 und p_2 von Abonnent K_1 und p_3 und p_4 von Abonnent K_2 dargestellt. Über den Vermittlern sind die zu filternden Profile aufgetragen. Abb. 5.6(b) zeigt Änderungen in den zu filternden Profilen und auftretende Nachrichten beim Anmelden der Profile p_5 und p_6. Profil p_5 wird an den lokalen Vermittler V_1 ausgeliefert. Dort wird das Profil zur Filterstruktur hinzugefügt (3.). Profil p_6 wird an den lokalen Vermittler von K_2, V_3, ausgeliefert. Es existiert kein bedeckendes Profil (2.), somit wird p_6 in Richtung des Rendezvousknotens, zu V_2, ausgeliefert (2.) und danach zur Filterstruktur hinzugefügt (3.). Profil p_6 erreicht V_2, es wurde von einem Vermittler gesendet (1.). Somit wird p_3, ebenfalls von V_3, aus der Filterstruktur entfernt (1.). V_2 ist kein Rendezvousknoten, somit erfolgt die Weiterleitung an V_1 (2.) und die Eintragung in die Filterstruktur (3.). Profil p_6 trifft beim Vermittler V_1 ein. Dort wird p_3 entfernt (1.). Danach wird p_6 zur Filterstruktur hinzugefügt (3.).

Beim Abmelden der Profile p_5 und p_6 erhält man wieder die Ausgangssituation gemäß Abb. 5.6(a): Der lokale Vermittler V_1 erhält die Abmeldung von p_5. Daraufhin wird p_5 aus der Filterstruktur entfernt (2.). V_3 erhält die Abmeldung von p_6, der kein Rendezvousknoten ist und keine p_6 bedeckenden Profile enthält. Somit wird Profil p_3, das direkt von p_6 bedeckte Profil, in Richtung Rendezvousknoten gesendet (1(a)) und die Abmeldung von p_6 ebenfalls in diese Richtung geschickt (1(b)). Danach wird p_6 aus der Filterstruktur entfernt (2.). Profil p_3 erreicht V_2. Dort existieren keine bedeckten oder äquivalenten Profile (Anmelden 1.), allerdings ein bedeckendes. Also greift der Fall „Anmelden 2." nicht. Schließlich wird p_3 zur Filterstruktur hinzugefügt (Anmelden 3.). Danach erreicht die Abmeldung von p_6 Vermittler V_2. Somit wird p_3 an V_1 gesendet (1(a)), ebenfalls wird die Abmeldung von p_6 zu V_1 versendet (1(b)).

Danach wird p_6 entfernt (3.). In V_1 wird beim Anmelden von Profil p_3 lediglich die Filterstruktur um dieses erweitert (Anmelden 3.). Beim Abmelden von p_6 wird dieses entfernt (2.).

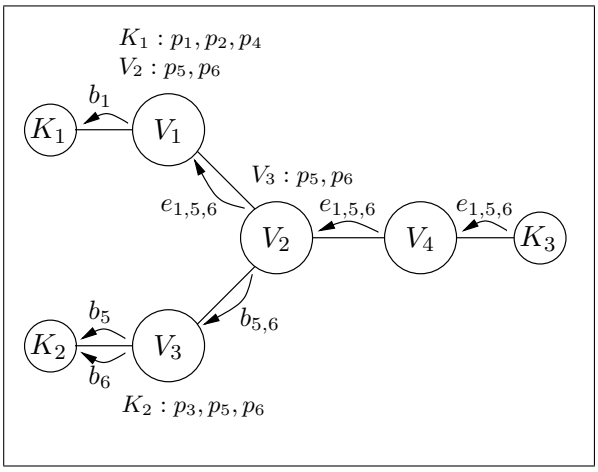


Abbildung 5.7: Filterung und Filterstruktur bei Rendezvousknoten.

Abb. 5.7 enthält eine Konfiguration mit den Vermittlern V_1 bis V_4, zwei Abonnenten K_1 und K_2 sowie einem Anbieter K_3. K_1 meldet die Profile p_1, p_2 und p_4 an, Abonnent K_2 die Profile p_3, p_5 und p_6. Es gelten die gleichen Bedeckungseigenschaften wie im vorigen Beispiel. Anbieter K_3 sendet jetzt Ereignis $e_{1,5,6}$, welches p_1, p_5 und p_6 erfüllt. Vermittler V_4 enthält keine Filtereinträge (1.) und ist kein Rendezvousknoten (2.), somit wird das Ereignis zu V_2 gesendet. V_2 filtert nun $e_{1,5,6}$ und benachrichtigt V_3 genau einmal, obwohl p_5 und p_6 passend sind (1.). Da Vermittler V_2 kein Rendezvousknoten ist, wird $e_{1,5,6}$ zu V_1 gesendet (2.). Vermittler V_3 filtert das in der Benachrichtigung $b_{5,6}$ enthaltene Ereignis und benachrichtigt Abonnent K_2 zweimal über das in $b_{5,6}$ enthaltene Ereignis $e_{1,5,6}$. Vermittler V_1 filtert das Ereignis und benachrichtigt K_1 (1.). Da V_1 Rendezvousknoten ist, wird das Ereignis nicht weitergeleitet (2.).

Bestimmung der Rendezvousknoten

Pietzuch und Bacon schlagen in [27] vor, dass die Rendezvousknoten anhand von Identifikatoren von Vermittlern und Hashwerten der Ereignistypen bestimmt werden. Derjenige Knoten ist Rendezvousknoten eines Ereignistyps, dessen zufälliger Identifikator dem Hashwert des Typen numerisch am nächsten ist. Ein logisches Overlaynetz ist für das Routing der Ereignisse und Profile verantwortlich.

Hier wird ein anderer Ansatz verfolgt, um Rendezvousknoten in DAS besser festlegen zu können; z.B. sollen Vermittler mit vielen Ressourcen den Rendezvousknoten bilden. Bei der Konfiguration des Vermittlungsnetzes können für alle Ereignistypen die Rendezvousknoten gewählt werden. Das Vermittlungsnetz sorgt dann für eine netzweite Verbreitung dieser Informationen über Rendezvousknoten. Durch die eigene Festlegung der Rendezvousknoten ergeben sich Vorteile wie:

- Rendezvousknoten nutzen leistungsfähigere Rechner

- Rendezvousknoten befinden sich zentral im Vermittlungsnetz

Die netzweite Kenntnis der Rendezvousknoten ist nicht ausreichend, um die Filterung durchführen zu können, da jeder Vermittler nur mit einer Teilmenge aller Vermittler kommuniziert. Vielmehr sollte jeder Vermittler wissen, in Richtung welchen Nachbarvermittlers ein Ereignis bzw. Profil versendet werden muss, da auf diesem Weg der Rendezvousknoten zu erreichen ist. Dazu wird von jedem Vermittler eine Hashtabelle $H = \{(T, V)\}$ verwaltet. Diese enthält Tupel mit Ereignistypen T als Schlüssel und Vermittlern V als Werten. Zur Verbreitung der Rendezvousinformationen werden beim Verbinden zweier Vermittler V_1 und V_2 die bisherigen Informationen der Hashtabelle H ausgetauscht. Das heißt, dass V_1 an V_2 die Typen $\widetilde{T} = \{T_e | E = (T_e, V_e), E \in H, V_e \neq V_2\}$ sendet. V_2 fügt diese Typen zur Hashtabelle hinzu: $\forall T_e \in \widetilde{T}(H = H \cup \{T_e, V_1\})$. Dann sendet V_2 diese Typen an alle Nachbarn außer V_1. V_2 sendet seine Informationen analog zu V_1. Dadurch ist jedem Vermittler des Netzes die Richtung der Rendezvousknoten bekannt, also an welchen Nachbarn Ereignisse bzw. Profile gesendet werden müssen.

5.3 Zusammenfassung

In diesem Kapitel wurden die Protokolle der Filteralgorithmen des Systems DAS und Implementierungsvarianten zur Verminderung von Profilredundanzen vorgestellt. In Abschnitt 5.1 wurde eine Aufwandsabschätzung beim Nutzen von Verschmelzen und Bedecken durchgeführt. Aufgrund der Umsetzbarkeit des Bedeckens ohne erheblichen Netz- und Speicheraufwand, wird das Bedecken in DAS genutzt. Danach wurden der entworfene profilbasierte und der bereichsbasierte Ansatz zur Berechnung der Optimierungen beschrieben. Beide Varianten sind in DAS implementiert.

Abschnitt 5.2 beschrieb dann ausführlich die Protokolle der Filteralgorithmen: Ereignisweiterleitung, Profilweiterleitung und Rendezvousknoten. Ebenfalls wurden aussagekräftige Beispiele zur Verdeutlichung der Protokolle angegeben. Mit diesen genauen Beschreibungen wird eine Lücke in vorhandenen Arbeiten geschlossen, welche stets nur unvollständig die genauen Filterprotokolle darstellen. Mit der profilbasierten und bereichsbasierten Methode zur Berechnung der Optimierungen werden ebenfalls bisher nicht untersuchte Implementierungsvarianten aufgezeigt.

Im nächsten Kapitel wird eine umfangreiche Evaluation der drei verteilten Filteralgorithmen des Systems DAS beschrieben. Dazu sind zahlreiche Experimente und deren Auswertungen dargestellt.

Kapitel 6

Experimente und Auswertung

Inhalt

Nachdem die theoretischen Grundlagen von Benachrichtigungsdiensten und deren praktische Umsetzung im Prototyp DAS beschrieben wurden, wird im Folgenden die Implementierung zahlreichen Experimenten unterzogen. Im Verlauf dieses Kapitels werden in Abschnitt 6.1 bisherige Arbeiten zur Evaluation von Benachrichtigungssystemen beschrieben. In Abschnitt 6.2 werden der Aufbau der Experimente dieser Arbeit, deren Systemkonfiguration und weitere Annahmen der Versuche dargestellt. Die Abschnitte 6.3 bis 6.8 betrachten den Einfluss der Variation verschiedener Systemparameter in Experimenten. Dabei werden jeweils zuerst Hypothesen formuliert und danach die Messergebnisse der Versuche gezeigt, ausgewertet und die aufgestellten Hypothesen überprüft. Abschließend werden in Abschnitt 6.9 die Experimente, deren Ergebnisse und Auswertungen zusammengefasst.

6.1 Untersuchungen bisheriger Arbeiten

In diesem Abschnitt werden bisherige Arbeiten zur Evaluation verteilter Filteralgorithmen dargestellt. Zuerst werden generelle Möglichkeiten einer Evaluation beschrieben, danach die in verschiedenen Arbeiten gewählten Ansätze gezeigt.

6.1.1 Untersuchungsmöglichkeiten

Die Bewertung von verteilten Systemen stellt eine große Herausforderung dar. Eine Bewertungsmöglichkeit ist das Nutzen von Simulatoren. Dabei werden verteilte Sys-

teme mit ihren Netzknoten und -verbindungen simuliert, so dass Aussagen über die Netzlast und die Auslastung getroffen werden können. Da es sich lediglich um Simulationen handelt, ist eine Verallgemeinerung der Ergebnisse auf reale Einsatz-Szenarien fraglich. Insbesondere tatsächliche Laufzeiten können nicht ermittelt werden, sondern man kann lediglich versuchen, sie aus den Simulationen zu berechnen. Weiterhin ist das Beachten und Berechnen aller realen Einflussfaktoren in einem Simulator äußerst schwierig.

Schwergewichtige verteilte Systeme mit umfangreichen Testdaten und viel Berechnungsaufwand stellen eine besonders anspruchsvolle Aufgabe dar. Entweder müssen mehrere Rechner parallel genutzt werden, oder die Simulationen benötigen einen sehr langen Zeitraum und werden dabei von unrealistischen Faktoren, wie der Auslagerung von Speicherseiten, beeinflusst. Eine verteiltes Benachrichtigungssystem ist ein solch schwergewichtiges System mit großer Speicherauslastung und viel Berechnungsarbeit. Aussagen von Simulatoren sind damit schwer zu erhalten und praxisfern, bzw. sie liefern keine Resultate über die Effizienz eines solchen Systems unter realistischen Bedingungen. Deshalb wird im Rahmen dieser Arbeit nicht simuliert, sondern es werden Experimente mit physisch getrennten Rechnern durchgeführt. Weiterhin wird von einem vorhandenen Verbindungsnetz ausgegangen. Allerdings stößt man auch bei praktischen Experimenten auf Probleme, da Rechner in großer Zahl beschafft werden müssen. Das ist natürlich in dieser Arbeit nur in einem beschränkten Rahmen möglich.

Eine weitere Herausforderung beim Testen eines Benachrichtigungssystem ist die Auswahl der Testszenarien, also der Profile, Ereignisse und der Netztopologie. Wichtige Parameter sind z.B. die Größe der Wertebereiche, die Anzahl der Attribute, die genutzten Operatoren und die Verteilung der Werte in den Attribut-Wert-Paaren der Ereignisse bzw. den Prädikaten der Profile. Je nach Anwendung sind hier andere Parameter zu erwarten, so dass bestimmte Annahmen getroffen werden müssen. Bevor im Folgenden die Experimente dieser Arbeit beschrieben werden, werden zunächst Schwächen der Auswertungen bisheriger Arbeiten gezeigt.

6.1.2 Bisherige Untersuchungen

Carzaniga nutzt in [5] 100 Vermittler. Diese werden in einem eigens dafür implementierten Simulator getestet. Es werden Graph- und Baumstrukturen als Vermittlungsnetz angenommen. Maximal 1.000 mögliche verschiedene Profile bzw. Ereignisse existieren in den Experimenten, was ein sehr unrealistisches Szenario darstellt. Insgesamt sind höchstens 10.000 Profile am Benachrichtigungssystem angemeldet, was erneut die Frage der Aussagekräftigkeit der Experimente aufwirft. Als Kostenmaß wird lediglich die erzeugte Netzlast betrachtet. Der Berechnungsaufwand für eine Filterung wird nicht gemessen oder ausgewertet.

Mühl nutzt in [24] 107 Vermittler. Hierbei wird keine Simulation durchgeführt. Insgesamt sind maximal 120.000 Profile am System angemeldet. Es existiert jedoch lediglich ein Anbieter von Ereignissen. Zwei Arten von Experimenten werden durchgeführt: Entweder existieren 1.000 oder 100.000 mögliche verschiedene Ereignisse. Als Kosten werden hauptsächlich die Größe der Routingtabellen und die Netzlast betrachtet.

Pietzuch und Bacon haben in einer aktuellen Arbeit [28] einen Simulator (DSSim) für die Evaluation entwickelt. Ihre Experimente kommen denen dieser Arbeit am nächsten. Es werden Vergleiche des Ansatzes der Rendezvousknoten [27]

und eines Ansatzes der Profilweiterleitung von Carzaniga [5], unter Ausnutzung von Überdeckungen, durchgeführt. Die Netztopologie besteht aus maximal 1.000 Vermittlern. Höchstens 25.000 Profile sind am System angemeldet. Als Kosten werden hauptsächlich die Größe der Routingtabellen, die Anzahl der Nachrichten und die Latenzzeit betrachtet. Diese Latenzzeit beschreibt jedoch nicht die intuitiv erwartete Größe, sondern die durchschnittliche Latenzzeit pro Benachrichtigung. Diese wird bei Belastung des Systems durch mehr Profile kleiner, was der eigentlichen Vorstellung widerspricht und bei nur einem angemeldeten Profil die größten Latenzzeiten zur Folge hat.

Eine detaillierte Untersuchung verschiedener Filteralgorithmen wurde bisher noch nicht durchgeführt. Insbesondere mangelt es den bisherigen Analysen an der Variation von verschiedenen Systemparametern. Außerdem bleibt der Berechnungsaufwand, also die Filterung selbst, meist unbeachtet. In den folgenden Abschnitten wird versucht diese Lücken zu schließen.

6.2 Versuchsaufbau

Die Experimente in dieser Arbeit werden auf maximal 20 PCs mit dem Betriebssystem Linux durchgeführt. Diese nutzen als lokales Netz Ethernet der Bandbreite 100 Mb/Sek. 256 MB Hauptspeicher und 1 GHz Rechnerleistung sind die weiteren Eigenschaften der Rechner des Testsystems. Auf dieser Hardware und Netzstruktur wird das logische Overlaynetz des Benachrichtigungssystems angelegt. Im Folgenden werden die untersuchten Messgrößen und getroffene Annahmen beschrieben.

6.2.1 Messgrößen

Die Bewertung der Filteralgorithmen wird im Rahmen dieser Arbeit anhand der folgenden Messgrößen durchgeführt (entsprechend den Beurteilungskriterien in Kapitel 3).

Filtereffizienz: Die Filtereffizienz beschreibt die Anzahl der vom Benachrichtigungssystem verarbeitbaren Ereignisse je Sekunde. Der Kehrwert dieser Messgröße beschreibt die für ein Ereignis durchschnittlich benötigte Filterzeit. Eine genaue Beschreibung der Berechnung und Definition der Filtereffizienz findet im folgenden Abschnitt statt.

Netzlast pro Ereignis: Die von einem Ereignis erzeugte Netzlast ist ein Maßstab für die Weiterleitung von Ereignissen im Vermittlungsnetz. Dieser Wert kann durch die Summe der von allen Vermittlern empfangenen Daten, vom Zeitpunkt des ersten Filterns eines Ereignisses bis zur letzten Filterung, geteilt durch die Ereignisanzahl, berechnet werden.

Profilduplikation: Die durchschnittliche Anzahl der Vorkommen eines bestimmten Profils in allen Vermittlern wird durch die Profilduplikation ausgedrückt. Werte von 1,0 erhält man bei Ereignisweiterleitung, da die Profile nicht im Netz verbreitet werden. Es sind also keine Redundanzen vorhanden. Werte von 2,0 besagen, dass jedes Profil im Durchschnitt in 2 Vermittlern gespeichert ist und gefiltert werden kann. Eine Auswirkung auf das System ist insoweit zu beobachten,

dass viele Profileinträge und -redundanzen mehr Speicherplatz benötigen. Weniger Profilredundanzen ermöglichen eine Nutzung der Filteralgorithmen ohne Seitenauslagerungen bzw. mit größerer Gesamtprofilanzahl. In den Tests wurde diese großen Gesamtprofilzahlen jedoch meist vermieden. Die Berechnung erfolgt durch den Quotienten der Summe der Profile in den Vermittlern und der am Benachrichtigungssystem angemeldeten Profilanzahl.

Weiterhin wird die Zeit für das Anmelden von Profilen bestimmt. Die Ergebnisse werden jedoch hier nicht dargestellt, da ihre Relevanz, beispielsweise im Anwendungskontext der Gebäudesteuerung, minimal ist.

Im Folgenden werden die Berechnung der Messgröße Filtereffizienz, die Bestimmung der Abweichung der Messergebnisse und einige weitere für die Experimente notwendige Definitionen dargestellt.

Berechnung der Filtereffizienz

Die Messgröße Filtereffizienz entspricht den vom System verarbeitbaren Ereignissen je Sekunde. Diese Größe gibt an, ab welcher Ereignisfrequenz f_{sat} ein Stau der Ereignisse zu erwarten ist. Daraus kann die durchschnittliche Bearbeitungszeit eines Ereignisses $t_e = \frac{1}{f_{sat}}$ bestimmt werden.

Die Bearbeitungszeit t_e gibt an, wie lange ein Ereignis im Mittel gefiltert wird, bis eine garantierte Auslieferung an alle Abonnenten erfolgt ist. Das heißt, dass das Absenden der Benachrichtigungen an alle Abonnenten eines Ereignisses spätestens t_e Zeiteinheiten nach Eintreffen dieses Ereignisses beim Benachrichtigungssystem durchgeführt ist.

Die Berechnung der Größe f_{sat} einer bestimmten Ereignismenge $\widetilde{\mathcal{E}}$ geschieht durch folgenden Ablauf:

1. Sende alle Ereignisse $e \in \widetilde{\mathcal{E}}$ an die lokalen Vermittler ihrer Anbieter.
2. Wenn alle Ereignisse versendet wurden, messe die Zeit t_1^V für alle Vermittler $V \in \mathcal{V}$ und beginne die Filterung der Ereignisse.
3. Messe die Zeiten t_2^V bei jeder Filterung eines Ereignisses bzw. eines Ereignisses einer Benachrichtigung für jeden Vermittler $V \in \mathcal{V}$.
4. Wenn in allen Vermittlern $V \in \mathcal{V}$ keine Ereignisse mehr bearbeitet werden, bestimme $\triangle t^V = t_2^V - t_1^V$ für alle Vermittler $V \in \mathcal{V}$.
5. Bestimme $f_{sat} = \frac{|\widetilde{\mathcal{E}}|}{\max\{\triangle t^V | V \in \mathcal{V}\}}$

Es ist ersichtlich, dass die Zeitmessung für die Ereignisfrequenzen sich auf die Vermittler beschränkt. Die Versendung der Ereignisse zu den Vermittlern bzw. der Benachrichtigungen zu den Abonnenten hat keinen Einfluss auf das Messergebnis.

Werden mehr als f_{sat} Ereignisse pro Sekunde an das Gesamtsystem gesendet, entsteht ein Ereignisstau: Das System kann die eintreffenden Ereignisse nicht schnell genug verarbeiten. Diese werden dann beim Eintreffen nicht sofort gefiltert, sondern zwischengespeichert und erst nach Abarbeitung aller vorherigen Ereignisse der Filterung unterzogen. Bei Frequenzen unter f_{sat} kann das Gesamtsystem im Regelfall ein Ereignis sofort nach dessen Ankunft bearbeiten, da alle vorherigen Ereignisse schon

gefiltert wurden. Das Gesamtsystem sind alle Vermittler des Vermittlungsnetzes $\mathcal{V}$. Es wird angenommen, dass Ereignisse gleichverteilt von den lokalen Anbietern der Vermittler veröffentlich werden.

Bestimmung der Abweichung

Um zuverlässige Messergebnisse zu erhalten, müssen eine große Zahl an Einzelmessungen pro Experiment durchgeführt und deren Mittelwerte als Ergebnisse genutzt werden. Weiterhin sollte die Standardabweichung bzw. der Fehler des Mittelwertes bestimmt werden, um Aussagen über die Genauigkeit des erhaltenen Ergebnisses zu bekommen.

Da die für eine Fehleranalyse nötigen Testläufe aufgrund der Zeitkomplexität der einzelnen Experimente nicht bei jeder Messung erfolgen können, werden einmalig 40 Einzelmessungen eines Experiments durchgeführt. Daraus werden die Standardabweichung und der Fehler des Mittelwertes bestimmt. Im Experiment werden drei Vermittler mit Ereignisweiterleitung genutzt, an denen eine große Profilmenge angemeldet ist. Dann werden 450.000 Ereignisse gefiltert. Die Standardabweichung einer Einzelmessung der verarbeitbaren Ereignisfrequenz und der Fehler des errechneten Mittelwerts liegen bei jeweils $0,73\%$. Die anderen beiden Filteralgorithmen zeigen ähnlich kleine Abweichungen.

Eine Einzelmessung kann also als relativ stabil und repräsentativ angesehen werden. Der kleine Wert für die Abweichung sollte auf alle Experimente verallgemeinert werden können. Ein Argument dafür ist der Umstand, dass eine Einzelmessung der Ereignisfrequenz schon einen Mittelwert über einer sehr großen Zahl von Ereignissen darstellt. Im Beispiel ergibt sich die Ereignisfrequenz aus dem Mittelwert von über 450.000 Ereignissen. Nur lang anhaltende äußere Einflüsse können so die Messungen verfälschen. Dem wird allerdings durch die zeitliche Nähe der Tests einer Messreihe entgegengewirkt. In den folgenden Experimenten wird deshalb ein Mittelwert aus nur 3 Testläufen gebildet.

Die anderen Messwerte, Netzlast und Profilduplikation, bedürfen keiner statistischen Fehleranalyse, da es sich um von äußeren Einflüssen unabhängige Ergebnisse handelt.

Definitionen

Zur Charakterisierung der Experimente wird der Anteil der erfüllenden Ereignisse p^e einer Ereignismenge bzw. der Anteil der passenden Profile p^p definiert. Sie berechnen sich bei einer Ereignismenge $\widetilde{\mathcal{E}}$ wie folgt

$$p^e = \frac{|\{e_x \in \widetilde{\mathcal{E}} | \exists p_x \in \mathcal{P}(e_x \succ p_x)\}|}{|\widetilde{\mathcal{E}}|} \text{ bzw.}$$
$$p^p = \frac{\sum_{e_x \in \widetilde{\mathcal{E}}} |\{p_x \in \mathcal{P} | e_x \succ p_x\}|}{|\widetilde{\mathcal{E}}|}.$$

Der Anteil erfüllender Ereignisse p^e beschreibt, wie viele Ereignisse zu einem oder mehreren Profilen passen. Je nach Überdeckungen und Definitionen der Profile werden bei gleicher Ereignismenge $\widetilde{\mathcal{E}}$ und konstantem p^e mehr oder weniger Benachrichtigungen ausgeführt. Der Anteil passender Profile p^p beschreibt, wie viele Profile benachrichtigt

werden, also den Anteil der Benachrichtigungen. Bei der gleicher Ereignismenge $\widetilde{\mathcal{E}}$ und konstantem p^p wird stets die gleiche Anzahl an Benachrichtigungen ausgeführt. Ferner wird der Wert der Ereignisauslastung σ definiert durch

$$\sigma = \frac{p^p}{p^e} = \frac{\sum_{e_x \in \widetilde{\mathcal{E}}} |\{p_x \in \mathcal{P} | e_x \succ p_x\}|}{|\{e_x \in \widetilde{\mathcal{E}} | \exists p_x \in \mathcal{P}(e_x \succ p_x)\}|}.$$

Die Ereignisauslastung σ gibt an, wie viele Benachrichtigungen im Durchschnitt von einem Ereignis mit passenden Profilen ausgelöst werden. So bedeutet $\sigma = 1$, dass nur ein Profil von einem Ereignis mit passenden Profilen benachrichtigt wird, $\sigma = 3$ heißt, dass durchschnittlich drei Profile von einem Ereignis mit passenden Profilen benachrichtigt werden.

6.2.2 Versuchsfestlegungen

Bei einer Systemanalyse stellt sich die Frage nach der Verallgemeinerbarkeit der Testszenarien und -ergebnisse. Zur besseren Charakterisierung der Ergebnisse dieser Arbeit werden im Folgenden getroffene Festlegungen bzgl. der Attribute und der Klienten dargestellt.

Attribute

In dieser Arbeit werden Experimente mit Ereignistypen mit nur einem Attribut durchgeführt. Trotz dieser Einschränkung lässt sich jedoch auf andere Szenarien schließen. So zeigt Abb. 6.1 Messungen mit 1 bis 10 Attributen bei einer Profilmenge der Größe 10.000. Auf der Horizontalen ist die Attributanzahl eingezeichnet. Die Vertikale zeigt die resultierende Filterzeit zum Filtern von 100.000 Ereignissen mit der entsprechenden Attributanzahl. Es haben jeweils verschiedene Anteile der Ereignisse passende Profile ($p^p = p^e$, da nur eindeutige Profile verwendet werden). Bei gleicher Profilzahl ist ein Anstieg der Filterzeit bei Erhöhung der Attributanzahl zu beobachten. Also ist die Effizienz des Systems bei vielen Attributen durch einen konstanten Faktor von der Effizienz bei weniger Attributen abhängig.

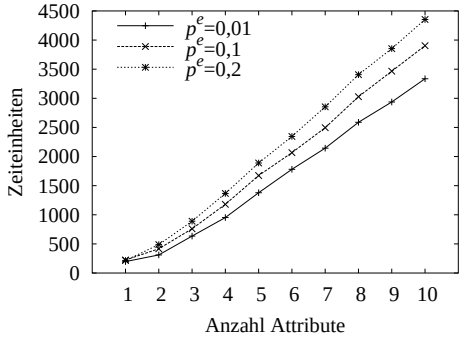


Abbildung 6.1: Filterzeit in Abhängigkeit von der Attributanzahl.

Ein wichtiger Parameter bei mehreren Attributen ist, nach wie vielen Attributauswertungen ein Ereignis ohne passende Profile als solches erkannt wird. Der genutzte

Filteralgorithmus kann diese Zahl der Auswertungen verkleinern (siehe Abschnitt 4.1.2 und [11]). In Abb. 6.1 wurden dennoch für die Filterzeiten die Mittelwerte der nötigen Attributauswertungen zum Erkennen eines Ereignisses ohne passende Profile genutzt. Folgendes Beispiel verdeutlicht diese Mittelwertbildung:

Beispiel 6.1 (Bildung des Mittelwertes der Filterzeit) *Sei der Ereignistyp T_6 durch $T_6 = \{a_1, a_2, \ldots, a_n\}$ definiert und gelte $\forall p \in \mathcal{P}(\mathcal{T}(p) = T_6)$ und $\forall e \in \widetilde{\mathcal{E}}(\mathcal{T}(e) = T_6)$. Weiterhin sei t_y die mittlere Zeit zum Erkennen eines Ereignisses ohne passende Profile nach Auswertung des Attributes a_y. Der Mittelwert zum Erkennen eines Ereignisses ohne passende Profile ist dann $\frac{1}{n} * \sum_{i=1}^{n} t_i$.*

Weiterhin werden als Vergleichsoperatoren in den Prädikaten der Profile ausschließlich Gleichheitsoperatoren angenommen. Damit können sowohl eindeutige Profile als auch Überdeckungen leicht erzeugt werden. Da die Experimente das Verhalten der Verteilung zeigen sollen, sind diese Annahmen vertretbar.

Klienten

In den Tests wird je Vermittler jeweils nur ein Abonnent bzw. Anbieter genutzt. Dieser meldet eine bestimmte (sehr große) Profilmenge an bzw. sendet (sehr viele) Ereignisse an seinen lokalen Vermittler. In realen Szenarien sind eher mehr Klienten mit weniger Profilen und Ereignissen zu erwarten. Die erhaltenen Ergebnisse können jedoch verallgemeinert werden, da die Profil- bzw. Ereignismengen für Effizienz und Skalierbarkeit entscheidend sind. Mehrere Klienten vermindern zwar die Effizienz eines Systems, da beispielsweise mehr Synchronisationsaufwand zwischen verschiedenen Threads notwendig ist. Allerdings kann dieser Aufwand auf ein Zwischensystem verlagert werden, welches mit den Klienten kommuniziert und als einziger Kommunikationspartner des Benachrichtigungssystems auftritt. Dann kommuniziert jeder Vermittler mit einem einzigen Klienten, wie in den Testszenarien angenommen.

Eine weitere Annahme bezieht sich auf das Verhalten der Klienten. Es wird davon ausgegangen, dass jeder Vermittler gleich viele Profile lokaler Abonnenten besitzt. Ebenfalls senden Anbieter gleich viele Ereignisse an ihre lokalen Vermittler. Das Abonnenten- und Anbieterverhalten wird somit als gleichartig und deren Last als gleichverteilt im gesamten Vermittlungsnetz angenommen (falls in den Experimenten nicht anders beschrieben).

6.3 Einfluss erfüllender Ereignisse und passender Profile

Als erstes Experiment wird in diesem Abschnitt der Einfluss erfüllender Ereignisse (p^e) bzw. passender Profile (p^p) auf Filtereffizienz (Abb. 6.2 und 6.3) und Netzlast (Abb. 6.4) untersucht. Die Profilduplikation wird nicht betrachtet, da während der Experimente keine Änderungen der Profile vorgenommen werden. Es wird ein verteiltes System mit vier Vermittlern genutzt. Diese sind als linearer Bus angeordnet, so dass die beiden äußeren Vermittler mit jeweils einem Nachbarn, die inneren mit je zwei Nachbarvermittlern verbunden sind. Der Rendezvousknoten des genutzten Ereignistyps ist auf einem inneren Vermittler gelegen. Jeder Vermittler verwaltet 50.000 lokale Profile. Es werden Messungen mit verschiedenen Ereignisauslastungen σ im Bereich von $\sigma = 1$ bis $\sigma = 9$ durchgeführt.

6.3.1 Hypothese

Sowohl bei steigendem p^p als auch p^e ist in allen drei Algorithmen eine fallende Filtereffizienz zu erwarten. Die Profilweiterleitung sollte, insbesondere bei geringem Anteil passender Profile und erfüllender Ereignisse, weit bessere Ergebnisse als Ereignisweiterleitung und Rendezvousknoten zeigen. Bei konstantem p^e fällt die Effizienz bei steigender Ereignisauslastung σ: Es werden die gleiche Anzahl von Filteroperationen, jedoch mehr Benachrichtigungen ausgeführt. Bei steigender Ereignisauslastung σ ist bei konstantem p^p eine bessere Effizienz zu erwarten. Insbesondere bei der Profilweiterleitung werden die gleiche Anzahl von Benachrichtigungen, die Filterung jedoch über weniger Profilen durchgeführt. Die Netzlast sollte bei Profilweiterleitung am geringsten sein (lediglich Weiterleitung von Ereignissen mit passenden Profilen), gefolgt von Rendezvousknoten (zusätzliche Weiterleitung aller Ereignisse zum Rendezvousknoten). Die Ereignisweiterleitung verursacht die maximale Netzlast, unabhängig von p^p bzw. p^e (ständiges Fluten aller Ereignisse).

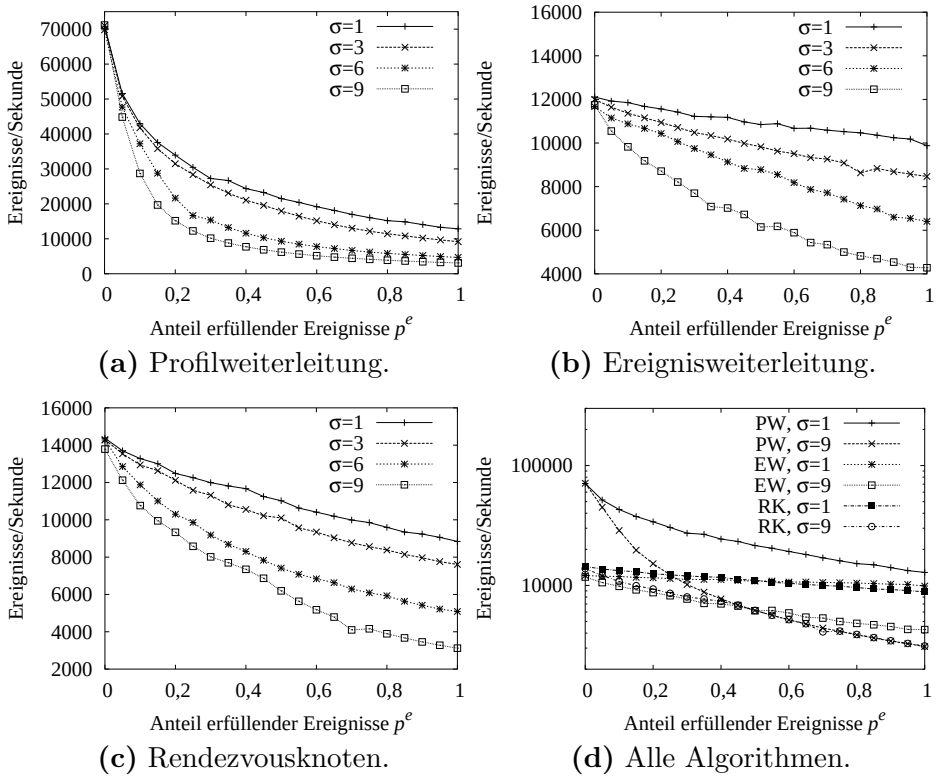
6.3.2 Messergebnisse und Auswertung

Abb. 6.2 zeigt die Filtereffizienz der drei verteilten Filteralgorithmen unter Veränderung des Anteils erfüllender Ereignisse p^e. In der Abbildung sind die unterschiedlichen Maßstäbe und die logarithmische Einteilung der Ordinate in Abb. 6.2(d) zu beachten. Die Profilweiterleitung (Abb. 6.2(a)) ist bei kleinem p^e sehr effizient. Bis zu 71.000 Ereignisse können pro Sekunde gefiltert werden. Mit steigendem p^e ist ein zunächst starker, dann ständig geringer werdender Abfall der Filtereffizienz zu beobachten. Gründe sind die Kosten für Benachrichtigungen der Abonnenten und Mehrfachfilterungen aufgrund der Weiterleitung von Ereignissen im System. Bei hoher Filterfrequenz fallen diese, stets gleich aufwendigen Mehrkosten mehr ins Gewicht als bei kleinerer. Passt jedes Ereignis zu genau einem Profil ($p^e = 1$), fällt die Effizienz auf 13.000 Ereignisse pro Sekunde. Passen jeweils 9 Profile zu jedem Ereignis ($p^e = 1, \sigma = 9$), werden nur noch ca. 3.000 Ereignisse pro Sekunde gefiltert.

Die Ereignisweiterleitung (Abb. 6.2(b)) zeigt nur geringe Veränderungen bei steigendem p^e. Hier sind keine zusätzlichen Filteroperationen notwendig, da stets alle Ereignisse im gesamten Vermittlungsnetz geflutet werden. Lediglich die Anzahl der Benachrichtigungen steigt an. Dies resultiert auch in einem etwa gleichmäßigen Abfall der Effizienz bei steigender Ereignisauslastung σ. Erfüllt kein Ereignis Profile, werden etwa 12.000 Ereignisse pro Sekunde gefiltert. Erfüllt jedes Ereignis Profile, liegen die Werte bei 10.000 (σ=1) bzw. 4.300 (σ=9).

Rendezvousknoten (Abb. 6.2(c)) zeigen mehr Beeinflussung durch steigendes p^e als die Ereignisweiterleitung, jedoch weit weniger als die Profilweiterleitung. Hier werden zum einen zusätzliche Benachrichtigungen ausgeführt, zum anderen Mehrfachfilterungen. Deren Anzahl ist jedoch viel geringer als bei der Profilweiterleitung, da die Ereignisse stets zum Rendezvousknoten weitergeleitet werden, unabhängig davon, ob Profile passen. Bei $p^e = 0$ können ca. 14.200 Ereignisse pro Sekunde gefiltert werden. Auf Werte von etwa 8.800 (σ=1) bzw. 3.100 (σ=9) Ereignissen pro Sekunde fällt die Filterfrequenz, wenn jedes Ereignis Profile erfüllt.

Abb. 6.2(d) zeigt die drei Algorithmen im Vergleich (logarithmische Ordinate). Bei $p^e = 1$ nähern sich die Algorithmen stark an, da alle Ereignisse in jedem Vermittler zu filtern sind. Je nach der Ereignisauslastung σ ist die Profilweiterleitung bzw.



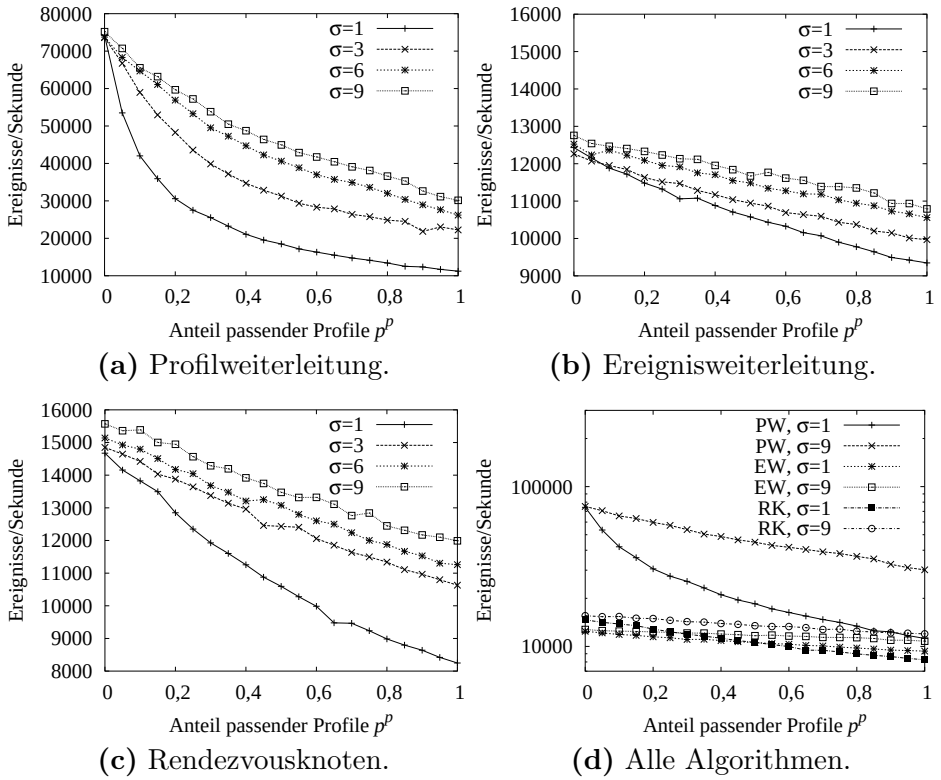
(a) Profilweiterleitung. **(b)** Ereignisweiterleitung.

(c) Rendezvousknoten. **(d)** Alle Algorithmen.

Abbildung 6.2: Filtereffizienz der verteilten Algorithmen abhängig vom Anteil erfüllender Ereignisse p^e.

die Ereignisweiterleitung die effizienteste Variante. Ein großes σ wirkt bei großem p^e der Profilweiterleitung entgegen, da in jedem Vermittler alle Profile gefiltert werden und Benachrichtigungen an Nachbarvermittler nur einmal verschickt werden dürfen (mehr Berechnungsaufwand). Bei der Ereignisweiterleitung (Fluten der Ereignisse) wird jedoch dieser zusätzliche Aufwand nicht erzeugt. Bei $p^e = 0$ ist die Profilweiterleitung weit besser als die anderen beiden Algorithmenvarianten. Schon beim lokalen Vermittler der Anbieter werden die Ereignisse ohne passende Profile (in diesem Fall alle) verworfen. Rendezvousknoten sind bei kleinem p^e effizienter als Ereignisweiterleitung. Bei steigendem p^e wird dieser Algorithmus jedoch ineffizienter. Ursache ist der zusätzliche Berechnungsaufwand beim Versand von Benachrichtigungen analog zur Profilweiterleitung (s.o.). Dieser ist auch bei σ=1 gegeben, da Benachrichtigungen nicht in Richtung des Pfades des gefilterten Ereignisses gesendet werden dürfen.

Die Filtereffizienz unter Veränderung des Anteils passender Profile (p^p) ist in Abb. 6.3 dargestellt (erneut sind die unterschiedlichen Maßstäbe und die logarithmische Einteilung der Ordinate in Abb. 6.3(d) zu beachten). Auch hier zeigt die Profilweiterleitung (Abb. 6.3(a)) die besten Ergebnisse. Passen keine Profile zu Ereignissen werden ca. 72.000 Ereignisse pro Sekunde gefiltert. Passen alle Profile zu Ereignissen, können 11.000 (σ=1) bis 30.000 (σ=9) Ereignisse pro Sekunde gefiltert werden. Bei großer Ereignisauslastung σ ist die Effizienz besser: Die Zahl von Mehrfachfilterungen in Vermittlern wird vermindert, die Anzahl der Benachrichtigungen insgesamt bleibt



(a) Profilweiterleitung. **(b)** Ereignisweiterleitung.

(c) Rendezvousknoten. **(d)** Alle Algorithmen.

Abbildung 6.3: Filtereffizienz der verteilten Algorithmen abhängig vom Anteil passender Profile p^p.

dabei konstant. Mit steigender Ereignisauslastung σ fällt dieser Gewinn durch weniger Filteroperationen immer weniger ins Gewicht (schnelle Filterung im Vergleich zu aufwendigen Benachrichtigungen).

Die Ereignisweiterleitung (Abb. 6.3(b)) zeigt erneut wenig Veränderung. Die Filtereffizienz liegt bei $p^p = 0$ bei ca. 12.500 Ereignissen pro Sekunde. Bei $p^p = 1$ fällt diese auf 9.350 (σ=1) bzw. 10.800 (σ=9). Das liegt daran dass, unabhängig von p^p, alle Ereignisse im gesamten Netz geflutet werden. Der Effizienzvorteil bei großem σ ergibt sich, da insgesamt zwar die gleiche Anzahl an Profilen benachrichtigt wird, jedoch weniger Ereignisse passende Profile besitzen. Die Filterung der Ereignisse ohne passende Profile kann schon eher abgebrochen werden (keine Bestimmung, welche Profile passend sind).

Die Beeinflussung der Rendezvousknoten (Abb. 6.3(c)) liegt bei steigendem p^p zwischen derjenigen der Ereignisweiterleitung und Profilweiterleitung. Bei großer Ereignisauslastung σ werden zwar alle Ereignisse zum Rendezvousknoten geleitet, jedoch weniger in den Rest des Vermittlungsnetzes. Dadurch liegt die Effizienz bei ca. 15.500 Ereignissen pro Sekunde ($p^p = 0$) bzw. 8.300 bis 12.000 ($p^p = 1$).

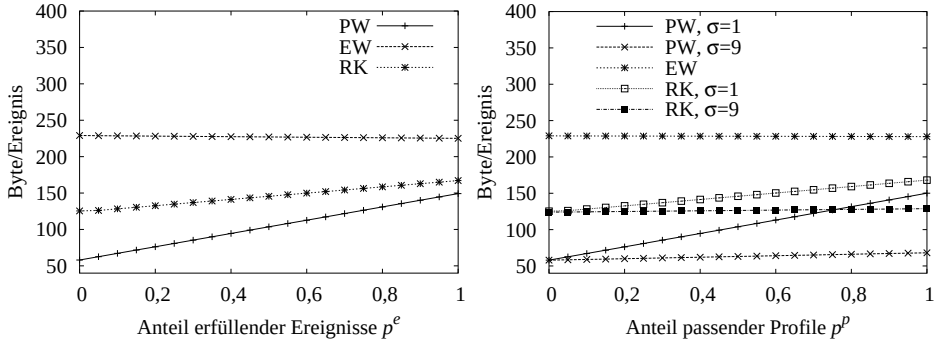
Vergleichend sind die drei Algorithmen in Abb. 6.3(d) dargestellt. Die Annäherung der Algorithmen bei großen p^p und σ ist nicht so stark wie bei großen p^e, da zu vielen Ereignissen keine passenden Profile existieren. Die Profilweiterleitung verhindert in diesem Fall die Weiterleitung solcher Ereignisse ohne passende Profile. Bei $\sigma = 1$

ist das Verhalten mit dem unter konstanten p^e bei gleicher Ereignisauslastung σ vergleichbar (s.o.): Ereignisweiterleitung ist effizienter als Rendezvousknoten.

Das erwartete Verhalten der Profilweiterleitung als eines sehr effizienten Algorithmus hat sich in diesen Experimenten bestätigt. Ebenfalls wurde die Hypothese des Einflusses einer steigenden Ereignisauslastung σ erhärtet: Bei konstantem p^e fällt die Filtereffizienz, bei konstantem p^p steigt sie.

Die Netzlast beim Veröffentlichen von Ereignissen zeigt Abb. 6.4. Bei konstantem p^e (Abb. 6.4(a)) hat die Ereignisauslastung σ keinen Einfluss auf die Netzlast. In jedem Fall werden gleich viele Ereignisse im Netz verteilt (es ist die Last unter den Vermittlern dargestellt, nicht diejenige zu Abonnenten). Die Ereignisweiterleitung flutet alle Ereignisse und zeigt wie erwartet die größte Netzlast. Rendezvousknoten senden alle Ereignisse mindestens zum Rendezvousknoten, weiter verbreitet werden jedoch nur solche mit passenden Profilen (erwartungsgemäß weniger Netzlast als bei der Ereignisweiterleitung). Am wenigsten Netzlast zeigt die Profilweiterleitung, da lediglich solche Ereignisse an Nachbarn verbreitet werden, die im Teilnetz dieses Nachbarn Profile erfüllen.

Bei konstantem p^p (Abb. 6.4(b)) wird die Netzlast von der Ereignisauslastung σ beeinflusst (außer bei Ereignisweiterleitung mit dem Fluten aller Ereignisse). Steigendes σ hat fallende Netzlasten zur Folge, da weniger Ereignisse die gleiche Profilanzahl erfüllen. Somit werden weniger Ereignisse im Netz verteilt. Diese führen dann zu Mehrfachbenachrichtigungen.



(a) Erfüllende Ereignisse p^e. **(b)** Passende Profile p^p.

Abbildung 6.4: Netzlast der verteilten Algorithmen beim Veröffentlichen abhängig vom Anteil erfüllender Ereignisse p^e bzw. passender Profile p^p.

6.4 Einfluss der Vermittlerzahl

Dieser Abschnitt betrachtet den Einfluss der Vermittlerzahl auf Filtereffizienz, Profilduplikation und Netzlast beim Veröffentlichen. Weiterhin wird die Verteilungseffizienz e betrachtet. Die Verteilungseffizienz e gibt an, wie parallelisierbar ein Algorithmus ist und berechnet sich durch $e = \frac{S}{|\mathcal{V}|}$. S stellt dabei den Speed-up dar, der durch $S = \frac{f_{sat}^v}{f_{sat}}$ berechnet wird. f_{sat}^v bezeichnet die Ereignisfrequenz bei einem Vermittlungsnetz $\mathcal{V}$ der

Größe $|\mathcal{V}| = v$.

In den Experimenten wird ein spezielles Vermittlungsnetz $\mathcal{V}$ ausgewählt. Dieses ist in Abb. 6.5 dargestellt. Genau ein Ereignistyp wird genutzt. $|\mathcal{V}|$ variiert dabei von 1 bis 9. Vermittler V_2 stellt in den Fällen $|\mathcal{V}| > 2$ den Rendezvousknoten des genutzten Ereignistyps dar (sonst Vermittler V_1). Insgesamt sind 200.000 eindeutige Profile am System angemeldet. Im Folgenden werden Experimente mit verschiedenem Anteil passender Profile p^p durchgeführt.

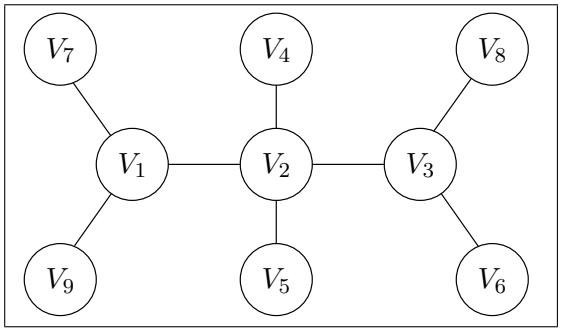


Abbildung 6.5: Netztopologie des Vermittlungsnetzes.

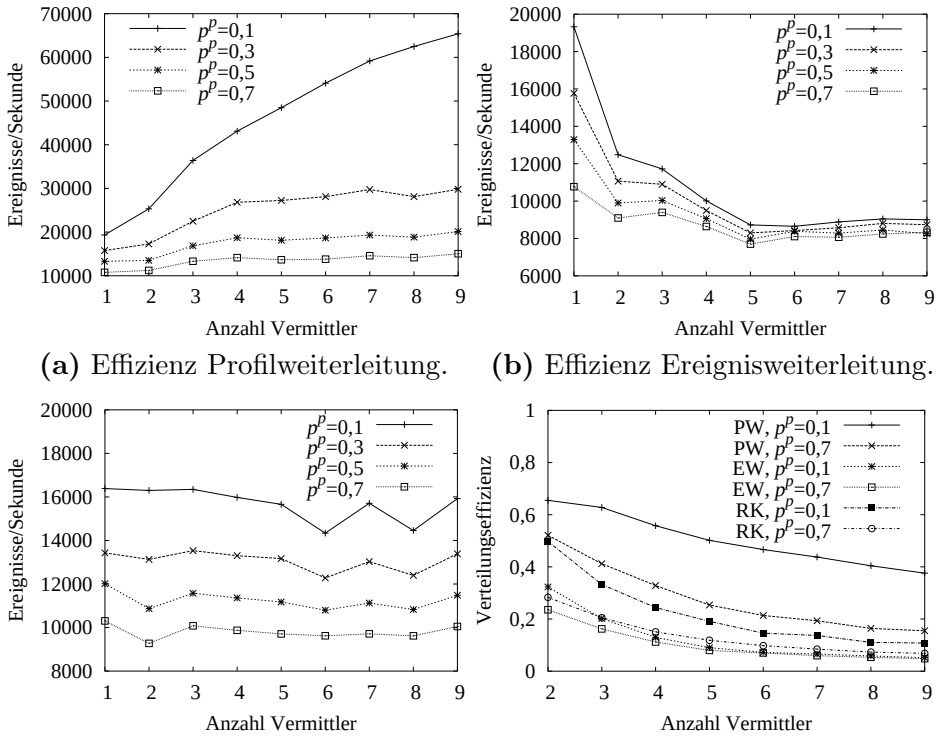
6.4.1 Hypothese

Die Profilweiterleitung sollte durch Hinzunahme von Vermittlern einen Effizienzgewinn verzeichnen, da der Filteraufwand unter den Vermittlern aufgeteilt wird. Die Ereignisweiterleitung sollte jedoch keine Vorteile durch die Verteilung des Benachrichtigungsdienstes zeigen, da stets alle Ereignisse in jedem Vermittler gefiltert werden. Die Filtereffizienz des Algorithmus Rendezvousknoten sollte zwischen den beiden anderen Filtervarianten liegen (ständige Weiterleitung von Ereignissen zum Rendezvousknoten).

Die Netzlast sollte bei allen drei Verfahren ansteigen. Die Profilweiterleitung verursacht am wenigsten Netzlast, gefolgt von den Rendezvousknoten. Die Profilduplikation sollte sich entgegengesetzt verhalten: Die Ereignisweiterleitung dupliziert keine Profile (also Profilduplikation von 1,0), die Profilweiterleitung alle Profile (große Profilduplikation). Rendezvousknoten sind ein Mittelweg. Die Profilweiterleitung zeigt die beste Verteilungseffizienz e, die Ereignisweiterleitung die schlechtesten Werte (extremer zusätzlicher Kommunikationsaufwand zwischen den Vermittlern).

6.4.2 Messergebnisse und Auswertung

Abb. 6.6 zeigt das Verhalten der Filter- und Verteilungseffizienz der Algorithmen. In der Abbildung sind die unterschiedlichen Maßstäbe der Ordinaten zu beachten. Die Filtereffizienz der Profilweiterleitung (Abb. 6.6(a)) steigt bei Hinzunahme von Vermittlern an. Bei $p^p = 0,1$ werden durch einen Vermittler ca. 20.000 Ereignisse pro Sekunde verarbeitet. Neun Vermittler erreichen Werte von 65.000 Ereignissen pro Sekunde. Steigendes p^p verringert die Filtereffizienz. Bei Hinzunahme von Vermittlern erfolgen dann nur noch geringe Effizienzsteigerungen, da die Hauptlast in den Vermittlern durch Benachrichtigungen erzeugt wird (diese werden dann zwar auf mehrere



(a) Effizienz Profilweiterleitung. **(b)** Effizienz Ereignisweiterleitung.

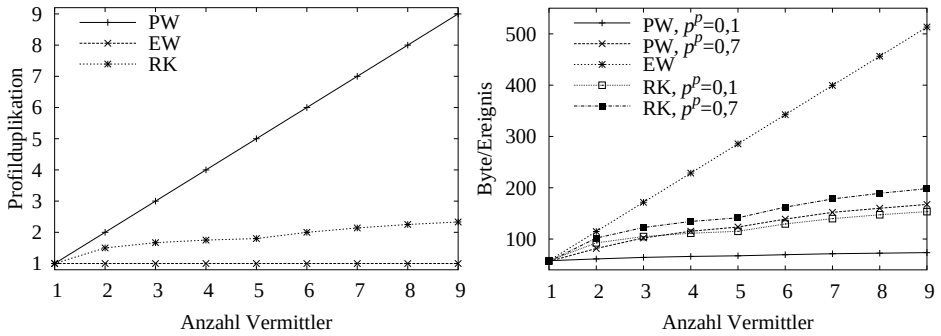
(c) Effizienz Rendezvousknoten. **(d)** Verteilungseffizienz.

Abbildung 6.6: Filtereffizienz und Verteilungseffizienz der verteilten Algorithmen abhängig von der Vermittlerzahl.

Vermittler verteilt, die Entlastung eines Vermittlers wirkt sich jedoch immer geringer aus). Dieser Verlauf entspricht dem erwarteten Verhalten der Profilweiterleitung.

Die Ereignisweiterleitung (Abb. 6.6(b)) zeigt eine fallende Effizienz bei Hinzunahme von Vermittlern. Bei großem $|\mathcal{V}|$ ist der Einfluss von p^p gering, da der Aufwand der zusätzlichen Benachrichtigungen im Vergleich zum ohnehin vorhandenen Kommunikationsaufwand aller Ereignisse klein ist. Durch diesen stark ansteigenden Kommunikationsaufwand ist der Effizienzeinbruch bei steigendem $|\mathcal{V}|$ zu erklären. Bei fünf Vermittlern ist ein Minimum erreicht, da Vermittler V_2, der Flaschenhals des Systems, hier die größte Belastung erfährt. Die Ereignisfrequenzen reichen von 19.000 bzw. 10.000 Ereignissen pro Sekunde bei $|\mathcal{V}|$=1 bis zu ca. 8.500 bei $|\mathcal{V}|$=9. Dies stärkt die im vorigen Abschnitt aufgestellte Hypothese des Verhaltens der Ereignisweiterleitung.

Rendezvousknoten (Abb. 6.6(c)) zeigen fast keine Veränderung der Filtereffizienz bei Erweiterung des Vermittlungsnetzes. Der Rendezvousknoten als Flaschenhals des Systems muss stets die gleiche Anzahl an Filteroperationen durchführen. Teilweise sind Effizienzverminderungen bei steigender Vermittlerzahl zu beobachten. Der Grund ist die auftretende Asymmetrie des Netzes. Dadurch werden einige Vermittler mehr belastet als andere, da die Profile und Ereignisse in den Experimenten gleichverteilt sind. Es werden Frequenzen von ca. 10.000 ($p^p = 0,7$) bis 16.000 ($p^p = 0,1$) Ereignissen pro Sekunde erreicht. Entsprechend der formulierten Hypothese liegt der Effizienzver-



(a) Profilduplikation. **(b)** Netzlast beim Veröffentlichen.

Abbildung 6.7: Profilduplikation und Netzlast der verteilten Algorithmen abhängig von der Vermittlerzahl.

lauf bei steigender Vermittlerzahl zwischen dem der Profilweiterleitung und dem der Rendezvousknoten.

Abb. 6.6(d) zeigt die Verteilungseffizienz e der drei Algorithmen. Die besten Werte erreicht die Profilweiterleitung. Hier wird eine relativ gute Lastaufteilung unter den Vermittlern erreicht, da nur wirklich notwendige Ereignisse weitergeleitet werden. Bei neun Vermittlern wird so noch ein Wert von $e = 0,4$ erreicht. Insgesamt sind die Ergebnisse recht ernüchternd, da der Kommunikationsaufwand zwischen den Vermittlern die Verteilungseffizienz stark mindert (insbesondere bei Ereignisweiterleitung, Rendezvousknoten und großem p^p). Der Verlauf der Verteilungseffizienz entspricht aber dem erwarteten Verhalten.

Die Profilduplikation (Abb. 6.7(a)) steigt bei der Profilweiterleitung linear an. Da keine Überdeckungen zwischen Profilen existieren, sind bei neun Vermittlern alle Profile in jedem Vermittler gespeichert. Die Ereignisweiterleitung zeigt konstante Werte von $1,0$. Die Profilduplikation der Rendezvousknoten ist dazwischen zu finden, die Werte steigen nicht über 2,5. Die Netzlast beim Veröffentlichen (Abb. 6.7(b)) verhält sich in erwarteter Weise konträr zur Profilduplikation. Die Ereignisweiterleitung zeigt einen linearen Anstieg der Netzlast. Die Profilweiterleitung verteilt nur notwendige Ereignisse im Vermittlungsnetz $\mathcal{V}$, als Resultat ist die Netzlast gering. Bei großem p^p ist der Unterschied in der Netzlast zwischen Rendezvousknoten und Profilweiterleitung gering, da die ständige Weiterleitung von Ereignissen zum Rendezvousknoten weniger Mehraufwand darstellt (die Ereignisse müssen sowieso in Richtung Rendezvousknoten verteilt werden). Die Hypothesen zur Profilduplikation und der Netzlast werden somit durch die Ergebnisse der Experimente unterstützt.

6.5 Einfluss der Überdeckungen

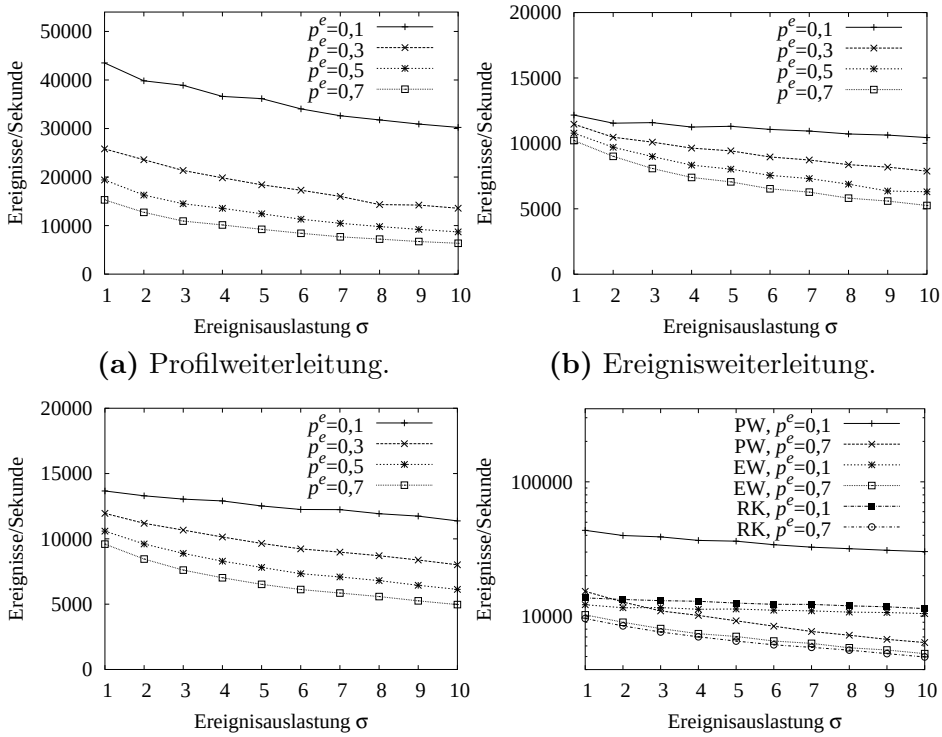
Dieser Abschnitt zeigt den Einfluss der Überdeckungen. Da nur Gleichheitsoperatoren verwendet werden, entspricht die Ereignisauslastung σ den Überdeckungen. So bedeutet $\sigma = 5$, dass zu einem Profil p_x im Durchschnitt fünf Profile p_y mit $p_x \sqsupseteq p_y$ existieren. In den Experimenten befinden sich die Überdeckungen zwischen den loka-

len Profilen jeweils eines Vermittlers. Es werden, wie in Abschnitt 6.3, Experimente sowohl bei verschiedenem p^p als auch p^e durchgeführt. Das Vermittlungsnetz $\mathcal{V}$ entspricht ebenfalls der in Abschnitt 6.3 vorgestellten Netzstruktur (Abb. 6.5). Genau ein Ereignistyp wird verwendet. 200.000 Profile sind am System angemeldet. Es werden Filtereffizienz, Profilduplikation und Netzlast bei Veröffentlichen untersucht.

6.5.1 Hypothese

Bei konstantem p^e sollte die Filtereffizienz mit steigenden Überdeckungen fallen (mehr Aufwand durch mehr Benachrichtigungen). Konstantes p^p sollte hingegen einen Effizienzgewinn mit steigenden Überdeckungen zeigen (weniger Ereignisweiterleitungen und -filterungen). Die Profilduplikation wird geringer, wenn mehr Überdeckungen vorhanden sind. Grund ist das Ausnutzen dieser Überdeckungen. Die Netzlast bei unverändertem p^e und veränderlichem σ sollte konstant sein, da die Anzahl der Benachrichtigungen in dieser Größe nicht betrachtet wird. Bei konstantem p^p und steigendem σ fällt die Netzlast, da insgesamt weniger Ereignisse weitergeleitet werden (ausgenommen die Ereignisweiterleitung mit dem Fluten der Ereignisse).

6.5.2 Messergebnisse und Auswertung

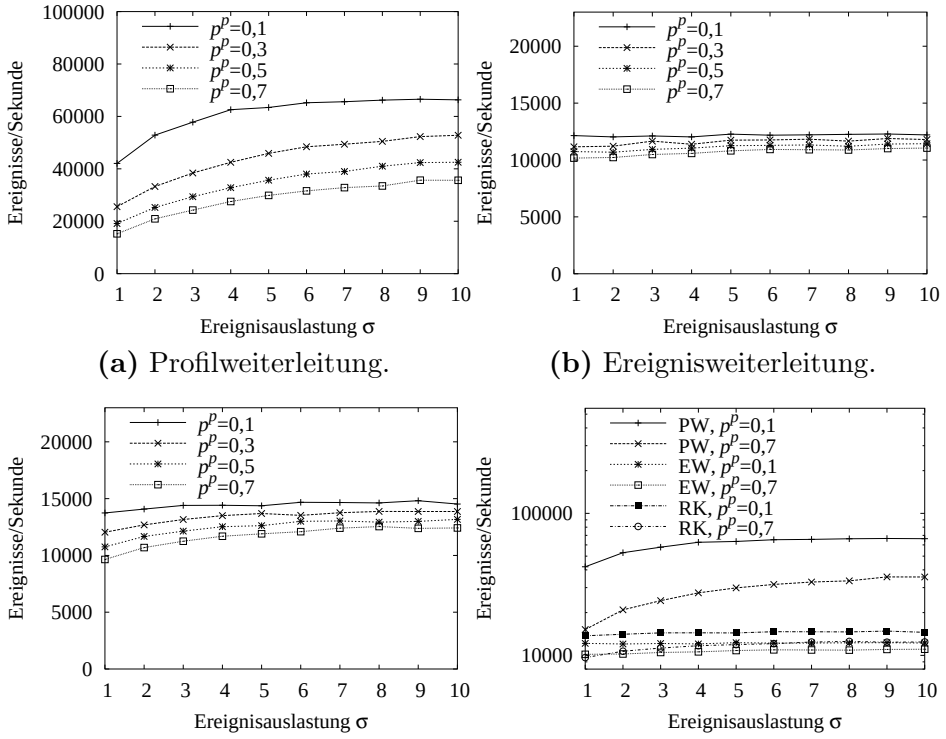


(a) Profilweiterleitung. **(b)** Ereignisweiterleitung.

(c) Rendezvousknoten. **(d)** Alle Algorithmen.

Abbildung 6.8: Filtereffizienz der verteilten Algorithmen abhängig von der Ereignisauslastung σ bei unterschiedlichem Anteil erfüllender Ereignisse p^e.

Abb. 6.8 zeigt die Filtereffizienz der Algorithmen bei verschiedenem Anteil erfüllender Ereignisse p^e. In der Abbildung sind die unterschiedlichen Maßstäbe und die logarithmische Einteilung der Ordinate in Abb. 6.8(d) zu beachten. Alle drei Algorithmen zeigen fallende Ereignisfrequenzen bei steigenden Überdeckungen, da mehr Benachrichtigungen ausgeführt werden. Bei großem p^e sind die Unterschiede zwischen den Algorithmen gering (Abb. 6.8(d), logarithmische Ordinate), da fast alle Ereignisse passende Profile haben und im Netz verteilt werden müssen. Je mehr Überdeckungen auftreten, desto weiter nähern sich die Frequenzen an: Der Aufwand zum Benachrichtigen und Verteilen der Ereignisse nimmt den meisten Teil des Gesamtaufwandes ein. Bei kleinem p^e ist die Profilweiterleitung (Abb. 6.8(a)) die mit Abstand effizienteste Variante. Je weiter p^e steigt, desto geringer fällt dabei die Verschlechterung der Filtereffizienz aus (Verhältnis Benachrichtigungsaufwand zu Gesamtaufwand). Bei der Ereignisweiterleitung (Abb. 6.8(b)) und den Rendezvousknoten (Abb. 6.8(c)) sind Effizienzverminderungen aufgrund des hohen Kommunikationsaufwandes in kleineren Dimensionen vorhanden. Bei der Profilweiterleitung werden 44.500 ($\sigma = 1, p^e = 0,1$) bis 6.500 Ereignisse pro Sekunde ($\sigma = 10, p^e = 0,7$) verarbeitet. Die Ereignisweiterleitung liegt bei den gleichen Parametern zwischen 12.000 und 5.300 Ereignissen pro Sekunde. Die Rendezvousknoten erreichen Werte von 13.700 bzw. 5.000 Ereignissen pro Sekunde. Abb. 6.9 zeigt die Filtereffizienz bei verschiedenem Anteil passender Pro-

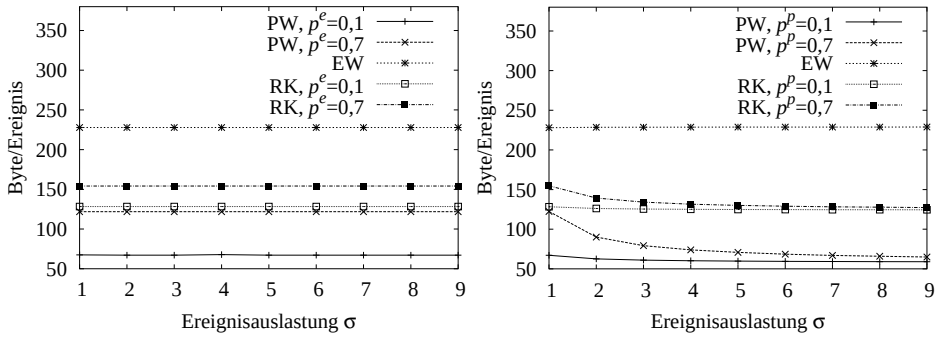


(a) Profilweiterleitung. **(b)** Ereignisweiterleitung.

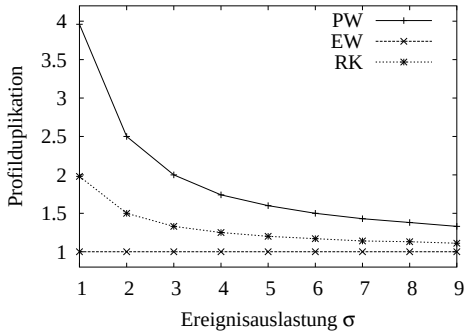
(c) Rendezvousknoten. **(d)** Alle Algorithmen.

Abbildung 6.9: Filtereffizienz der verteilten Algorithmen abhängig von der Ereignisauslastung σ bei unterschiedlichem Anteil passender Profile p^p.

file p^p (erneut sind die unterschiedlichen Maßstäbe und die logarithmische Einteilung der Ordinate in Abb. 6.9(d) zu beachten). Bei steigender Ereignisauslastung σ und konstantem p^p ist eine Verbesserung der Filtereffizienz zu beobachten. Ursache ist, dass bei gleicher Benachrichtigungsanzahl weniger Ereignisse im Netz verteilt und damit gefiltert werden müssen. Die Profilweiterleitung (Abb. 6.9(a)) zeigt die stärksten Effizienzgewinne, da keine Ereignisse ohne passendes Profil im Netz verbreitet werden. Die Frequenzen liegen zwischen 15.200 ($\sigma = 1, p^p = 0,7$) und 66.500 ($\sigma = 10, p^p = 0,1$) Ereignissen pro Sekunde. Der Abstand zu den anderen beiden Algorithmen steigt bei großer Ereignisauslastung σ (Abb. 6.9(d), logarithmische Ordinate). Dieses Verhalten ist konträr zum Effizienzverlauf bei konstantem p^e (s.o.), da hier ein gegenteiliges Benachrichtigungsverhalten auftritt. Bei den Rendezvousknoten (Abb. 6.9(c)) sind ebenfalls Verbesserungen der Effizienz zu beobachten, jedoch in geringerem Ausmaß, da die Profile auf jeden Fall zum Rendezvousknoten geleitet werden. Die erreichten Frequenzen liegen zwischen 9.600 und 15.000 Ereignissen (gleiche Parameter) pro Sekunde. Die Filterfrequenzen der Ereignisweiterleitung (Abb. 6.9(b)) sind unabhängig von der Ereignisauslastung σ, da stets alle Ereignisse im gesamten Vermittlungsnetz verbreitet werden. Der erreichte Frequenzbereich liegt zwischen 10.000 und 12.000 Ereignissen pro Sekunde.



(a) Netzlast beim Veröffentlichen. **(b)** Netzlast beim Veröffentlichen.



(c) Profilduplikation.

Abbildung 6.10: Netzlast beim Veröffentlichen und Profilduplikation der verteilten Algorithmen abhängig von den Überdeckungen σ bei unterschiedlichem Anteil erfüllender Ereignisse p^e bzw. passender Profile p^p.

Die Netzlast beim Veröffentlichen ist bei unverändertem p^e konstant (Abb. 6.10(a)), da stets die gleiche Anzahl von Ereignissen im Vermittlungsnetz verteilt wird (trotzdem werden mehr Benachrichtigungen ausgeführt). Großes p^e erhöht erwartungsgemäß die Netzlast. Die Ereignisweiterleitung zeigt die größten Werte, welche von den anderen Algorithmen nicht erreicht werden. Bei konstantem p^p (Abb. 6.10(b)) ist fallende eine Netzlast zu beobachten: Weniger Ereignisse passen zur gleichen Anzahl von Profilen. Von einem Ereignis werden also mehrere Benachrichtigungen durchgeführt, wodurch die Netzlast zwischen den Vermittlern sinkt (Weiterleitung weniger Ereignisse). Dieser Einfluss wird jedoch mit steigenden Überdeckungen stets geringer. Großes p^p erhöht die Netzlast (ausgenommen die Ereignisweiterleitung, die stets die gleiche Netzlast zeigt).

Die Profilduplikation (Abb. 6.10(c)) wird mit steigender Zahl der Überdeckungen geringer. Die Profilweiterleitung zeigt die größten Werte der Profilduplikation. Rendezvousknoten verteilen Profile seltener als die Profilweiterleitung, die Ereignisweiterleitung überhaupt nicht. Bei vielen Überdeckungen nähern sich die beiden anderen Filteralgorithmen der Ereignisweiterleitung an: Bei $\sigma = 9$ wird lediglich $\frac{1}{9}$ der Profile im Netz verteilt (Profilweiterleitung und Rendezvousknoten).

Das erwartete Verhalten der Filtereffizienz bei steigenden Überdeckungen hat sich in den Experimenten bestätigt: Bei konstantem p^e fällt die Filtereffizienz, bei konstantem p^p steigt sie. Gleichfalls wurden die Hypothesen bzgl. Profilduplikation und Netzlast erhärtet: Die Profilduplikation fällt bei steigenden Überdeckungen, die Netzlast bleibt konstant (konstantes p^e) bzw. fällt (konstantes p^p).

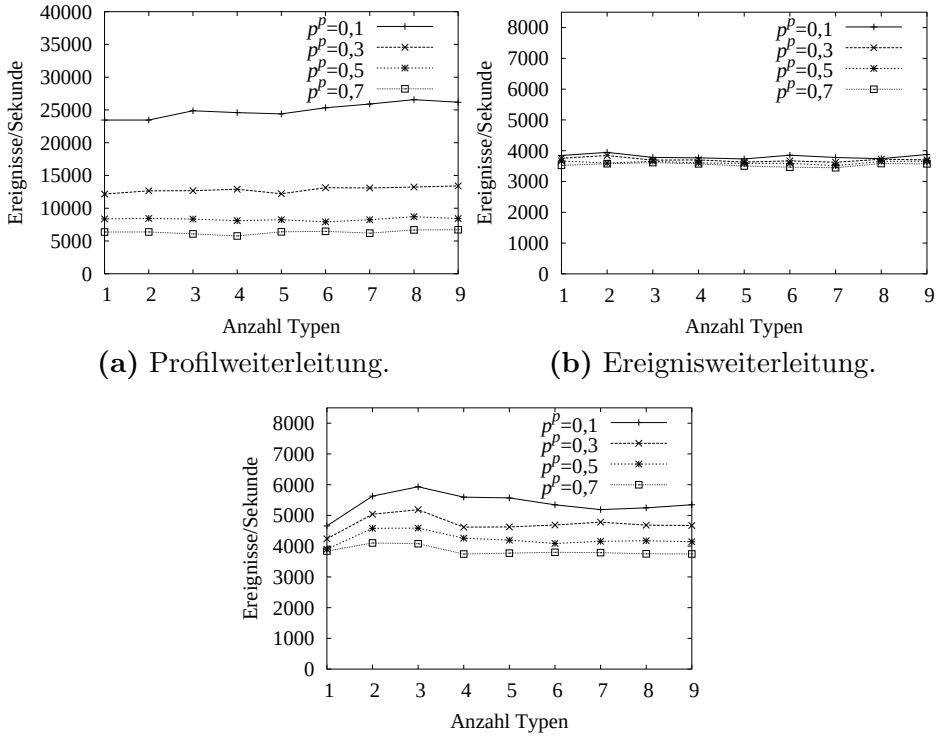
6.6 Einfluss der Typanzahl

Der Einfluss der Anzahl der Ereignistypen wird in diesem Abschnitt untersucht. In den Experimenten wird die in Abb. 6.5 dargestellte Netztopologie genutzt. Jeder der Vermittler stellt dabei den Rendezvousknoten für maximal einen Ereignistyp dar. Insgesamt sind 180.000 eindeutige Profile am System angemeldet, so sind z.B. bei drei Ereignistypen 60.000 Profile jedes Typs vorhanden. Bei n Typen sind die Vermittler V_1 bis V_n die Rendezvousknoten. Andere Vermittler fungieren nicht als Rendezvousknoten. Es werden Filtereffizienz, Profilduplikation und Netzlast beim Veröffentlichen untersucht.

6.6.1 Hypothese

Die Anzahl der Typen sollte die Filtereffizienz nur geringfügig beeinflussen. Insbesondere bei der Profil- und Ereignisweiterleitung ist die Effizienz relativ unabhängig von der Zahl der Typen. Bei den Rendezvousknoten sollte bei deren geschickter Verteilung im Vermittlungsnetz ein Effizienzgewinn zu verzeichnen sein.

Die Profilduplikation sollte, außer bei den Rendezvousknoten, unabhängig von der Typanzahl sein. Bei deren Nutzung sind die Pfade zum Rendezvousknoten für die Profilduplikation entscheidend (also Anstieg bei mehr Ereignistypen). Gleiches gilt für die Netzlast beim Veröffentlichen von Ereignissen.



(a) Profilweiterleitung.

(b) Ereignisweiterleitung.

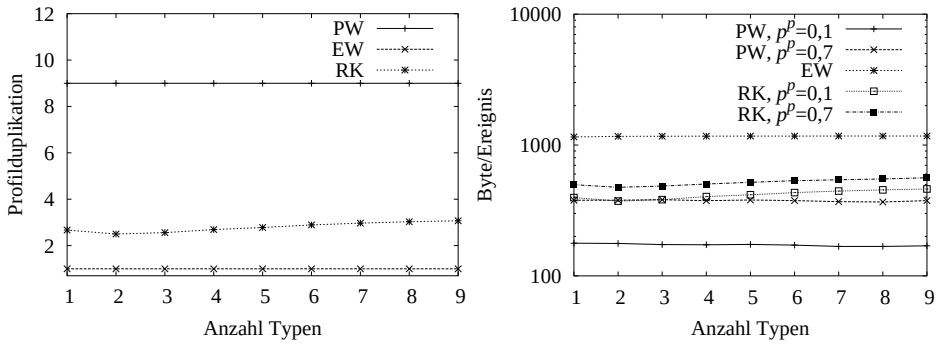
(c) Rendezvousknoten.

Abbildung 6.11: Filtereffizienz der verteilten Algorithmen abhängig von der Typanzahl bei unterschiedlichem Anteil passender Profile p^p.

6.6.2 Messergebnisse und Auswertung

Abb. 6.11 zeigt die Filtereffizienz unter Veränderung der Typanzahl. In der Abbildung sind die unterschiedlichen Maßstäbe der Ordinaten zu beachten. Die Profilweiterleitung (Abb. 6.11(a)) zeigt relativ konstante Werte (entsprechend der Hypothese) um 25.000 Ereignisse pro Sekunde ($p^p = 0,1$). Mehr passende Profile vermindern die Filtereffizienz aufgrund weiterer durchzuführender Benachrichtigungen. Die Anzahl der Typen hat fast keinen Einfluss. Lediglich eine kleine Verbesserung der Effizienz ist erkennbar. Die Begründung dafür ist der Filteralgorithmus, der für jeden Typ eine eigene Filterstruktur aufbaut (siehe Abschnitt 4.1.1). Diese ist bei mehr Typen (und damit weniger Profilen je Typ) schneller zu durchlaufen. Eine bessere Aufwandsaufteilung bzw. -einsparung zwischen den Vermittlern wird jedoch nicht erreicht.

Ähnlich verhält sich die Ereignisweiterleitung (Abb. 6.11(b)). Die erreichte Frequenz liegt um 3.750 Ereignisse pro Sekunde. Der Anteil passender Profile hat nur geringen Einfluss, da der meiste Aufwand bei diesem Algorithmus in der Verteilung aller Ereignisse liegt. Die Typanzahl ist nicht ausschlaggebend für die Effizienz des Verfahrens, da ein Fluten der Ereignisse in jedem Fall geschieht. Der bei der Profilweiterleitung sichtbare Vorteil (zentraler Filteralgorithmus) ist bei diesem Kommunikationsmodell deshalb vernachlässigbar gering.



(a) Profilduplikation. **(b)** Netzlast beim Veröffentlichen.

Abbildung 6.12: Profilduplikation und Netzlast der verteilten Algorithmen beim Veröffentlichen abhängig von der Typanzahl.

Rendezvousknoten (Abb. 6.11(c)) zeigen ein unterschiedliches Verhalten, je nach ihrer Aufteilung im Vermittlungsnetz. Sind die Rendezvousknoten zentral im Vermittlungsnetz gelegen (Fälle bis drei Typen), ist eine Verbesserung der Filtereffizienz bei Vergrößerung der Typanzahl zu beobachten (entsprechend der Hypothese). In diesem Fall werden die Rendezvousknoten entlastet, da weniger Ereignisse gefiltert werden (die Gesamtprofilanzahl bleibt konstant). Ab vier Typen fällt die Effizienz jedoch wieder. Hier liegen die Rendezvousknoten teilweise am Rand des Vermittlungsnetzes. Somit müssen fast alle Ereignisse über die inneren Knoten verteilt werden, welche dann den Flaschenhals des Systems darstellen. Die Ereignisfrequenzen liegen zwischen 4.500 und 6.000 Ereignissen pro Sekunde (bei $p^p = 0,1$).

Die Profilduplikation (Abb. 6.12(a)) ist unabhängig von der Typanzahl. Da nur eindeutige Profile verwendet werden, ist die Profilduplikation bei der Profilweiterleitung 9,0 und der Ereignisweiterleitung (wie in allen Experimenten) 1,0. Bei den Rendezvousknoten fällt sie bei zwei Ereignistypen, steigt danach jedoch wieder an. Verantwortlich ist die Lage der Rendezvousknoten im Netz. Durch eine äußere Lage werden Profile etwas weiter verteilt und verursachen so mehr Filtereinträge in den Vermittlern. Dieses Verhalten entspricht den formulierten Erwartungen.

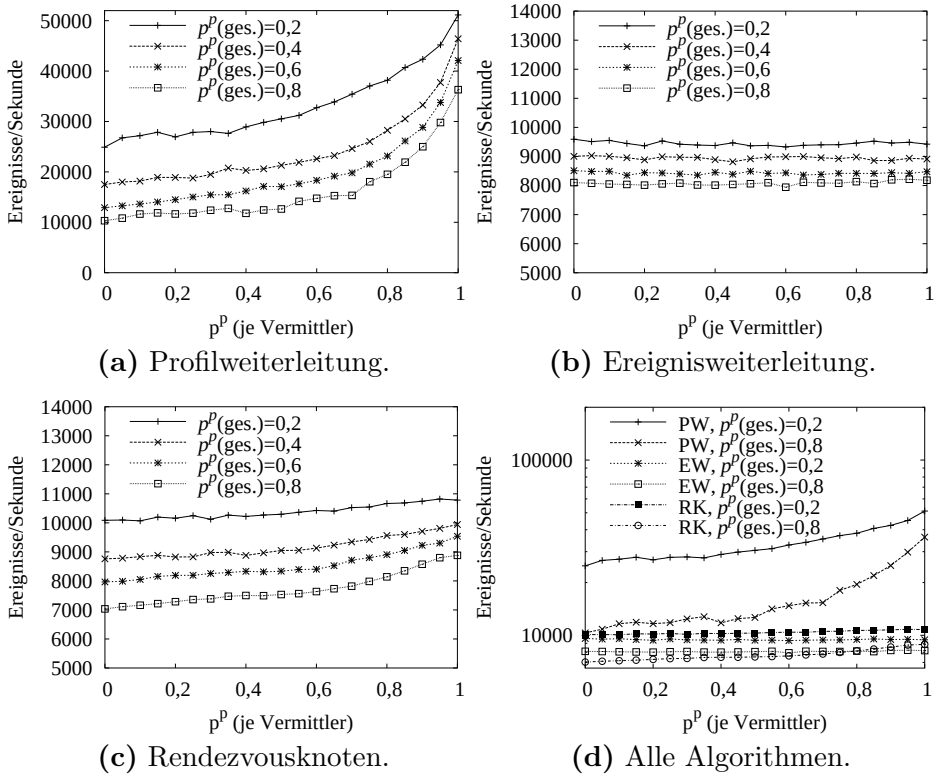
Analoges Verhalten zeigt die Netzlast beim Veröffentlichen (Abb. 6.12(b), logarithmische Einteilung der Ordinate). Ereignisweiterleitung und Profilweiterleitung zeigen relativ konstante Werte, jedoch in vertauschten Dimensionen (Profilweiterleitung wenig Last, Ereignisweiterleitung viel Last). Bei den Rendezvousknoten gilt die Analogie der Netzlast beim Veröffentlichen zum kurzen Abfallen der Profilduplikation und deren Ansteigen danach: Wie die Profile werden auch die Ereignisse zum Rendezvousknoten verteilt. Bei äußerer Lage der Rendezvousknoten werden mehr Vermittler zur Weiterleitung genutzt. Steigendes p^p erhöht die Netzlast bei Profilweiterleitung und Rendezvousknoten, wobei bei der Profilweiterleitung ein größerer Mehraufwand entsteht (keine ständigen „überflüssigen Weiterleitungen“, wie bei den Rendezvousknoten).

6.7 Einfluss der Lokalität

Die Lokalität der Profile und Ereignisse wird in diesem Abschnitt näher betrachtet. Lokalität bedeutet, dass die Ereignisse der lokalen Anbieter eines Vermittlers nur zu den Profilen lokaler Abonnenten passen. In den Experimenten werden vier Vermittler in der in Abschnitt 6.3 beschriebenen Weise als linearer Bus genutzt. 160.000 Gesamtprofile nutzen genau einen Ereignistyp. In den Experimenten wird der Anteil passender Profile bezüglich eines Vermittlers (p^p (je Vermittler), also die Lokalität) erhöht. Es werden Filtereffizienz und Netzlast beim Veröffentlichen untersucht. Die Profilduplikation wird nicht betrachtet, da in den Experimenten nur die Ereignisse verändert werden.

6.7.1 Hypothese

Die Filtereffizienz bei der Profilweiterleitung verbessert sich stark bei vermehrter Lokalität der Profile und Ereignisse (weniger Weiterleitungen von Ereignissen). Außerdem sollte bei den Rendezvousknoten ein Anstieg der Filtereffizienz, wenn auch geringer, zu verzeichnen sein: Die Verteilung der Ereignisse zum Rendezvousknoten muss in jedem Fall geschehen. So wird nur ein geringer Teil des Kommunikationsaufwands bei verstärktem Lokalitätsverhalten eingespart. Die Filtereffizienz der Ereignisweiterleitung sollte unabhängig von der Lokalität sein (ständige Filterung aller Ereignisse in



(a) Profilweiterleitung. **(b)** Ereignisweiterleitung.

(c) Rendezvousknoten. **(d)** Alle Algorithmen.

Abbildung 6.13: Filtereffizienz der verteilten Algorithmen abhängig von der Lokalität bei unterschiedlichem Anteil passender Profile p^p.

allen Vermittlern). Die Netzlast bei Profilweiterleitung und Rendezvousknoten vermindert sich bei erhöhter Lokalität (weniger Weiterleitungen von Ereignissen und Benachrichtigungen). Die Ereignisweiterleitung ist unbeeinflusst von der Lokalität (ständige Weiterleitung aller Ereignisse).

6.7.2 Messergebnisse und Auswertung

Abb. 6.13 zeigt die Filtereffizienz unter dem Einfluss der Lokalität. In der Abbildung sind die unterschiedlichen Maßstäbe und die logarithmische Einteilung der Ordinate in Abb. 6.13(d) zu beachten. Bei der Profilweiterleitung steigt die Filterfrequenz f_{sat} um das 2- bis 3,5-fache bei einer Veränderung der Lokalität von 0 auf 1. Die Frequenzen steigen von 10.000 bis 25.000 Ereignissen pro Sekunde auf 35.000 bis 50.000 Ereignisse pro Sekunde. Die Begründung ist das Verwerfen von Ereignissen ohne passende Profile bereits bei deren lokalen Vermittler.

Die Ereignisweiterleitung zeigt sich unbeeinflusst vom Lokalitätsverhalten. Es ergeben sich stabile Filterfrequenzen aufgrund der Verteilung aller Ereignisse im Vermittlungsnetz. Rendezvousknoten zeigen weniger Beeinflussung als die Profilweiterleitung: maximal 1,25-fache Effizienzsteigerung bei ausschließlicher Lokalität im Vergleich zu keiner Lokalität. Der Grund ist erneut die Weiterleitung der Ereignisse zum Rendezvousknoten unter allen Umständen.

Insgesamt zeigt die Profilweiterleitung weit bessere Anpassung an Lokalitätseigenschaften als die anderen beiden Algorithmen (Abb. 6.13(d)). Auch das Anpassungsverhalten der Rendezvousknoten ist im Vergleich sehr gering, da mindestens ein Teil des Vermittlungsnetzes (der Pfad zum Rendezvousknoten) von jedem Ereignis besucht wird. Der dafür notwendige Kommunikationsaufwand übersteigt fast den Vorteil des Filterns in weniger Vermittlungsknoten.

Die Netzlast beim Veröffentlichen zeigt Abb. 6.14. Die Ereignisweiterleitung wird von der Lokalität gar nicht beeinflusst (ständiges Fluten). Bei Rendezvousknoten und Profilweiterleitung fällt die Netzlast (Verwerfen von Ereignissen), allerdings bei den Rendezvousknoten weniger, da stets alle Ereignisse zum Rendezvousknoten geleitet werden müssen.

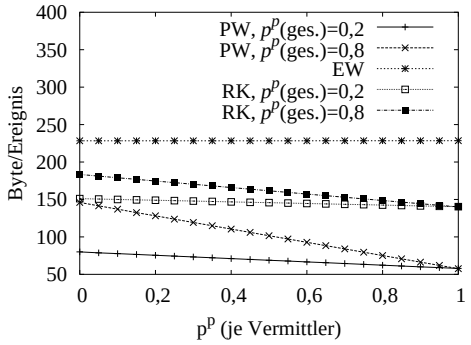


Abbildung 6.14: Netzlast der verteilten Algorithmen beim Veröffentlichen abhängig von der Lokalität.

Die Hypothesen bzgl. der Filtereffizienz wurden in den Experimenten erhärtet: Die

Profilweiterleitung zeigt die größte Effizienzsteigerung, gefolgt von den Rendezvousknoten. Die Ereignisweiterleitung wird vom Lokalitätsverhalten nicht beeinflusst. Das erwartete Verhalten der Netzlasten wurde ebenfalls bestätigt: Profilweiterleitung und Rendezvousknoten erfahren eine Verminderung, die Netzlast der Ereignisweiterleitung ist unabhängig vom Lokalitätsverhalten.

6.8 Einfluss der Profilanzahl

In diesem Abschnitt wird der Einfluss der Gesamtprofilanzahl betrachtet. Es dienen erneut vier Vermittler als linearer Bus. Das Vermittlungsnetz nutzt die in Abschnitt 6.3 dargestellte Struktur. Am System werden verschiedene Mengen eindeutiger Profile angemeldet ($\sigma = 1$). Es werden jeweils gleichartige Ereignisse zum System gesendet. Der Anteil erfüllender Ereignisse beträgt $p^e = 0,8$. Es wird lediglich die Filtereffizienz untersucht. Die Netzlast ist, aufgrund des Hinzufügens nicht passender Profile, stets gleich. Die Profilduplikation verhält sich analog zu den vorigen Experimenten.

Die Messung des Einflusses der Gesamtprofilzahl auf die Filtereffizienz gestaltet sich etwas komplizierter als die bisherigen Messungen. Um die wirkliche Speicherauslastung des Systems widerzuspiegeln, darf keine vorzeitige Auslagerung in den Hintergrundspeicher erfolgen. Bei den vorigen Experimenten wurden u.U. nacheinander verschiedene Ereigniskonfigurationen auf der gleichen Profilmenge gefiltert. Hierbei findet schon während der ersten Konfiguration eine Auslagerung der bei der Filterung nicht benötigten Daten statt, so dass mehr Profile in den Hauptspeicher passen, ohne dass Effizienzminderungen beobachtet werden. Um dieses zu verhindern, werden jetzt genau einmal Ereignisse auf der von den Profilen erzeugten Filterstruktur gefiltert. Eine Auslagerung nicht benötigter Daten wird damit unterbunden. Die Ergebnisse geben so die tatsächliche Speicherauslastung wider.

6.8.1 Hypothese

Mit steigender Profilzahl sollte sich die Filtereffizienz vermindern. Ab einer gewissen Gesamtmenge an Profilen bricht die Filtereffizienz stark ein. Hier ist der Hauptspeicher ausgenutzt, also muss ein Teil der Profile bzw. der Filterstruktur in den Hintergrundspeicher verdrängt werden. Bei der Profilweiterleitung und den Rendezvousknoten tritt dieser Verlagerungseffekt ziemlich schnell in Erscheinung. Bei der Ereignisweiterleitung sollte dies erst bei größeren Profilzahlen der Fall sein, da Profile nicht in Vermittlern dupliziert werden. Bei vier Vermittlern sollte der Effizienzeinbruch bei der Ereignisweiterleitung bei der ca. vierfachen Profilmenge auftreten.

6.8.2 Messergebnisse und Auswertung

Abb. 6.15 zeigt die Filtereffizienz der drei Algorithmen bei steigender Gesamtprofilzahl. Bis zu ca. 100.000 bzw. 350.000 Profile können von den Algorithmen im Hauptspeicher verwaltet werden. Hierbei ist zu beachten, dass es sich um eindeutige Profile handelt, also die Ereignisauslastung $\sigma = 1$ ist. Bei Vorhandensein von Überdeckungen (wie teilweise in den vorherigen Experimenten) können von der Profilweiterleitung und den Rendezvousknoten mehr Profile verwaltet werden: Die Vermittler müssen nicht alle Profile speichern und filtern, sondern lediglich Profile ohne überdeckende Profile.

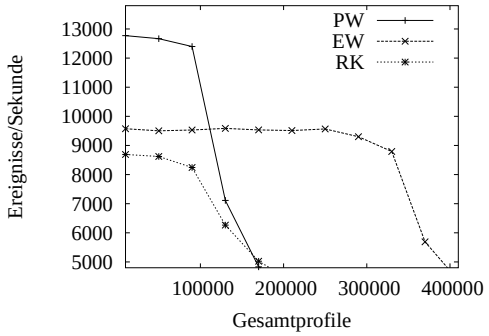


Abbildung 6.15: Filtereffizienz der verteilten Algorithmen abhängig von der Profilanzahl.

Bei der Ereignisweiterleitung existiert dieser Vorteil nicht, da sämtliche lokalen Profile gespeichert und gefiltert werden müssen (jedoch nicht verteilt).

Aufgrund des großen Anteils erfüllender Ereignisse mit $p^e = 0,8$ sind Rendezvousknoten ineffizienter als die Ereignisweiterleitung (Begründung siehe Abschnitt 6.3.2). Die Profilweiterleitung zeigt eindeutig die effizienteste Filterung, solange keine Seitenauslagerungen (ab ca. 100.000 Profilen) auftreten. Rendezvousknoten können ca. 8.500 Ereignisse pro Sekunde verarbeiten, die Ereignisweiterleitung 9.500 und die Profilweiterleitung 12.500 Ereignisse pro Sekunde.

Ab ca. 100.000 Profilen bricht die Filtereffizienz von Rendezvousknoten und Profilweiterleitung ein: Beim Rendezvousknoten ist der größte Speicheraufwand gerade in ihm selbst zu verzeichnen, bei der Profilweiterleitung in allen Vermittlern. Aufgrund des großen Hauptspeicherbedarfs entstehen Seitenauslagerungen, die einen Effizienzeinbruch zur Folge haben. Ab ca. 350.000 Profilen (etwas weniger als die vierfache Profilmenge entsprechend den Erwartungen) bricht auch die Filtereffizienz bei der Ereignisweiterleitung ein: In jedem Vermittler wird ein Viertel der Gesamtprofile gespeichert, Seitenauslagerungen sind dann auch hier die Folge.

6.9 Zusammenfassung

In diesem Kapitel wurden Experimente mit dem Prototyp DAS, deren Ergebnisse und Auswertungen dargestellt. Abschnitt 6.1 zeigte Schwächen und Mängel bisheriger Untersuchungen verteilter Benachrichtigungssysteme auf, die insbesondere auf der Variation zu weniger Systemparameter und der Betrachtung unzureichender Beurteilungskriterien beruhen. Der Aufbau, die Bewertungskriterien und die Annahmen der Experimente dieser Arbeit wurden in Abschnitt 6.2 beschrieben. Die Evaluation und deren Ergebnisse waren Bestandteil der Abschnitte 6.3 bis 6.8. Dabei wurden Variationen am Anteil passender Profile p^p, am Anteil erfüllender Ereignisse p^e, der Vermittlerzahl, den Überdeckungen, der Anzahl der Ereignistypen, dem Lokalitätsverhalten von Ereignissen und Profilen und der Gesamtprofilanzahl vorgenommen. Bezogen auf die gewählten Bewertungskriterien sind die Ergebnisse der durchgeführten Experimente die folgenden:

Filtereffizienz: Die Profilweiterleitung kann in den meisten Fällen die größten Ereignisfrequenzen f_{sat} verarbeiten (sofortiges Verwerfen von Ereignissen ohne passende Profile). Insbesondere bei geringem Anteil erfüllender Ereignisse oder passender Profile sind die von der Profilweiterleitung verarbeitbaren Ereignisfrequenzen f_{sat} weit höher als diejenigen bei Nutzung von Rendezvousknoten oder Ereignisweiterleitung (sofortiges Verwerfen und damit nur einmaliges Filtern fast aller Ereignisse). Bei hohem Anteil passender Profile nähern sich die filterbaren Ereignisfrequenzen f_{sat} der drei verteilten Filteralgorithmen stark aneinander an. Die Vorteile der Profilweiterleitung verschwinden, da hier auch jedes Ereignis im Netz verbreitet werden muss. Die Ereignisweiterleitung kann in sehr wenigen Fällen die höchsten Ereignisfrequenzen f_{sat} verarbeiten. Der Grund ist das einfache Filterprotokoll, so dass bei einem hohen Anteil passender Profile die Profilweiterleitung und die Rendezvousknoten mit ihren komplexeren Protokollen mehr Aufwand erzeugen. Rendezvousknoten erreichen meist Ereignisfrequenzen, die zwischen denen der Profil- und der Ereignisweiterleitung liegen. Ein Vorteil der Rendezvousknoten bei großer Anzahl an Ereignistypen ergibt sich nicht, da innere Netzknoten stets den Flaschenhals des verteilten Benachrichtigungssystems darstellen. Sie müssen alle Ereignisse weiterleiten und teilweise filtern, so dass der Vorteil einer Verteilung entfällt.

Netzlast pro Ereignis: Die Netzlast bei der Ereignisweiterleitung ist unabhängig von den meisten Systemparametern (eine Ausnahme ist die Größe des Vermittlungsnetzes), da stets alle Ereignisse geflutet werden. Die Profilweiterleitung zeigt stets die geringste Netzlast (alleinige Weiterleitung von Ereignissen mit passenden Profilen), Rendezvousknoten liegen zwischen den beiden anderen Filteralgorithmen (ständige Weiterleitung in einen Teil des Vermittlungsnetzes, nämlich zu den Rendezvousknoten). Bei einem hohen Anteil erfüllender Ereignisse oder passender Profile steigt die Netzlast bei der Profilweiterleitung und den Rendezvousknoten an, die Werte der Ereignisweiterleitung werden jedoch nie ganz erreicht (dazu müsste jedes Ereignis einen lokalen Abonnenten bei jedem Vermittler des Netzes besitzen).

Profilduplikation: Die Profilduplikation zeigt bei der Profilweiterleitung die größten Werte (jeder Vermittler kann die Profile filtern, die notwendig sind, um Ereignisse ohne passende Profile zu verwerfen). Insbesondere bei ausschließlicher Nutzung eindeutiger Profile ist die Profilduplikation und damit der Speicherbedarf der Profilweiterleitung sehr hoch (jedes Profil wird in jedem Vermittler gespeichert). Gleiches gilt, in etwas abgeschwächter Weise, für die Rendezvousknoten (jeder Vermittler auf dem Pfad von einem lokalen Vermittler V_x zum Rendezvousknoten speichert alle Profile lokaler Abonnenten von V_x). Sind jedoch Überdeckungen vorhanden, ist dieser Nachteil von Rendezvousknoten und Profilweiterleitung nicht vorhanden. Die Ereignisweiterleitung dupliziert keine Profile und kann dadurch bei der gleichen Vermittlerzahl die größte Profilmenge verarbeiten.

Aufgrund dieser Abhängigkeit der Filteralgorithmen von den Systemparametern sollte ein Benachrichtigungssystem unterschiedliche Filteralgorithmen unterstützen. Je nach Systemlast und -nutzung bzw. der aktuellen Anwendung kann das Benachrichtigungssystem, bei Unterstützung verschiedener Algorithmen, den jeweils optimalen

Filteralgorithmus zur Filterung auswählen. Existieren zu fast jedem Ereignis passende Profile, sollte die Ereignisweiterleitung genutzt werden. Das Protokoll ist sehr einfach und erzeugt wenig zusätzlichen Aufwand. Die Verteilung der Ereignisse im gesamten Netz erhöht den ohnehin nötigen Netzverkehr nur noch geringfügig. Die Ereignisweiterleitung sollte ebenfalls bei sehr großen Profilzahlen genutzt werden, da die Profilduplikation und damit die Speicherauslastung am geringsten ausfällt. Die Profilweiterleitung sollte in den meisten anderen Fällen genutzt werden (weniger Profile, kleiner Anteil erfüllender Ereignisse, Überdeckungen). Hierbei wird wenig Netzlast erzeugt und eine Filterung ist weit effizienter möglich als bei der Ereignisweiterleitung und den Rendezvousknoten. Rendezvousknoten haben sich hingegen unter keiner getesteten Systemkonfiguration als vorteilhaft erwiesen. Exemplarische Einsatzmöglichkeiten der drei unterschiedlichen verteilten Filteralgorithmen werden im Folgenden beschrieben.

Für die Gebäudesteuerung stellt die Profilweiterleitung das geeignetste Verfahren dar. Überdeckungen sind vorhanden, da im Allgemeinen beispielsweise nicht jede Heizung oder Jalousie exklusive Temperatur- oder Sonnensensoren nutzt. Auch ist ein starkes Lokalitätsverhalten zu beobachten, da z.B. Temperatursensoren im Kellergeschoss hauptsächlich für Heizungen im Keller relevant sind. Die Gesamtprofilanzahl in den Vermittlern steigt deshalb nicht in solche Dimensionen, dass sie von der Profilweiterleitung nicht verarbeitet werden kann.

Die Ereignisweiterleitung sollte in Anwendungen mit weit verteilten Abonnenten gleicher oder ähnlicher Interessen genutzt werden. Beispielsweise ist in einer digitalen Bibliothek über Veröffentlichungen aus dem Bereich der Informatik eine weltweite Abdeckung verschiedener Forschungsthemen gegeben. Eine Auslieferung der Ereignisse muss deshalb an alle Vermittler geschehen, so dass die Ereignisweiterleitung in diesem Fall keine Nachteile liefert.

Rendezvousknoten sind bei geschickter Verteilung der Rendezvousknoten bei Informations- bzw. Nachrichtendiensten im Internet effizient nutzbar. Sind beispielsweise die Ereignistypen bezüglich örtlicher Themen vergeben und die Rendezvousknoten ebenfalls entsprechend dieses Ortes im Vermittlungsnetz verteilt, ergibt sich kein zusätzlicher Aufwand aufgrund der Weiterleitung zum Rendezvousknoten: Lokale Nachrichten werden nur von lokalen Anbietern veröffentlicht und nur von lokalen Abonnenten angefragt. Sind alle Anbieter und Abonnenten mit dem Rendezvousknoten des Typs ihrer Ereignisse bzw. Profile verbunden, werden keine Ereignisse im Vermittlungsnetz weitergeleitet.

Kapitel 7

Zusammenfassung und Ausblick

Inhalt

7.1 Zusammenfassung

Das in Benachrichtigungssystemen eingesetzte Abonnentenprinzip steht im Gegensatz zu der heutzutage im Internet hauptsächlich angewandten Kommunikationsform, dem Anfrage-Antwort-Prinzip. Bei diesem Abonnentenprinzip wird eine einmalige Anfrage eines jeden Abonnenten, ein sog. Profil, an das Benachrichtigungssystem gesendet. Informationen der Anbieter, bezeichnet als Ereignisse, werden ebenfalls zum Benachrichtigungssystem geschickt. Dieses ordnet Ereignisse passenden Profilen zu und benachrichtigt die Abonnenten über alle Ereignisse, die ihre Profile erfüllen. Diese Vorgänge werden Filterung und Benachrichtigung genannt. Insbesondere bei Realzeitsystemen wie der Gebäudesteuerung ist die Effizienz der eingesetzten Filterkomponente eines der wichtigsten Entwurfskriterien.

Im Rahmen dieser Arbeit wird eine solche Filterkomponente entwickelt, analysiert und bewertet. Zur effizienten Bedienung vieler Informationsanbieter und Abonnenten mit einer Vielzahl von Profilen und Ereignissen wird diese Filterkomponente als verteiltes System realisiert.

Die Arbeit besteht aus einem theoretischen und einem praktischen Teil. Im theoretischen Teil werden vorhandene Ansätze der verteilten Filterung (Kapitel 2) und eine umfangreiche Klassifikation verteilter Filteralgorithmen vorgestellt (Kapitel 3). Im praktischen Teil werden der Entwurf (Kapitel 4) des verteilten Benachrichtigungsdienstes DAS und die Umsetzung mehrerer verteilter Filtervarianten beschrieben (Kapitel 5). Abschließend folgen umfangreiche Experimente und deren Auswertung (Kapitel 6). Die Kapitel lassen sich wie folgt zusammenfassen:

Kapitel 2: Verwandte Arbeiten. Kapitel 2 stellt bisherige Arbeiten aus dem Bereich der verteilten Filterung vor. Es sind verschiedene Kategorien von Filteralgorithmen zu finden: Rendezvousknoten, verteilte Filterung in hierarchischen Netzen

und verteilte Filterung in Punkt-zu-Punkt-Netzen. Weiterhin existieren in der Literatur verschiedene Optimierungsstrategien zur Verminderung von Profilredundanzen: Äquivalenz, Bedecken und Verschmelzen. Die vorhandenen Filteralgorithmen verfolgen teils gleichartige und teils voneinander abweichende Ansätze. Zur Verbesserung der Filtereffizienz sollten die Konzepte dieser Ansätze deshalb kombiniert und so optimal ausgenutzt werden. Für eine Kombination und einen Vergleich der Algorithmen müssen diese allerdings umfangreich klassifiziert werden.

Kapitel 3: Klassifikation verteilter Filteralgorithmen. Eine formelle Klassifikation verteilter Filteralgorithmen wird in Kapitel 3 beschrieben. Zur Charakterisierung dieser Algorithmen werden vier Kategorien eingeführt: Kommunikation mit den Abonnenten, Aufteilung des Filteraufwandes, Ort der Filterung und Speicherstrategie. Danach werden die verschiedenen Umsetzungsmöglichkeiten der eingeführten Kategorien kombiniert. Dabei ergeben sich 17 verschiedene Varianten zur Implementierung der verteilten Filteralgorithmen. Diese Varianten der Algorithmen werden dann einer Bewertung unterzogen. Als Kriterien dienen die erzeugte Netzlast, der benötigte Speicher, die Effizienz und die Skalierbarkeit. Es lassen sich drei Hauptarten von Filteralgorithmen erkennen. Daraus wird jeweils die in Hinblick auf die vier Bewertungskriterien beste Algorithmenvariante für eine praktische Untersuchung gewählt.

Kapitel 4: Systementwurf. Der Systementwurf als Teil der praktischen Umsetzung der verteilten Filterung in DAS wird in Kapitel 4 vorgestellt. Es beschreibt die zugrunde liegende Filterkomponente der genutzten Implementierung PrimAS [3] eines zentralisierten Benachrichtigungssystems und deren Erweiterungen und Anpassungen. Dadurch wird eine Effizienzsteigerung und eine Verminderung des notwendigen Speicherbedarfs erreicht. Weiterhin werden die Profildefinitionssprache und Aspekte des eingesetzten Netzwerks aufgezeigt und beschrieben. Die danach entwickelte Systemarchitektur von DAS enthält drei Hauptkomponenten: Vermittler, Anbieter und Abonnenten.

Kapitel 5: Verteilte Filterprotokolle. Kapitel 5 beschreibt die Umsetzung der Protokolle der drei in Kapitel 3 ausgewählten Filteralgorithmen (Profilweiterleitung, Ereignisweiterleitung und Rendezvousknoten) in DAS. Damit wird eine Lücke in bisherigen Arbeiten geschlossen, die stets nur unvollständige Beschreibungen der Protokolle der Algorithmen liefern. Darüber hinaus werden zwei neue Verfahren zur Berechnung der Optimierungsstrategien Bedecken und Verschmelzen vorgeschlagen: bereichsbasierte und profilbasierte Berechnung.

Kapitel 6: Experimente und Auswertung. Die detaillierte praktische Analyse und Auswertung von DAS wird in Kapitel 6 vorgestellt. Im Kontrast zu zahlreichen anderen Arbeiten werden keine Simulationen genutzt, die Evaluation wird direkt mit dem Prototyp DAS durchgeführt. Ebenfalls im Unterschied zu bisherigen Evaluationen wird der Einfluss zahlreicher Systemparameter auf den Speicherbedarf, die erzeugte Netzlast und die Filtereffizienz der Algorithmen betrachtet. Die untersuchten Systemparameter sind: Anteil passender Profile, Anteil erfüllender Ereignisse, Vermittlerzahl, Überdeckungen, Anzahl der Ereignistypen, Lokalitätsverhalten und Gesamtprofilanzahl. Die folgenden Ergebnisse werden erzielt:

1. Die Profilweiterleitung zeigt in vielen Untersuchungen die beste Filtereffizienz und Netzlast, jedoch einen großen Speicherbedarf.

2. Die Ereignisweiterleitung zeigt bei hohem Anteil erfüllender Ereignisse eine bessere Filtereffizienz (einfaches Protokoll) als die anderen Algorithmen und bei hoher Profilanzahl eine bessere Skalierbarkeit (keine Profilredundanzen). Die Netzlast ist stets sehr hoch, der Speicherbedarf optimal.

3. Die Rendezvousknoten können unter keiner der untersuchten Systemkonfigurationen bessere Ergebnisse als andere Verfahren erreichen.

Deshalb sollte ein Benachrichtigungssystem unterschiedliche Filteralgorithmen unterstützen und je nach Systemlast und -nutzung bzw. der aktuellen Anwendung den verwendeten Filteralgorithmus auswählen. Im Anwendungsgebiet der Gebäudeverwaltung erreicht die Profilweiterleitung aufgrund der dort typischen Profildefinitionen die besten Ergebnisse.

7.2 Ausblick

Im Bereich der verteilten Filterung sind einige Aspekte bisher wenig bzw. nicht zufrieden stellend geklärt und bieten daher Raum für weitere Forschung. Im Folgenden werden drei mögliche Erweiterungen des Systems DAS skizziert.

Zusammengesetzte Profile. Zusammengesetzte Profile sind eine Erweiterung der im Rahmen dieser Arbeit beschriebenen Profile. Sie sind eine Komposition von Profilen mit diversen Operatoren wie Konjunktion, Disjunktion und Sequenz. Die Filterung zusammengesetzter Profile in einer verteilten Umgebung ist ein bisher wenig untersuchtes Forschungsgebiet (in [5] existiert lediglich ein Ansatz für die Sequenz).

Um die Einsetzbarkeit der verteilten Implementierung DAS für die Filterung zusammengesetzter Profile zu zeigen, wurde in dieser Arbeit bereits eine einfache Variante zur Filterung realisiert: Die lokalen Vermittler der Abonnenten mit zusammengesetzten Profilen sind für die Filterung der Kompositionen zuständig, aufbauend auf der Implementierung CompAS von König [17]. Teilprofile werden entsprechend der in DAS gewählten Verteilungsstrategie behandelt, also z.B. zum Rendezvousknoten ihres Typs gesendet. Bei Erfüllung von Teilprofilen in einem lokalen Vermittler wird die zentralisierte Implementierung des Systems CompAS angestoßen. Bei der Erfüllung von zusammengesetzten Profilen wird eine Benachrichtigung erstellt und an die Abonnenten weitergeleitet.

Diese Integration von zusammengesetzten Profilen ist ein erster Ansatz zur Umsetzung der verteilten Filterung. Es besteht jedoch noch weiterer Forschungsbedarf, um die Filterung geschickter zu verteilen und zu optimieren. So ist fraglich, an welcher Stelle im Vermittlungsnetz zusammengesetzte Profile gefiltert werden sollen. Vielleicht existieren auch Möglichkeiten zum Verschmelzen von zusammengesetzten Profilen? Oder können Überdeckungen genutzt werden? Hinze führt in [10] zahlreiche Parameter im Zusammenhang mit zusammengesetzten Profilen und eine neue Methode (Ein-Schritt-Verfahren) zur effizienten Filterung in zentralisierten Systemen ein. Auch hier ist fraglich, wie diese Parameter in einer verteilten Umgebung umzusetzen und ob das Ein-Schritt-Verfahren auf die verteilte Filterung angepasst werden kann.

Hierarchische Benachrichtigungsdienste. Aktuell können verschiedene Implementierungen von Benachrichtigungssystemen nicht miteinander arbeiten und sich gegenseitig nutzen. So müssen Abonnenten bei jedem System mit potentiellen Informationen Profile anlegen. Diese Profile müssen in jeweils anderen Profildefinitionssprachen definiert werden. Zusätzlich benutzen verschiedene Sprachen unterschiedliche Semantiken für gleiche Operatoren [16].

Jung [16] hat gezeigt, wie eine mächtige Ereignis-Algebra als gemeinsame Basis für verschiedene heute eingesetzte Profildefinitionssprachen genutzt werden kann. Durch die Nutzung dieser Ereignis-Algebra bzw. einer auf ihr aufbauenden Profildefinitionssprache können Profile durch einmalige Definition angelegt und, unter Berücksichtigung der Semantik der Operatoren, in verschiedenen Systemen genutzt werden. Zudem werden in [16] Gruppen für die Profildefinitionssprachen der derzeit populärsten und bekanntesten Benachrichtigungssysteme gebildet sowie Transformationen dieser Sprachen ineinander entwickelt. Die entworfenen Transformationen sind sog. äquivalente, positive, negative oder transferierende Transformationen. Durch einen Einsatz dieser Transformationen stehen die Informationen mehrerer Systeme gleichzeitig und zusammengefasst in einem Benachrichtigungssystem zur Verfügung. So entsteht eine Verbindung und gegenseitige Nutzung verschiedener Systeme, welche als hierarchische Benachrichtigungsdienste bezeichnet werden.

Die Integration der o.g. Ideen in DAS wirft weitere Fragen auf: Es müssen sowohl die Profildefinitionssprachen zahlreicher Benachrichtigungssysteme implementiert als auch Kommunikationsmöglichkeiten mit diesen Systemen geschaffen werden. Weiterhin müssen Benachrichtigungen eines Systems als Ereignisse eines anderen Dienstes auftreten. Dabei ist insbesondere fraglich, wie dies effizient geschehen kann, z.B. wenn Teilereignisse zusammengesetzter Ereignisse in verschiedenen Systemen auftreten. Eine Analyse, wann positive bzw. transferierende Transformationen genutzt werden sollten, muss aus Effizienzgründen ebenfalls vorgenommen werden.

Mobilität der Klienten. Ein im Kontext von Benachrichtigungsdiensten relativ neues Forschungsgebiet ist die Unterstützung mobiler Klienten. Die Vermittler des Benachrichtigungssystems agieren dabei ähnlich den Mobilen Service Stationen (MSS) in einem Mobilfunknetz zur Abdeckung räumlicher Gebiete. Klienten im Bereich eines Vermittlers wählen diesen als lokalen Vermittler. Von Podnar und anderen [29] wurden verschiedenen Arten von Mobilität im Kontext von Benachrichtigungssystemen erarbeitet. Sie unterscheiden sich im Vermittlerwechsel, also der Wahl eines neuen lokalen Vermittlers, aufgrund eines Ortswechsels. Diese Vermittlerwechsel stellen gemäß Huang und Garcia-Molina [12] das Hauptproblem bei der Nutzung mobiler Klienten dar.

Bei einem Vermittlerwechsel dürfen z.B. keine Benachrichtigungen verloren gehen oder mehrfach ausgeliefert werden. Weiterhin muss ein u.U. nötiger Umbau der Filterstruktur effizient geschehen. Fraglich ist, ob alle Benachrichtigungen ausgeliefert werden sollen oder lediglich die aktuellen, insbesondere bei längeren Zeiten der Verbindungsunterbrechung zwischen Abonnenten und Vermittlern. Ebenfalls ist eine Vergabe von Prioritäten denkbar, so dass unwichtige Benachrichtigungen verworfen werden.

Die Unterstützung von Mobilität in Implementierungen von Benachrichtigungssystemen ist bisher kaum vorhanden. Auch sind Vergleiche und Analysen der oben

erwähnen Unterscheidungen und anderer Mobilitätsstrategien in der vorhandenen Literatur nicht zu finden.

Es bieten sich also weitere Fragestellungen im Rahmen von Benachrichtigungssystemen, die in zukünftigen Forschungsvorhaben untersucht und gelöst werden sollten.

Literaturverzeichnis

[1] Aguilera, M. K., R. E. Strom, D. C. Sturman, M. Astley und T. D. Chandra: *Matching Events in a Content-Based Subscription System.* In: *Proceedings of the 18th ACM Symposium on Principles of Distributed Computing (PODC '99)*, Seiten 53–61, Atlanta, USA, 4.–6. Mai 1999.

[2] Banavar, G., T. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom und D. C. Sturman: *An Efficient Multicast Protocol for Content-based Publish-Subscribe Systems.* In: *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems (ICDCS '99)*, Seiten 262–272, Austin, USA, 31. Mai–4. Juni 1999.

[3] Bittner, S.: *Implementierung eines effizienten Matchingverfahrens für Benachrichtigungssysteme.* Studienarbeit, Freie Universität Berlin, Institut für Informatik, September 2002.

[4] Brin, S. und L. Page: *The Anatomy of a Large-Scale Hypertextual Web Search Engine.* Computer Networks and ISDN Systems, 30(1–7):107–117, 1998.

[5] Carzaniga, A., D. S. Rosenblum und A. L. Wolf: *Interfaces and Algorithms for a Wide-Area Event Notification Service.* Technischer Bericht CU-CS-888-99, Fachbereich Informatik, Universität Colorado, Oktober 1999. Überarbeitet Mai 2000.

[6] European Installation Bus (EIB): *Forum.* Verfügbar unter `http://www.eib-forum.de/`, Stand 20. August 2003.

[7] Faensen, D., L. Faulstich, H. Schweppe, A. Hinze und A. Steidinger: *Hermes – A Notification Service for Digital Libraries.* In: *Proceedings of the ACM/IEEE Joint Conference on Digital Libaries (JCDL '01)*, Seiten 373–380, Roanoke, USA, 24.–28. Juni 2001.

[8] Garey, M. R. und D. S. Johnson: *Computers and Intractability – A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[9] Gough, J. und G. Smith: *Efficient Recognition of Events in a Distributed System.* In: *Proceedings of the 18th Australasian Computer Science Conference (ACSC-18)*, Adelaide, Australien, 1.–3. Februar 1995.

[10] Hinze, A.: *A-MEDIAS: Concept and Design of an Adaptive Integrating Event Notification Service.* Doktorarbeit, Freie Universität Berlin, Institut für Informatik, Juli 2003.

[11] Hinze, A. und S. Bittner: *Efficient Distribution-based Event Filtering.* In: *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW '02)*, Seiten 525–532, Wien, Österreich, 2.–5. Juli 2002.

[12] Huang, Y. und H. Garcia-Molina: *Publish/Subscribe in a Mobile Environment.* In: *Proceedings of the 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE '01)*, Seiten 27–34, Santa Barbara, USA, 20. Mai 2001.

[13] Institute of Electrical and Electronic Engineers (IEEE): *Local Area Network Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, Juni 1983. American National Standard ANSI/IEEE 802.3.

[14] Institute of Electrical and Electronic Engineers (IEEE): *Local Area Network Token Ring Access Method and Physical Layer Specifications*, 1985. American National Standard ANSI/IEEE 802.5.

[15] Jacobsen, H. A., G. Ashayer und H. Leung: *Predicate Matching and Subscription Matching in Publish/Subscribe Systems.* In: *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW '02)*, Seiten 539–548, Wien, Österreich, 2.–5. Juli 2002.

[16] Jung, D.: *Benachrichtigungsdienste: Analyse und Transformation ausgewählter Profildefinitionssprachen.* Staatsexamensarbeit, Freie Universität Berlin, Institut für Informatik, September 2002.

[17] König, S.: *Implementierung und Untersuchung eines parametergesteuerten Benachrichtigungssystems für kombinierte Events.* Diplomarbeit, Freie Universität Berlin, Institut für Informatik, 2003. In Arbeit.

[18] Lichtvision: *Persönliche Kommunikation mit Vertretern der Firma Lichtvision GmbH.* Homepage verfügbar unter `http://www.lichtvision.de/`.

[19] Liebig, C., B. Boesling und A. Buchmann: *A Notification Service for Next-Generation IT Systems in Air Traffic Control.* In: *Proceedings of the GI-Workshop: Multicast-Protokolle und Anwendungen*, Seiten 55–68, Braunschweig, Deutschland, 19.–21. Mai 1999.

[20] Liu, L., C. Pu, W. Tang und W. Han: *CONQUER: A Continual Query System for Update Monitoring in the WWW.* International Journal of Computer Systems, Science and Engineering, Special Issue on Web Semantics, 14(2):99–112, 1999.

[21] LNO: *LON User Organisation, Homepage.* Verfügbar unter `http://www.lno.de/`, Stand 20. August 2003.

[22] Luxmate Controls: *Homepage.* Verfügbar unter `http://www.luxmate.com/`, Stand 20. August 2003.

[23] Mühl, G.: *Generic Constraints for Content-Based Publish/Subscribe Systems.* In: *Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS '01)*, Seiten 211–225, Trento, Italien, 5.–7. September 2001.

[24] Mühl, G.: *Large-Scale Content-Based Publish/Subscribe Systems*. Doktorarbeit, Technische Universität Darmstadt, September 2002.

[25] Mühl, G. und L. Fiege: *Supporting Covering and Merging in Content-Based Publish/Subscribe Systems: Beyond Name/Value Pairs*. IEEE Distributed Systems Online (DSOnline), 2(7), 2001.

[26] Mühl, G., L. Fiege und A. Buchmann: *Filter Similarities in Content-Based Publish/Subscribe Systems*. In: *Proceedings of the International Conference on Architecture of Computing Systems (ARCS '02)*, Seiten 224–238, Karlsruhe, Deutschland, 8.–12. April 2002.

[27] Pietzuch, P. und J. Bacon: *Hermes: A Distributed Event-Based Middleware Architecture*. In: *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW '02)*, Seiten 611–618, Wien, Österreich, 2.–5. Juli 2002.

[28] Pietzuch, P. und J. Bacon: *Peer-to-Peer Overlay Broker Networks in an Event-Based Middleware*. In: *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS '03)*, San Diego, USA, 8. Juni 2003.

[29] Podnar, I., M. Hauswirth und M. Jazayeri: *Mobile Push: Delivering Content to Mobile Users*. In: *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW '02)*, Seiten 563–570, Wien, Österreich, 2.–5. Juli 2002.

[30] Rowstron, A. I. T., A.-M. Kermarrec, M. Castro und P. Druschel: *SCRIBE: The Design of a Large-Scale Event Notification Infrastructure*. In: *Proceedings of the 3rd International Workshop on Networked Group Communications (NGC 2001)*, Seiten 30–43, London, Großbritanien, 7.–9. November 2001.

[31] Schweppe, H., A. Hinze und D. Faensen: *Database Systems as Middleware – Events, Notifications, Messages*. In: *Proceedings of 2000 ADBIS-DASFAA Symposium Symposium on Advances in Databases and Information Systems*, Seiten 21–22, Prag, Tschechische Republik, 5.–8., September 2000.

[32] The Internet Engineering Task Force (IETF): *Internet Protocol*, September 1981. Herausgegeben von J. Postel. RFC 791.

[33] The Internet Engineering Task Force (IETF): *Transmission Control Protocol*, September 1981. Herausgegeben von J. Postel. RFC 793.

[34] The Internet Engineering Task Force (IETF): *Network News Transfer Protocol*, Februar 1986. Herausgegeben von B. Kantor und P. Lapsley. RFC 977.

[35] The Internet Engineering Task Force (IETF): *Internet Protocol, Version 6 (IPv6)*, Dezember 1998. Herausgegeben von S. Deering und R.Hinden. RFC 2460.

[36] The Internet Engineering Task Force (IETF): *Hypertext Transfer Protocol – HTTP*, Juni 1999. Herausgegeben von R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach und T. Berners-Lee. RFC 2616.

[37] Wu, B. und K. Dube: *PLAN: A Framework and Specification Language with an Event-Condition-Action (ECA) Mechanism for Clinical Test Request Protocols.* In: *Proceedings of the 34th Hawaii International Conference on System Science (HICSS-34)*, Maui, USA, 3.–6. Januar 2001.

[38] Yu, H., D. Estrin und R. Govindan: *A Hierarchical Proxy Architecture for Internet-scale Event Services.* In: *Proceedings of IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '99)*, Seiten 78–83, Stanford, USA, 16.–18. Juni 1999.

Anhang A

Signaturen ausgewählter Klassen

```
public class DistributedSubscriber extends Subscriber implements Runnable {

  //Listener - Warten und Behandeln von Nachrichten
  protected void listen();
  protected boolean handleMessages( String strService, String strHost,
                    String strPort, String strMessageBody);

  //Nachricht an Vermittler senden
  public void sendMessage( String strMessage);

  //Profile an-/abmelden
  protected boolean subscribeProfile( Profile p);
  protected void unsubscribeProfile(String strID);

  //Verbindung zum Server: herstellen/beenden/warten
  public void establishConnection();
  public synchronized boolean waitForConnection();
  public void closeConnection();
  ...}
```

Code Ausschnitt A.1: Distributed Subscriber Klassensignatur - diese Klasse repräsentiert einen Abonnenten.

```
public class DistributedPublisher extends Publisher implements Runnable {

  //Listener - Warten und Behandeln von Nachrichten
  protected void listen();
  protected boolean handleMessages( String strService, String strHost,
                    String strPort, String strMessageBody);

  //Nachricht an Vermittler senden
  public void sendMessage( String strMessage);

  //Ereignisse veroeffentlichen
  public boolean publishEvents( Event[] arrEvents);

  //Verbindung zum Server: herstellen/beenden/warten
  public void establishConnection();
  public synchronized boolean waitForConnection();
  public void closeConnection();
  ...}
```

Code Ausschnitt A.2: Distributed Publisher Klassensignatur - diese Klasse repräsentiert einen Anbieter.

```
public abstract class UniversalTree {

  //Hinzufuegen/Entfernen von Profilen
  public abstract void addProfiles ( Profile[] arrProfiles);
  public abstract void removeProfiles ( Profile[] arrProfiles);

  //Verteilungsabhaengige Umordnung der Filterstruktur
  public abstract void optimizeByEvents( Event[] arrEvents);
  public abstract void optimizeByProfiles();

  //Berechnung der Bedeckungen
  public abstract Profile[] getCoveredProfiles( Profile p);
  public abstract Profile[] getCoveringProfiles( Profile p);

  //Filtern eines Ereignisses und Ausfuehren der Benachrichtigungen
  public int postEvents( Event[] arrEvent);
  ...}
```

Code Ausschnitt A.3: Filterbaum Klassensignatur - diese Klasse verwaltet die Filterstruktur.

```
public abstract class Neighbour {

  //Verbindungsinformationen
  public String getHost();
  public int getPort();
  public Socket getSocket();

  //Senden einer Nachricht
  public void send( String message);

  //Spezielle Operationen beim Beenden der Verbindung
  public abstract void remove();
  public abstract void whenClosing();
  ...}
```

Code Ausschnitt A.4: Neighbour Klassensignatur - diese Klasse repräsentiert einen Nachbarn im Vermittlungsnetz.

```
public abstract class NeighbourBroker extends Neighbour {

  //Spezielle Operationen beim Beenden der Verbindung
  public void remove();
  public void whenClosing();

  //Aufbauen einer Verbindung
  protected boolean connectToServerSocket();
  ...}
```

Code Ausschnitt A.5: NeighbourBroker Klassensignatur - diese Klasse repräsentiert einen Nachbarvermittler im Vermittlungsnetz.

```
public class Broker implements Runnable {

  //Senden an einen Nachbarn
  public Neighbour sendToNeighbour( Neighbour neigh, String msg);

  //Senden an alle Nachbarn eines Typs, ausser ...
  public void sendToNeighbourBrokers( Neighbour neigh, String msg);
  public void sendToNeighbourSubscribers( Neighbour neigh, String msg);
  public void sendToNeighbourPublishers( Neighbour neigh, String msg);

  //Beenden der Verbindung zu einem Nachbarn
  public Neighbour disconnectNeighbour( Neighbour neigh);

  //Warten auf Verbindungsanfragen
  public void run();
  ...}
```

Code Ausschnitt A.6: Broker Klassensignatur - diese Klasse repräsentiert einen Vermittlungsserver.