Bedarfsgesteuerte n:m-Verteilung von Inhaltsobjekten in Rich Media Collaboration Applications

Daniel Schuster und Alexander Schill

TU Dresden, Fakultät Informatik, Lehrstuhl Rechnernetze, 01062 Dresden
Daniel.Schuster@tu-dresden.de, Alexander.Schill@tu-dresden.de

Abstract: Die effiziente Verteilung von Inhaltsobjekten wie Präsentationsfolien oder Bildern ist ein Problem, das vor allem in Rich Media Collaboration Applications, also Systemen mit kombinierter Videokonferenz- und Groupware-Funktionalität von Bedeutung ist. In diesem Artikel definieren wir eine Taxonomie für solche Verteilungsmethoden und stellen mit Remote Objects Container eine neue Methode vor, die die existierenden Methoden um die Möglichkeit bedarfsgesteuerter Verteilung bei symmetrischem Nutzungsverhalten (n:m) ergänzt.

1 Einleitung

Ein grundlegendes Problem bei der Entwicklung synchroner Groupware ist die Konsistenzsicherung oder Synchronisierung der gemeinsam betrachteten oder bearbeiteten Objekte. Der Synchronisierungszeitpunkt ist dabei entweder der Zutritt zu einer Groupware Session, die Änderung eines Objekts oder die Änderung einer gemeinsamen Sicht auf eines oder mehrere Objekte. Da in letzter Zeit Funktionalität synchroner Groupware zunehmend in Verbindung mit Conferencing genutzt wird (z.B. Videokonferenz-Integration in WebEx [1]), ergeben sich für diese Objektsynchronisierung neue Anforderungen.

Mit der Zusammenführung von IP Conferencing und synchroner Groupware innerhalb einer Rich Media Collaboration Application [2] prallen zwei grundlegende Paradigmen zur Objektverteilung aufeinander: Während in Konferenzsystemen das n:m-Schema mit gleichberechtigten Teilnehmern vorherrscht, orientieren sich Kooperationssysteme für Shared Viewing vorrangig am 1:n-Schema (1 Moderator, n Zuhörer). Das 1:n-Schema bildet für klassische Präsentationen und eLearning nach wie vor das soziale Protokoll am Besten nach. Durch neue Anwendungsszenarien wie z.B. regelmäßige Online-Meetings von geografisch verteilten Projektteams gewinnt das n:m-Schema jedoch immer mehr an Bedeutung.

In einem solchen Anwendungsszenario, in dem Teilnehmer wahlfrei auf Inhaltsobjekte wie Präsentationsfolien oder Dokumentseiten zugreifen können und der Gesamtumfang aller verfügbaren Objekte sehr groß wird (z.B. mehrere Projektbezogene Dokumente bei jedem Teilnehmer), werden Methoden zum bedarfsgesteuerten Zugriff auf Inhaltsobjekte benötigt, um Requests in für den Nutzer akzeptabler Zeit beantworten zu können.

Wir zeigen anhand einer Analyse verwandter Arbeiten zu Objektsynchronisierung in synchroner Groupware (Abschnitt 2), dass von vier möglichen Verteilungsschemen

die bedarfsgesteuerte n:m-Verteilung (n:m pull) bislang von noch keinem System unterstützt wird. In Abschnitt 3 wird mit Remote Objects Container ein Lösungsansatz für n:m pull, der auf die Idee des Personal Web Serving [3] aufbaut, präsentiert. Zur Validierung wurde der Ansatz von uns prototypisch implementiert und testweise in das Videokonferenzsystem VidConference integriert, um die Nutzbarkeit unserer Lösung innerhalb von Rich Media Collaboration Applications nachzuweisen (Abschnitt 4). In Abschnitt 5 fassen wir das Paper zusammen und diskutieren Möglichkeiten und Grenzen der Lösung.

2 Objektsynchronisierung in synchroner Groupware

Wir definieren eine Taxonomie für Methoden zur Synchronisierung gemeinsamer Objekte in synchroner Groupware, die nach Nutzungsmodell der Groupware und Art der Synchronisierung differenziert. Wir unterscheiden zunächst nach Nutzungsmodell in die klassische asymmetrische Nutzung mit einem Moderator und n Zuhörern (*1:n*) und das symmetrische Nutzungsmodell mit gleichberechtigten Teilnehmern (*n:m*).

Darüber hinaus unterscheiden wir nach Holmer [4] die beiden grundlegenden Synchronisationsarten der Vollreplikation und Teilreplikation, also der vollständigen oder bedarfsgesteuerten Konsistenzsicherung gemeinsamer Objekte. Wir bezeichnen die Technik der vollständigen Synchronisierung als *push*-Ansatz, während die bedarfsgesteuerte Technik als *pull* benannt wird. Damit ergeben sich vier mögliche Schemen zur Synchronisierung gemeinsamer Objekte in synchroner Groupware: 1:n push, 1:n pull, n:m push und n:m pull.

Klassische synchrone Groupware-Toolkits wie GroupKit [5] oder das Java Shared Data Toolkit [6] offerieren einen 1:n-Push-Dienst auf Basis eines Dienstprimitivs, das eine Message beliebigen Inhalts zuverlässig an alle Teilnehmer schicken kann. Diesen Verfahren gemeinsam ist, dass sie zwar vergleichsweise einfach implementierbar sind, jedoch sehr viel Bandbreite benötigen. Ein Einsatz ist bei häufiger Änderung und/oder großem Umfang der gemeinsamen Objekte nicht effizient.

Bei der Methode des 1:n pull wird zwar noch das klassische 1:n-Schema zugrunde gelegt, allerdings erfolgt die Synchronisierung gemeinsamer Objekte bedarfsgesteuert. Begole et. al. [7] entwickelten für das Problem des Zugriffs auf externe Dateien und andere Ressourcen (*Externalities*) in synchronen Kooperationssystemen den Ansatz der Proxied Externality Replication. Zugriffe auf Externalities werden lokal an einen Externality Proxy gestellt, der diese an den Externality Server weiterleitet, wo sie auf dem System des Presenters ausgeführt werden. Die Proxied Externality Replication arbeitet ähnlich unserem Ansatz mit URI-ähnlichen Pfaden zum Zugriff auf Objekte. Es wird jedoch nur ein Master in der Session unterstützt, der die zu synchronisierenden Objekte explizit vorgibt. Deshalb ist dieser Ansatz nicht für n:m-Synchronisierung geeignet.

Die Methode des n:m Push wird als Verteilungsmodell von MVC-basierter synchroner Groupware wie z.B. RendezVous [8], Clock [9] oder DistView [10] unterstützt. MVC steht dabei für das bekannte Entwurfsmuster Model-View-Controller. Die damit realisierten Anwendungen reichen von verteilten Projektmanagement-Tools bis hin zu n:m-Application-Sharing. Diese Systeme erlauben zwar im Gegensatz zu den bislang vorgestellten Arbeiten ein symmetrisches Nutzungsverhalten

(n:m), führen jedoch eine vollständige Synchronisierung der gemeinsamen Objekte durch. Sie sind deshalb ineffizient, wenn jeder Nutzer nur einen bestimmten Teil der gemeinsamen Objekte betrachten möchte.

3 N:m pull - Remote Objects Container

Eine bedarfsgesteuerte n:m-Verteilung von Inhaltsobjekten wird heute zwar bereits in Peer-to-Peer-Systemen genutzt, bislang jedoch noch nicht in Echtzeit-Systemen wie synchroner Groupware eingesetzt. Diese Art der Synchronisierung wird jedoch benötigt, wenn in Rich Media Collaboration Applications wahlfrei auf Inhaltsobjekte wie Präsentationsfolien oder Dokumentseiten zugegriffen werden soll, die von beliebigen anderen Teilnehmern in der Session zur Verfügung gestellt werden können.

Abbildung 1 zeigt unseren Ansatz für bedarfsgesteuerte n:m-Verteilung in synchroner Groupware, die Technik der *Remote Objects Container*. Er greift den aus Peer-to-Peer-Systemen bekannten Gedanken der nutzerseitigen Bereitstellung von Dateien auf und erweitert diesen um für synchrone Groupware wichtige Mechanismen wie semantische Zerlegung, Zuordnung zu einem Shared Workspace, Caching, Prefetching und Priorisierung. Jeder Teilnehmer der Groupware Session hält einen Container mit Adressen und Meta-Informationen entfernter Objekte, die von anderen Teilnehmern in die Groupware Session eingebracht werden. Beim Einfügen einer Menge von Objekten werden nur die URIs und Meta-Informationen der Objekte an alle Teilnehmer mittels eines Subscribe/Notify-Dienstes verteilt.

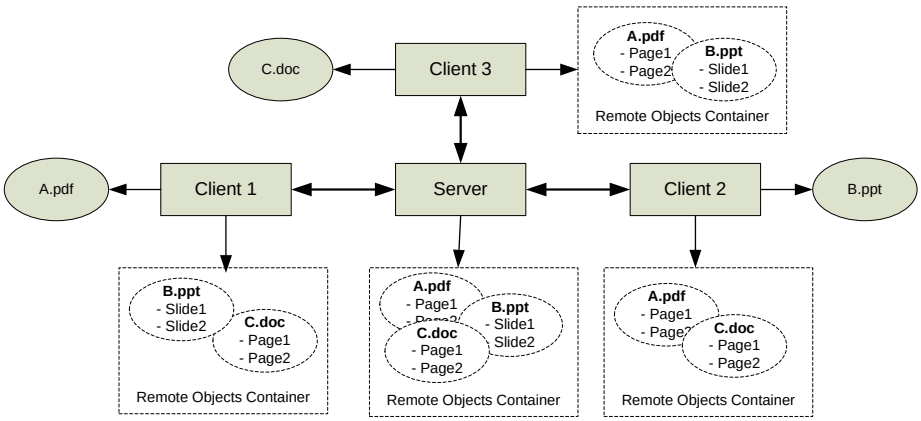

Abbildung 1: Remote Objects Container

Diese Meta-Informationen umfassen z.B. Platzierung im Shared Workspace, Dateigröße, Darstellungsgröße, Auflösung, Mime-Type, Beschreibung und Rotation des darzustellenden Objektes. Sie können jedoch generisch erweitert werden. Anhand dieser Informationen entscheidet der einzelne Teilnehmer, welche Objekte für ihn relevant sind.

Der eigentliche Zugriff auf Objekte, der mehr als 95% des übertragenen Datenvolumens ausmacht, erfolgt dann jedoch bedarfsgesteuert (pull). In dem Moment, in dem das entfernte Objekt in der Oberfläche des Nutzers dargestellt werden soll, wird überprüft, ob es im Remote Objects Container schon vorhanden ist. Ist dies nicht der Fall so wird ggf. ein Request mit der URI des Objektes an den Server gestellt. Dieser leitet dann den Request an den entsprechenden Client weiter. URIs werden dabei aus der User-ID des Besitzers und dessen Zugriffspfad unterhalb eines Root-Directories gebildet. So steht beispielsweise die URI

```
csp://client1/A.pdf/Page1
```

für die erste Seite der Datei „A.pdf“, die von Client 1 zur Verfügung gestellt wird.

Jedes Inhaltsobjekt kann von weiteren Objekten abhängig sein. So werden z.B. Schriftarten zur Darstellung benötigt, wenn die Seiten eines Dokuments als EMF-Dateien vorliegen [11]. Diese sekundären Objekte werden jedoch nicht über das Publish/Subscribe bekannt gemacht, sondern erst nach Anforderung des Objektes als URI-Liste in der Antwort auf den GET-Request mitgeliefert. Dieser Ansatz der sekundären Objekte ermöglicht einfache Erweiterbarkeit auf andere Anwendungsszenarien.

Eine Objekt-Übertragung erfolgt immer vom Besitzer des Objektes über den Server als Proxy zum anfragenden Client. Somit stellt jeder Client in der Groupware Session einen Mini-Server für seine Inhaltsobjekte dar. Diese Idee ist auch als Personal Web Serving bekannt und wurde bereits im YouServ-Projekt [3] demonstriert, bislang jedoch noch nicht für Echtzeit-Übertragungen angewendet. Das Konzept der Mini-Server ermöglicht das ad-hoc-Sharing von Inhaltsobjekten durch beliebige Teilnehmer und damit eine echte n:m-Verteilung. Ein weiterer Vorteil dieses Verfahrens ist die Möglichkeit der Optimierung der Effizienz der Content-Verteilung durch Caching und Prefetching. Jeder Remote Objects Container ist zugleich ein Cache. Für die Clients wirkt dieser Cache wie ein lokaler Browser-Cache, d.h. einmal angezeigte Objekte müssen nicht erneut geladen werden. Der Remote Objects Container des Servers wirkt dagegen wie ein Web Proxy Cache, der also die von mehreren Nutzern angefragten Objekte im Speicher hält und Anfragen anderer Clients nach diesen Objekten direkt beantworten kann.

4 Implementierung und Integration

In [12] wurde von uns bereits das Content Sharing Protocol vorgestellt, das adaptive Verteilung von Inhaltsobjekten erlaubt. Aufbauend auf diese Arbeiten wurde eine Implementierung des Ansatzes der Remote Objects Container realisiert. Das Protokoll und die Implementierung wurden entsprechend erweitert, um bedarfsgesteuerte n:m-Verteilung von primären und sekundären Objekten zu unterstützen. Abbildung 2 zeigt die dabei entstandene Anwendung Present and Share und das Videokonferenzsystem VidConference.

Das Present and Share wird aus VidConference heraus gestartet und baut parallel zur schon laufenden Videokonferenz-Session eine Groupware Session mit den bestehenden Teilnehmern auf. Innerhalb dieser Session kann jeder Nutzer Präsentationen und Dokumente importieren, deren Seiten dann in ein geeignetes Transferformat wie PNG, JPG oder EMF konvertiert werden. Über einen Notification Service wird für

jede Seite des Dokuments eine URI und eine zugehörige Beschreibung bekannt gemacht.

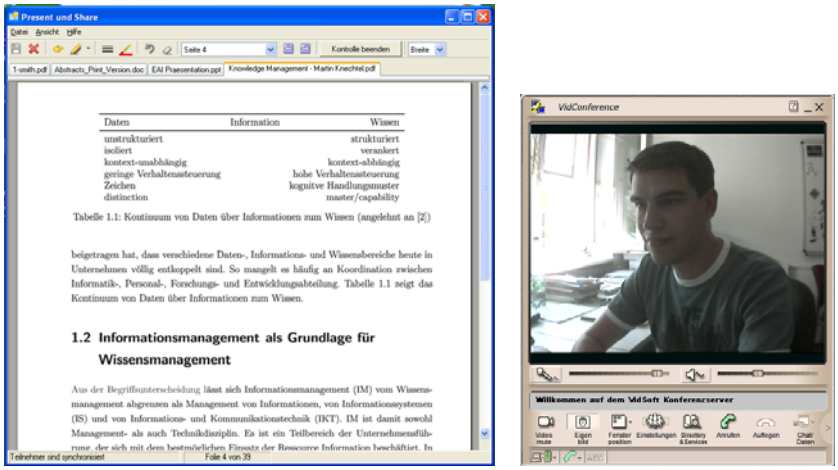

Abbildung 2: Screenshots VidConference mit Present and Share

Die anderen Nutzer bekommen daraufhin die in der Session zur Verfügung gestellten Dokumente als Tab in der Oberfläche angezeigt und können auf jedes Dokument wechseln und eine beliebige Seite anfordern und darstellen lassen. Der Notification Service überträgt die entsprechenden Meta-Informationen im Late-Join-Fall vollständig, so dass auch verspätet zur Session hinzugetretene Teilnehmer alle verfügbaren Dokumente angezeigt bekommen und diese betrachten können.

In [11] wurden Tests mit einer Stichprobe von 100 Word- und 100 PDF-Dokumenten durchgeführt. Die untersuchten Dokumente sind durchschnittlich 293 KB groß und haben im Mittel 19 Seiten. Obwohl die Dokumente bei der Konvertierung um durchschnittlich 120% an Größe zunehmen, ergibt sich durch die semantische Zerlegung und den bedarfsgesteuerten Zugriff auf beliebige Seiten des Dokumentes ein erheblicher Effizienzgewinn im Vergleich zum Push-Ansatz. Es werden durchschnittliche Antwortzeiten von weniger als 6 Sekunden erreicht, die durch Prefetching noch einmal erheblich verbessert werden.

5 Zusammenfassung und Ausblick

In diesem Paper wurde die Problematik der Synchronisierung gemeinsamer Objekte in synchronen Groupware-Sessions untersucht. Dabei wurde eine Taxonomie von Verteilungsmethoden vorgeschlagen, die die Kategorien 1:n push, 1:n pull, n:m push und n:m pull beinhaltet. Für die ersten drei Kategorien existieren bereits entsprechende Forschungsarbeiten, die hier analysiert wurden. Für die Kategorie n:m pull wurde ein an Personal Web Serving angelehntes Verfahren vorgestellt, die Remote Objects Container.

Die verschiedenen Ausprägungen synchroner Groupware zeigen ganz unterschiedliche Anforderungen hinsichtlich der Konsistenzsicherung für gemeinsame Objekte. Die hier vorgestellte Methode ist deshalb wie die anderen vorgestellten Methoden auch nicht für jede Art synchroner Groupware gleichermaßen geeignet. Sie hat vor allem dann entscheidende Vorteile, wenn die Synchronisierung in kurzer Zeit erfolgen muss und zum Synchronisierungszeitpunkt nur ein kleiner Teil der gemeinsamen Objekte von Interesse ist.

Nach der gerade abgeschlossenen Integration des Systems in die Videokonferenzsoftware VidConference sind umfangreiche Tests mit größeren Nutzergruppen im realen Einsatz geplant, um detaillierte Aussagen zu Performanceverhalten und Eignung der Methode für verschiedene Anwendungsszenarien treffen zu können.

6 Literatur

1. WebEx Inc., Webseite von WebEx, http://www.webex.com/, 2006.
2. Kerravala, Z., Hamilton, G., Unified Collaborative Communications for the Real-Time Enterprise, Yankee Group Report, The Yankee Group, 2004.
3. Bayardo Jr., R. J., Somani, A., Gruhl, D., Agrawal, R., YouServ: A Web Hosting and Content Sharing Tool for the Masses, 11th International World Wide Web Conference (WWW-2002), Honolulu, Hawaii, USA, 2002.
4. Holmer, T., Haake, J., Streitz, N., Kollaborationsorientierte synchrone Werkzeuge, In: G. Schwabe, N. Streitz, R. Unland, CSCW-Kompendium, Springer-Verlag, Berlin, Heidelberg, 2001.
5. Roseman, M., Greenberg, S., Building real time groupware with GroupKit, a groupware toolkit, ACM Transactions on Computer-Human Interaction, 3/1, 66-106, 1996.
6. CollabNet, Java Shared Data Toolkit - jsdt Project Home, https://jsdt.dev.java.net/, 2006.
7. Begole, J., Smith, R. B., Struble, C. A., Clifford, A. S., Resource Sharing for Replicated Synchronous Groupware, IEEE/ACM Transactions on Networking, 9/6, 2001.
8. Hill, R. D., Brinck, T., Rohall, S. L., Patterson, J. F., Wilner, W., The Rendezvous language and architecture for constructing multiuser applications, ACM Transactions on Computer-Human Interaction, 1/2, 81-125, 1994.
9. Graham, T. C. N., Urnes, T., Nejabi, R., Efficient distributed implementation of semi-replicated synchronous groupware, 9th annual ACM symposium on User interface software and technology, Seattle, Washington, United States, 1996.
10. Prakash, A., Shim, H. S., DistView: support for building efficient collaborative applications using replicated objects, 1994 ACM conference on Computer supported cooperative work, Chapel Hill, North Carolina, USA, 1994.
11. Küstner, M., Document Sharing für synchrone Groupware in heterogenen Anwendungsumgebungen, Diplomarbeit, TU Dresden, Fakultät Informatik, 2006.
12. Schuster, D., Kuemmel, S., Towards Adaptive Distribution of Multimedia Content within Collaborative Conferencing Sessions, 9th International Conference on Internet and Multimedia Systems and Applications, Honolulu, Hawaii, 2005.