Technologische Grundlagen des Computer Supported Cooperative Work (CSCW)

Silvano Maffeis
maffeis@ifi.unizh.ch

Institut für Informatik der Universität Zürich
Winterthurerstr. 190, CH–8057 Zürich
Tel: +41 1 257 4327 Fax: +41 1 363 0035

IFI TR 93.29

21. Juli 1993

Abstract

Diese Arbeit vermittelt einen Einblick in einige der für CSCW Systeme wichtigen technologischen Grundlagen. Als für das Gelingen von CSCW Applikationen wichtige technische Disziplinen erachten wir dabei verschiedene Aspekte der *verteilten Systeme*. Behandelt werden Konzepte wie Prozessgruppen, Broadcasting, Ordnung von asynchronen Ereignissen, virtuelle Synchronie und Replikation. Ebenfalls angesprochen werden für CSCW Anwendungen relevante Aspekte der Computer Kommunikation, wie Protokolle für Echtzeit Video und Audio Übertragung und *wide area networks*.

Inhalt

1 Einführung **1**

1.1 Dimensionen des CSCW . . . 1

1.2 Taxonomie von CSCW Anwendungen . . . 1

2 CSCW und Verteilte Systeme **2**

2.1 Anforderungen an verteilte CSCW Anwendungen . . . 3

3 Anforderungen an eine verteilte Entwicklungsplattform für CSCW Systeme **3**

3.1 Punkt-zu-Punkt Kommunikation . . . 3

3.2 Gruppen-Kommunikation . . . 4

3.3 Ordnung der Ereignisse . . . 4

3.4 Persistente Objekte . . . 5

3.5 Objekt-Replikation . . . 6

3.6 Kommunikations-Protokolle . . . 6

3.7 Beispiele . . . 6

3.8 Der CORBA Standard . . . 7

4 Anforderungen an das Kommunikations-Medium **7**

4.1 Kapazität . . . 8

4.2 Zweckmässigkeit . . . 8

4.3 Konstanter Durchsatz . . . 8

4.4 Netzwerk-Multicast . . . 9

5 Global verteilte CSCW Systeme **9**

5.1 Internet Wachstum . . . 9

5.2 Das IETF Internet Audiocast Experiment . . . 11

5.3 Das Terrestrial Wideband Network . . . 11

6 Ausblick **12**

1 Einführung

Ziel dieser Arbeit ist es Bedürfnisse, vorwiegend technischer Natur, von CSCW Anwendungen darzustellen und Ansätze vorzuschlagen welche zur Befriedigung dieser Bedürfnisse beitragen. Als für das Gelingen von CSCW Anwendungen wichtige Disziplinen erachten wir dabei *verteilte Systeme* und *Computer-Kommunikation*. Ebenso wichtig wie diese technischen Disziplinen, in dieser Arbeit aber nicht näher behandelt, sind Konzepte wie Mensch-Maschine-Schnittstellen, künstliche Intelligenz, soziologische Aspekte und so fort [EGG91].

1.1 Dimensionen des CSCW

Das primäre Einsatzgebiet von CSCW Systemen ist dort wo Individuen in ihrer Zusammenarbeit unterstützt werden sollen. Dabei kann eine *synchrone* und eine *asynchrone* Arbeitsweise unterschieden werden [RB91]. Bei der synchronen Arbeitsweise müssen die Beteiligten gleichzeitig präsent sein, was beispielsweise bei einem Telekonferenz-System der Fall ist. Die asynchrone Arbeitsweise spielt sich in der Regel über längere Zeiträume ab und eine simultane Präsenz der Beteiligten ist nicht nötig, was beispielsweise bei der Kooperation mittels Austauschen von Multimedia E-mails der Fall ist.

Die *geographische Distanz* ist ebenfalls ein wichtiges Klassifizierungs-Merkmal für CSCW Anwendungen. E-mail Systeme z.Bsp. werden eingesetzt wenn sich Interaktionspartner an unterschiedlichen Orten befinden, Präsentations-Systeme dann wenn sich die Interaktionspartner im selben Raum befinden.

Mittels den Dimensionen *Arbeitsweise* und *geographische Distanz* kann das in Abb. 1 angegebene, einfache Klassifikations-Schema aufgestellt werden.

Arbeitsweise

Distanz	synchron	asynchron
selber Ort	"Meeting Room Technology"	Elekronisches Anschlagsbrett
weit entfernt	Telekonferenz, Boardnoter	E-Mail

Abb. 1: Einfaches Klassifikations-Schema für CSCW Systeme.

Die in dieser Arbeit vorgestellten Technologien sollen für Applikationen aus allen vier Quadranten in Abb. 1 massgebend sein. Solche Applikationen werden oft als *Groupware* bezeichnet [EGG91]:

computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.

1.2 Taxonomie von CSCW Anwendungen

CSCW Anwendungen können anhand folgender systemorientierten Analysekriterien untersucht werden [Kol93]. Diese Kriterien stellen eine Verfeinerung und Erweiterung der im letzten Kapitel vorgestellten Klassifikation dar.

Interaktion : Die Art wie mittels einer CSCW Anwendung interagiert wird erlaubt Rückschlüsse auf die Technologien die bei der Realisierung der Anwendung massgebend sind. Wiederum kann man synchrone und asynchrone Interaktion unterscheiden. Wichtig ist aber auch die Gliederung nach der Struktur der Kommunikations-Partner:

- Punkt-zu-Punkt Kommunikation via *message-streams* oder RPC. Die Kommunikation erfolgt dabei von Individuum zu Individuum.
- Gruppen-Kommunikation mittels Prozessgruppen [Bir91] oder *Thread*-Gruppen [BSS91]. Die Kommunikation erfolgt dabei vom Individuum zur Gruppe, von Gruppe zu Gruppe oder von der Gruppe zum Individuum hin.

Verteilung : Dabei steht die geographische Verteilung der Interaktionsgruppe im Vordergrund. Der Grad der Verteilung kann von lokal (im selben Raum) bis zu global (weltweit verteilt) variieren.

Konsistenz : Grundsätzlich versteht man unter Konsistenz die Widerspruchsfreiheit zwischen den Zuständen von unter Umständen replizierten Objekten. In einem verteilten *Boardnoter* System [S$^+$88] beispielsweise kann man sich die virtuelle Wandtafel welche jeder Benutzer auf seinem Bildschirm sieht als repliziertes Objekt vorstellen. In solch einem WYSIWIS System sollten die Bildschirminhalte unterschiedlicher Benutzer konsistent, also widerspruchsfrei sein.

Wichtig für die Konsistenzerhaltung ist aber auch die Koordination von verteilten Aktivitäten. In einem *Group-Editor* beispielsweise kann verlangt werden, dass Schreibzugriffe auf ein Objekt (einen Text-Abschnitt z.Bsp.) exklusiv erfolgen (readers-writers Synchronisation). Zur Einhaltung dieser Forderung können aus der Datenbank-Forschung bekannte Synchronisations-Algorithmen wie Token-Passing, Weighted Voting [Gif79] oder Quorum-Algorithmen [Her86] zum Zuge kommen.

Dort wo infolge permissiveren Konsistenzbedingungen oder globaler Verteilung solche strikte Algorithmen nicht in Frage kommen können Techniken wie *Lazy Consistency* [NG92] oder *update propagation* [Gol91] zum Zuge kommen.

Virtuelle Synchronie [Lam78, BJ87, vR93] als fein-granulare Methode verteilte Prozesse zu synchronisieren ist für verteilte Groupware Systeme besonders interessant. Auf den Begriff der virtuellen Synchronie wird in Kapitel 3.3 eingegangen.

Gruppendynamik : Die an einer CSCW Session beteiligten Instanzen (Prozesse) haben zu jedem Zeitpunkt eine eigene Vorstellung welche anderen Instanzen ebenfalls zur Kommunikationsgruppe gehören und folgedessen über Aktivitäten innerhalb der Gruppe informiert werden müssen. Zudem sollen Instanzen dynamisch solchen Gruppen beitreten oder diese verlassen können. Ausfälle von Rechnern oder Netzwerkverbindungen bewirken ein "unfreiwilliges" austreten aus einer Gruppe. Die Technologie zur Realisierung von CSCW Systemen sollte folglich die Bildung von Kommunikationsgruppen ermöglichen. Auf Kommunikationsgruppen wird in Kapitel 3.2 eingegangen.

2 CSCW und Verteilte Systeme

Da CSCW Anwendungen oft örtlich und zeitlich *verteilt* ablaufen ist es zweckmässig solche Anwendungen als *verteilte Systeme* [Mul89, TvR85, Tan92] zu betrachten. Im folgenden wollen wir eine verteilte CSCW Anwendung unter den Begriff des verteilten Systemes subsumieren. Ein verteiltes System kann wie folgt definiert werden [TvR85]:

A distributed [operating] system is one that looks to its users like an ordinary centralized [operating] system, but runs on multiple, independent CPUs. The key concept here is transparency, in other words, the use of multiple processors should be invisible (transparent) to the user. Another way of expressing the same idea is to say that the user views the system as a "virtual uniprocessor", not as a collection of distinct machines.

2.1 Anforderungen an verteilte CSCW Anwendungen

Es sollen folgende Anforderungen an ein verteiltes CSCW System gestellt werden [Tan92]:

Transparenz : Wie in obiger Definition angegeben sollte der Benutzer im Idealfall nichts von der eigentlichen verteilten Natur des zugrundeliegenden Computer-Systemes merken. Diese Transparenz wird vorwiegend in Software realisiert und es ist diese Software-Schicht welche ein Computernetzwerk zu einem verteilten System macht.

Flexibilität : Ein verteiltes System sollte erweiterbar und rekonfigurierbar sein. Ein interessanter Ansatz dazu stellen Microkernel basierte verteilte Systeme wie das AMOEBA Betriebssystem [M+89] dar. Hierbei wird ein minimaler Betriebssystem-Kern auf den Beteiligten Maschinen installiert. Dienste wie File-Serving, Directory-Serving, Gateway-Serving oder Dienste einer verteilten CSCW Applikation werden als eigenständige Module (services) zur Laufzeit dem System beigefügt. Ein solcher Dienst kann ganz im Sinne der objekt-orientierten Programmierung als *Objekt* betrachtet werden welches einen inneren Zustand über eine klar definierte Schnittstelle zugänglich macht. Es ist deshalb naheliegend objekt-orientierte Techniken für die Realisierung flexibler, verteilter Systeme anzuwenden [Maf93].

Zuverlässigkeit : Der Ausfall einer einzelnen Ressource (z.Bsp. eines File-Servers) in einem verteilten System sollte nicht zu einem Ausfall des Gesamtsystemes führen. *Verfügbarkeit* kann mittels Redundanz, also Objekt-Replikation, erreicht werden. *Ausfalltoleranz*, die zweite Dimension der Zuverlässigkeit, kann mittels Protokollen welche Ausfälle maskieren und das wiederaufstarten von Diensten ermöglichen erreicht werden.

Leistung : Die Leistung eines verteilten Systemes ist ebenfalls wichtig. Im Falle von CSCW Anwendungen müssen möglichst geringe und gleichbleibende Antwortzeiten durch die Wahl geeigneter Kommunikations-, Konsistenz- und Synchronisations- Protokolle gewährleistet werden damit das System Akzeptanz finden kann.

Skalierbarkeit : Es ist anzunehmen dass künftige CSCW Systeme, z.Bsp. Telekonferenz Systeme oder Concurrent Engineering Systeme [DR93], globale Dimensionen erreichen werden und tausende von kooperierenden Individuen weltweit umfassen werden. Bei der Wahl von Technologien zur Kommunikation, Synchronisation und Konsistenz-Erhaltung ist auf Skalierbarkeit vermehrt zu achten.

3 Anforderungen an eine verteilte Entwicklungsplattform für CSCW Systeme

In diesem Kapitel werden Anforderungen an eine Programmierumgebung oder Entwicklungsplattform gestellt. Bei den meisten derzeitigen CSCW Anwendungen wurde von bestehenden Betriebssystem Schnittstellen, z.Bsp. Socket-Interprozesskommunikation in UNIX, ausgegangen. Wir erachten diese Dienste aber als nicht zweckmässig genug für die Realisierung verteilter CSCW Anwendungen und stellen deshalb weitergehenden Anforderungen an die Programmierumgebung.

3.1 Punkt-zu-Punkt Kommunikation

CSCW Anwendungen wie Multimedia E-mail Systeme operieren vorallem über Punkt-zu-Punkt Verbindungen. Die Entwicklungsplattform für CSCW Systeme sollte demnach um solche Anwendungen unterstützen zu können Mechanismen zur zuverlässigen Punkt-zu-Punkt Kommunikation bieten. Darunter Fallen vorallem Message-streams (z.Bsp. TCP/IP, X.25, TP4) und *Remote Procedure Calling (RPC)* [Nel81].

3.2 Gruppen-Kommunikation

Prozessgruppen [Bir91] erachten wir als das geeignete Programmier- und Kommunikations-Paradigma für diverse Arten von CSCW Anwendungen. Eine Entwicklungsplattform wie Isis [B^+90] oder Horus [vRBC$^+$92] stellt Mechanismen zur Verfügung welche es erlauben über zahlreiche Rechner verteilte Gruppen von Prozessen (in Isis) oder gar Gruppen von *Threads* (leichtgewichtige Prozesse, in Horus) zu verwalten. Isis und Horus werden in Kapitel 3.7 kurz vorgestellt. Prozessgruppen können dann mittels *broadcast* Operationen als Einheit angesprochen werden. In einem Boardnoter System [S^+88] beispielsweise kann jede Aktion eines Benutzers zu allen anderen Benutzer im System mit einer einzigen broadcast Nachricht versandt werden. Die Entwicklungsplattform stellt dabei sicher dass die Nachricht bei jedem Gruppenmitglied oder bei gar keinem, dann nämlich wenn sich ein Ausfall eines Gruppenmitgliedes ereignet hat, ankommt. Dies wird als *reliable broadcast* [KTHB] bezeichnet.

Prozessgruppen und Gruppen-Zugehörigkeit sind demnach ein wichtiges Merkmal von CSCW Anwendungen. Die Entwicklungsplattform für CSCW Anwendungen sollte zudem über Mechanismen verfügen welche die Gruppensicht welche jedes Gruppenmitglied zu einem gegebenen Zeitpunkt hat, automatisch konsistent hält. Derartige Mechanismen sind im bereits erwähnten Horus Toolkit realisiert und in [Ric92] beschrieben.

3.3 Ordnung der Ereignisse

Der Ordnung der Ereignisse in einem verteilten CSCW System kommt eine wichtige Bedeutung zu. Beispielsweise Boardnoter Systeme müssen gewährleisten, dass die Aktionen der Beteiligten, z.Bsp. das Schreiben von Text auf die virtuelle Wandtafel, das Plazieren einer Figur, das Manipulieren eines *telepointers* etc., bei allen Beteiligten in der gleichen Reihenfolge erfolgen. Das Ordnen von Ereignissen in einem verteilten System soll nun anhand von drei Modellen untersucht werden, wobei wir dabei die Rolle eines "externen Betrachters" annehmen. Im folgenden wird von einer CSCW Anwendung welche mittels Prozess-Gruppen strukturiert ist ausgegangen.

In einem *synchronen* System (siehe Abb. 2) erfolgt jegliche Kommunikation ohne zeitliche Verzögerung, das heisst jede am verteilten System beteiligte Gruppen-Instanz (ein Prozess oder ein *Thread*) sieht sämtliche Ereignisse in derselben Reihenfolge. Da in einem reellen verteilten System jegliche Kommunikation eine zeitliche Verzögerung bedingt und zudem keine gemeinsame Zeit [Lam78] existiert entspricht dieses Modell nicht der Realität. Es dient lediglich dem besseren Verständnis der nun folgenden zwei Modelle.

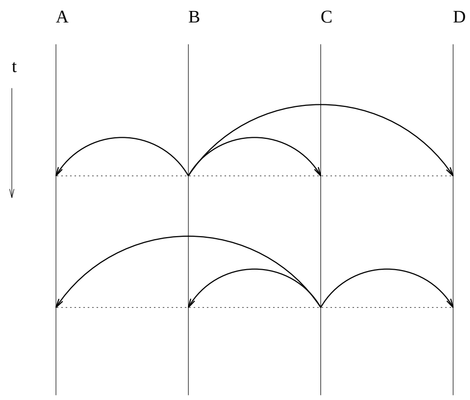

Abb. 2: Ereignisse in einem synchronen, verteilten System. B und C führen broadcasts aus welche ohne zeitliche Verzögerung bei den Empfängern eingehen. Folgedessen sieht jedes Prozessgruppen-Mitglied beide broadcast-Ereignisse in derselben Reihenfolge.

In einem *lose-synchronen* System (siehe Abb. 3) treten zwar zeitliche Verzögerungen zwischen dem Absenden und dem Empfangen einer Nachricht auf, jedoch kommen sämtliche Ereignisse nachwievor in derselben Reihenfolge bei allen beteiligten Gruppen-Instanzen an. Ein broadcast-Protokoll welches diese Eigenschaften garantiert ist das im Isis Toolkit [B^+90] realisierte *abcast – atomically ordered process-group broadcast* Protokoll oder das in AMOEBA eingesetzte broadcast Protokoll [KTHB]. Die Komplexität des Isis abcast Protokolles ist aber überlinear bezüglich der Anzahl Prozessgruppen-Mitglieder [BSS91] und deshalb für CSCW Anwendungen mit zahlreichen Instanzen ungeeignet.

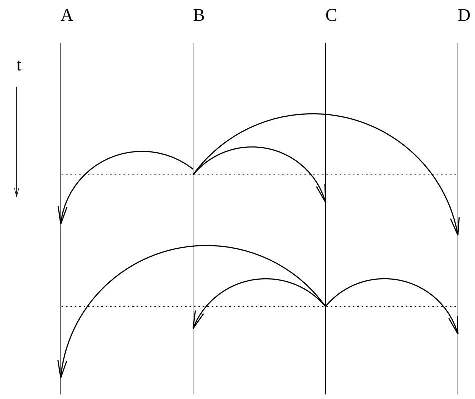

Abb. 3: Ereignisse in einem lose-synchronen, verteilten System. Die beiden broadcasts werden mit einer zeitlichen Verzögerung empfangen. Das *abcast* Protokoll garantiert aber dass alle Prozesse dieselbe Ordnung der Ereignisse erfahren.

Diese strenge Ordnung der Ereignisse kann in vielen Fällen aber abgeschwächt werden um zu einer *virtuell-Synchronen* Ordnung [BJ89, vR93] zu gelangen (siehe Abb. 4). Eine virtuell-synchrone Ausführung verhält sich zu einer lose-synchronen Ausführung ähnlich wie sich serialisierbare Datenbank-Transaktionen zu seriell ablaufenden Transaktionen verhalten. In virtuell-synchronen Systemen werden Ereignisse nur dann synchronisiert, wenn diese *kausal* [Lam78] voneinander abhängig sind. Zwei Ereignisse sind dann voneinander kausal abhängig wenn ein Informationsfluss, z.Bsp. in Form von RPCs oder Message-passing, direkt oder indirekt zwischen diesen zwei Ereignissen stattfindet. In vielen Fällen genügt es in einem verteilten System nur kausal voneinander abhängigen Ereignisse zu synchronisieren. Nicht kausal-abhängige Ereignisse dürfen sich dann ungehindert im Netzwerk überholen. Ein Protokoll welches virtuell-synchrone Prozessgruppen-Kommunikation ermöglicht ist das Isis *cbcast – causal broadcast* Protokoll. Die Komplexität dieses Protokolles verhält sich linear zu der Anzahl Gruppen-Instanzen. Es ist folgedessen geeignet um skalierbare CSCW Anwendungen zu realisieren welche eine kausale Ordnung der Ereignisse gewährleisten. In einem verteilten System kann Kausalität ohne Verwendung von zusätzlichen Nachrichten gewahrt werden, indem die vom Synchronisations-Protokoll benötigte Information an die versandten Nutzdaten gehängt wird (*piggybacking*). Ein einfaches causal-broadcast Protokoll welches mittels Zeitvektoren arbeitet ist in [Tan92][S. 458] beschrieben.

3.4 Persistente Objekte

Da in CSCW Anwendungen der Zustand von Objekten, z.Bsp. der Bildschirm-Inhalt in einer Boardnoter Session, Metadaten zu einem Projekt welches mittels eines Concurrent-Engineering Systemes verwaltet wird, abgespeichert werden müssen, sind persistente Objekte eine wichtige Anforderungen an die Entwicklungsplattform. Transparente Übertragung von Objekten über ein Netzwerk verlangt zudem ähnliche Mechanismen wie sie in persistenten objekt-orientierten Datenbanken realisiert sind [BR92].

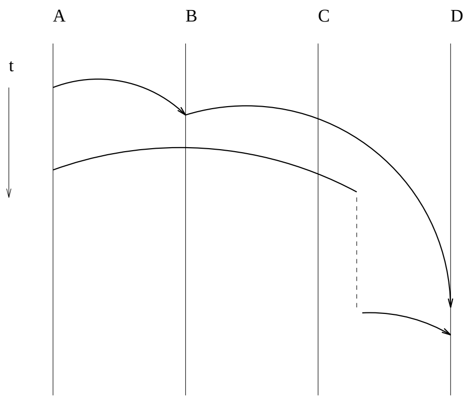

Abb. 4: Ereignisse in einem virtuell-synchronen, verteilten System. In diesem neuen Szenario sendet A eine Punkt-zu-Punkt Nachricht an B, und B wiederum an D. Da die folgende Nachricht von A nach D Kausal mit derjenigen von B nach D zusammenhängt, muss in diesem Fall die erstgenannte Nachricht vom System abgebremst werden (gestrichelte Linie) damit sie an zweiter Stelle bei D ankommt.

3.5 Objekt-Replikation

Objekt-Replikation dient in einem WYSIWIS System dazu einen über die Instanzen replizierten Zustand darzustellen oder Ausfalltoleranz zu realisieren. Replizierte Objekte können widerum mittels Prozessgruppen verwaltet und konsistent gehalten werden.

3.6 Kommunikations-Protokolle

Da CSCW Systeme oftmals Individuen aus den verschiedensten Bereichen (Unternehmungen, Universitäten, öffentlich-rechtliche Betriebe) in unterschiedlichen Ländern verbinden sollte die Entwicklungsplattform auch verschiedene Kommunikations-Protokolle und -Standards unterstützen. Protokolle und Standards wie Ethernet, ATM, ISDN, TCP/IP, Mach-Messages, X.25 etc. sollten von derselben CSCW Anwendung auch gleichzeitig verwendbar sein.

3.7 Beispiele

In diesem Abschnitt werden kurz drei Entwicklungsplattformen für verteilte Systeme vorgestellt welche die wichtigsten der oben aufgeführten Anforderungen erfüllen.

ISIS [B+90] ist eine bekannte, kommerzielle Programm-Bibliothek in C mit Hilfe derer ausfalltolerante verteilte Systeme realisiert werden können. ISIS unterstützt Prozessgruppen, RPC, virtuelle Synchronie usw. ISIS wurde ursprünglich an der Cornell University, New York unter der Leitung von Prof. Kenneth P. Birman entwickelt. Die neueren Versionen (ab Version 2.2) werden von der Firma ISIS Distributed System Inc., New York vertrieben. Die Version 2.1 von ISIS wird unentgeltlich zur Verfügung gestellt.

HORUS [vRBC+92] ist ein Forschungsprojekt der Cornell University und zielt auf die Neu-Implementierung von ISIS ab. Der grösste Nachteil von ISIS liegt darin dass nur Gruppen von "schwergewichtigen" Prozessen unterstützt werden. Praktische Erfahrungen (siehe beispielsweise [SBM89]) haben gezeigt dass grössere verteilte Anwendungen mit zahlreichen (> 1000) Prozessgruppen signifikante Leistungseinbussen erfahren und nicht gut skalieren. HORUS wählt folgedessen einen "leichtgewichtigen" Ansatz, indem *threads* (leichtgewichtige Prozesse) zu einzelnen Gruppen zusammengefasst werden können. Zudem Unterstützt HORUS gleichzeitig verschiedenen Betriebssysteme

(SUN OS, HP-UX, AIX, MACH, Amoeba, X-Kernel, Chorus etc.) und verschiedenene Netzwerk-Protokolle (UDP, TCP, IP, Deering-IP, Amoeba-Flip, Mach Messages, Raw Ethernet etc.). Hardware-broadcast Vorrichtungen (z.Bsp. bei Ethernet der Fall) werden von Horus Anwendungen genutzt falls vorhanden. Das Horus Toolkit wird unentgeltlich zur Verfügung gestellt. Ein Nachteil von Horus, Isis und anderen modernen Toolkits für verteilte Anwendungen ist, dass die gebotene Programmierschnittstelle "low-level" und sehr Betriebssystem nah ist. Die Kommunikation zwischen Horus-Anwendungen erfolgt hauptsächlich dadurch dass Buffer von unstrukturierten Daten versandt werden.

Electra [Maf93] baut auf dem Horus Toolkit auf und erweitert dieses mit objekt-orientierten Abstraktionen. Mittels einer eigenen Spezifikationssprache (der Electra Service Definition Language) können rechnerfremde Dienste über ihre Schnittstellen auf hoher Abstraktionsebene spezifiziert werden. Neue Dienste können mittels Vererbung (multiple inheritance) aus bereits realisierten Diensten abgeleitet werden. Electra erlaubt dem Programmierer transparent ganze Objekte mittels Punkt-zu-Punkt oder broadcast Operationen zu versenden. Eine einfache Form von Objekt-Persistenz ist ebenfalls gegeben. Die Realisierung verteilter Electra-Anwendungen erfolgt in C++.

3.8 Der CORBA Standard

Entwicklungsplattformen welche anerkannte, händlerunabhängige Standards berücksichtigen erlauben offene CSCW Anwendungen zu realisieren. Als Beispiel eines solchen neuartigen Standards soll CORBA (Common Object Request Broker Architecture) [Dig91, Bey93] kurz vorgestellt werden.

Wie von der OMG (Object Management Group) spezifiziert besteht ein Common Object Request Broker aus Mechanismen mit Hilfe deren Objekte auf unterschiedlichen Rechnern transparent kommunizieren können. Der CORBA Standard wird von der OMG entwickelt und von den Firmen DEC, HP, HyperDesk, NCR, Object Design und SunSoft unterstützt.

CORBA erlaubt verteilte Anwendungen objekt-orientiert für heterogenen Umgebungen zu realisieren. Unterstützung unterschiedlicher Betriebssysteme gewährleistet der Object Request Broker (ORB) der auf jeder beteiligten Rechenanlage installiert wird. Der ORB bietet eine Betriebssystem unabhängige Schnittstelle für die Kommunikation, für das Auffinden von Objekten usw. an.

Unterstützung unterschiedlicher Programmiersprachen geschieht in CORBA dadurch dass die Schnittstellen rechnerfremder Dienste in einer eigenen Deklarationssprache, der CORBA-IDL (Interface Definition Language) deklariert werden. Diese Schnittstellen Deklarationen werden dann von einem Stub Generator in die entsprechende Programmiersprache (C, C++, Smalltalk, Lisp, usw.) übersetzt.

Der Vorteil von CORBA im CSCW Bereich ist dass CSCW Systeme durchgehend objektorientiert modelliert und realisiert werden können und dass die Kommunikation in CORBA sehr expressiv ist. Diese Expressivität äussert sich dadurch dass ganze Benutzer-definierte Objekte (z.B. Graphische Objekte in einem Telekonferenz-System, Windows etc.) übermittelt werden können und ein verteiltes System wie in 2.1 gefordert mittels abstrakten Diensten in CORBA-IDL modelliert werden kann.

Die Schwäche der derzeitigen CORBA Spezifikation ist dass broadcasting, Replikation, Ordnung verteilter Ereignisse, Behandlung von Ausfällen (consistent failure reporting [BG93]) im Standard nicht enthalten sind, obwohl diese Aspekte wichtig für das realisieren zuverlässiger verteilter Systeme sind.

4 Anforderungen an das Kommunikations-Medium

In diesem Kapitel sollen Anforderungen an die physikalischen Netzwerkverbindungen und an die Netzwerkprotokolle welcher einer CSCW Anwendung zugrundeliegen gestellt werden.

4.1 Kapazität

Das physische Übertragungsmedium muss in der Lage sein die für die CSCW Anwendung erfordeliche Kapazität (bits/s) zu liefern. Tabelle 1 gibt Beispiele von CSCW Datenströmen, der erforderlichen Kapazität und möglicher heutiger Technologien dazu. Tabelle 2 gibt Beispiele heutiger Kommunikations-Medien und deren maximalen Kapazität. (K = Kilo, M = Mega, G = Giga bits/s).

Datenstrom	Kapazität (bits/s)	Technologie
Text (Anzeige)	9.6 K	Modem, Twisted Pair
Sprache	64 K	Twisted Pair, Mietleitung, ISDN
Audio	bis 1.4 M	ISDN, Ethernet, IBM Token Ring
Video	mehrere M	Ethernet, B-ISDN, ATM, Fiberoptik

Tabelle 1: Datenströme, erforderliche Kapazitäten und Technologien.

Technologie	max. Kapazität (bits/s)
IBM Token Ring	4 M
Ethernet	10 M
Coaxial Kabel	31 M
64 μm Fiberglass	31 M
FDDI	100 M
Telephon Leitung (Analog, Twisted Pair)	30 K
Telephon Leitung (Digital, Twisted Pair)	2 M
Arpanet Mietleitung	56 K
PTT Mietleitung	9.6 K – 2 M
T1 Links (Internet)	1 M
T3 Links (Internet)	45 M
Satellit	2 K – mehrere G

Tabelle 2: Netzwerk-Technologien und deren maximale Kapazität.

4.2 Zweckmässigkeit

Eine Netzwerk-Technologie ist für eine CSCW Anwendung dann zweckmässig, wenn diese die Anforderungen der Anwendung erfüllt und zudem die erforderlichen Hardware und Software Schnittstellen vorhanden sind sodass die Netzwerk-Technologie mühelos an die beteiligten Systeme angeschlossen werden kann. Anerkannte Kommunikations-Standards wie TCP/IP oder X.25 führen eher zu zweckmässigen Lösungen als exotische, händlerspezifische Protokolle.

Unterschiedliche Netzwerk-Technologien können heute mittels, oftmals teuren, Gateway-Produkten integriert werden. Beispielsweise können NovelNet, TOPS, Appletalk, TCP/IP Ethernet etc. miteinander verbunden, TCP/IP Verkehr über serielle Leitungen (SLIP) und X.25 Verbindungen (IP-PAD) geroutet, weitentfernte Ethernet-Segmente über Telefonleitungen gekoppelt werden usw.

4.3 Konstanter Durchsatz

Verteilte CSCW Anwendungen welche Echtzeit Audio und Video unterstützen benötigen von der darunterliegenden Kommunikationsschnittstelle Garantieen bezüglich dem Durchsatzwert mit dem die Audio oder Video Daten geliefert werden. Einfach ist es wenn das CSCW System über einer Netzwerkarchitektur mit virtuellen Verbindungen (*virtual circuits* [Tan89]) oder gar dedizierten physikalische Verbindungen arbeitet. Bei solchen Netzwerken wird ein bestimmter Durchsatzwert konstant gehalten.

Anders ist es bei verbreiteten Netzwerken wie Ethernet oder allgemein Paket-orientierten Netzwerken (WANs wie das *Internet* [Keh92] beispielsweise), wo im Extremfall jedes einzelne Datagramm eine eigene Route einschlagen kann. Hier ist es dem Übertragungs-Medium heute meist nicht möglich einen konstanten Durchsatz zu garantieren. Das Problem kann aber abgeschwächt werden indem eine Software-Schicht realisiert wird welche zu den einzelnen CSCW-Sessionen *read-ahead buffers* verwaltet und für die CSCW Applikation Daten im Voraus buffert. Dieser Buffer dient dazu Durchsatzschwankungen aufzufangen indem die Applikation Daten aus dem Buffer und nicht direkt von der Netzwerkverbindung liest. Für Techniken zur Realisierung solcher Buffering-Schichten sei auf [Kur93] verwiesen. Auf ein WAN Protokoll welches konstante Datenströme garantiert wird in Kapitel 5.3 eingegangen.

4.4 Netzwerk-Multicast

Unter Netzwerk-Multicast versteht man eine Hardware oder Software Vorrichtung welche ermöglicht dass *dasselbe* Netzwerk-Paket von mehreren Stationen empfangen wird. Mittels einer solchen Vorrichtung kann, gerade bei Telekonferenz Systemen, die Netzbelastung erheblich vermindert werden [PP90], da eine Datenpaket nur einmal versandt werden muss, jedoch von allen Teilnehmern empfangen wird.

Multicast-Unterstützung kann Hardwaremässig erfolgen (z.Bsp. Ethernet nach IEEE 802.3 [IEE]) oder in Software realisiert werden (z.Bsp. Deering IP-Multicasting [Dee89]). Prozessgruppen Kommunikation wird mittels darunterliegendem Netzwerk Multicast bedeutend leistungsfähiger indem ein Prozessgruppen-broadcast mittels nur einer Kommunikations-Operation abgewickelt werden kann. Wo kein Netzwerk Multicast vorhanden ist muss ein Prozessgruppen-broadcast mittels einer Punkt-zu-Punkt Kommunikation *pro Gruppenmitglied* abgearbeitet werden.

5 Global verteilte CSCW Systeme

Künftige CSCW Systeme werden vermehrt auch für die Kooperation innerhalb global verteilter Gruppen von Individuen Einsatz finden. Beispielsweise wird es für international agierende Unternehmungen zunehmend wichtig, innert kürzester Zeit auf Änderungen in ihrer Umgebung (z.Bsp. neue Konkurrenten, Lieferanten, andere Kundenbedürfnisse, neue Gesetze) global reagieren zu können. Philip Kotler spricht in diesem Zusammenhang von der Ära des “turbo marketing” [Kot90], in der es der multinational agierenden Unternehmung gelingen muss schnell auf Änderungen in ihrem Umfeld zu reagieren und kostengünstig qualitativ hochstehende Produkte herzustellen.

Global verteilte CSCW Systeme können einer global agierenden Unternehmung dazu verhelfen die Entscheidungsfindung zu beschleunigen (mittels globalen Telekonferenz Systemen z.Bsp.), oder die Zusammenarbeit zwischen weit entfernten Projektteams zu verbessern (mittels Concurrent Engineering Systemen z.Bsp.). Global verteilte CSCW Systeme können somit zu Wettbewerbsvorteilen verhelfen.

In diesem Kapitel soll vorallem auf Beobachtungen und Erfahrungen bezüglich global verteilten CSCW Anwendungen am Internet eingegangen werden.

5.1 Internet Wachstum

Die globale Vernetzung von lokalen Netzwerken an Universitäten, Forschungslabors und je länger je mehr auch Unternehmungen schreitet rasch voran. Das weltweit grösste autonome Netzwerk ist heute das Internet [Keh92]. Abb. 5 zeigt eindrücklich wie seit 1981 die Anzahl Rechner am Internet rapide angestiegen ist. Gegenwärtig beträgt die Wachstumsrate der

Anzahl Rechner am Internet ca. 30% im halben Jahr. Die Anzahl Rechner per Mai 1993 wird auf 1.5 Millionen geschätzt.

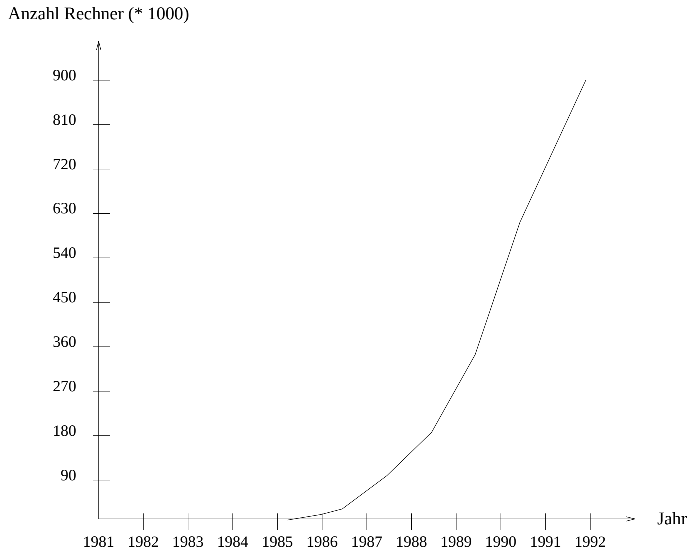

Abb. 5: Internet Wachstum 1981 bis 1992.

Momentan befinden sich einige wenige globale CSCW Anwendungen am Internet im Einsatz. Es handelt sich dabei vorwiegend um Anwendungen welche das asynchrone kooperieren von Individuen unterstützen, z.Bsp. E-mail, Gopher, World-Wide-Web etc. Echtzeit Video und Audio basierte Anwendungen wie Telekonferenz-Systeme konnten am Internet noch nicht erfolgreich eingesetzt werden. Grund dafür sind die niedrigen (siehe Tabelle 3) und stark schwankenden Durchsatz- sowie Antwortzeit-Werte für Internet-Verbindungen.

Internet-Domain	Land	Durchschn. Bandbreite (kbits/s)
at	Österreich	4.8
au	Australien	21.6
be	Belgien	3.2
ch	Schweiz	147
de	Deutschland	4.8
fr	Frankreich	21.6
gr	Griechenland	0.8
it	Italien	28.8
jp	Japan	12.0
nl	Holland	11.2
tw	Taiwan	15.2
uk	England	29.6
za	Süd-Afrika	9.6
com	Commercial	26.4
edu	Educational	60.8

Tabelle 3: Durchschnittliche Kapazität von Internet Verbindungen zur Universität Zürich. Gemessen für FTP Verkehr im Monat April 1993.

5.2 Das IETF Internet Audiocast Experiment

Das Audio zum Meeting der *Internet Engineering Task-Force* von 1992 wurde als Experiment zu 20 Internet-Sites in 16 verschiedenen Zeitzonen live übertragen [Cas92]. Die Vorträge welche an diesem Meeting stattfanden wurden in Echtzeit digitalisiert und über das UDP Protokoll und eines mit multicast-Erweiterungen [Dee89] versehenen IP Protokolles global übertragen. Zusammengefasst die Beobachtungen und Erfahrungen welche in diesem Experiment gemacht wurden:

- Die Qualität des empfangenen Tones schwankte fortlaufend zwischen “sehr klar” und “komplett unverständlich”. Dies weil Protokolle wie IP keine stabilen Transferraten und Antwortzeiten garantieren können.
- Da Gesprächspausen ebenfalls digitalisiert und übertragen wurden war das Netzwerk konstant überlastet und die beteiligten IP-Router Hosts hatten nur selten die Möglichkeit ihre Paket-Warteschlangen zu entleeren.
- Unerklärliche Paket-Verluste von bis zu 20% wurden gemessen.
- “Resource-control” Algorithmen welche auf den Router-Hosts laufen sind eine Notwendigkeit um synchrone CSCW Applikationen am Internet zu ermöglichen. Diesbezügliche Arbeiten sind bereits im Gange und werden im nächsten Kapitel angeschnitten.

5.3 Das Terrestrial Wideband Network

Das *Terrestrial Wideband Network (TWBNet)* ist eine DARPA Testumgebung mit Hilfe derer hochgeschwindigkeits Protokolle und WAN basierte CSCW Applikationen untersucht werden [Seo90]. TWBNet verbindet 15 LANs mittels 1.5 MBps T1 Verbindungen (Abb. 6). TWBNet wird unter anderem für transkontinentale Telekonferenzen (Audio und Video) eingesetzt. Telekonferenz-Räume mit Workstations sind in San Francisco, Los Angeles, Washington D.C. und Boston stationiert.

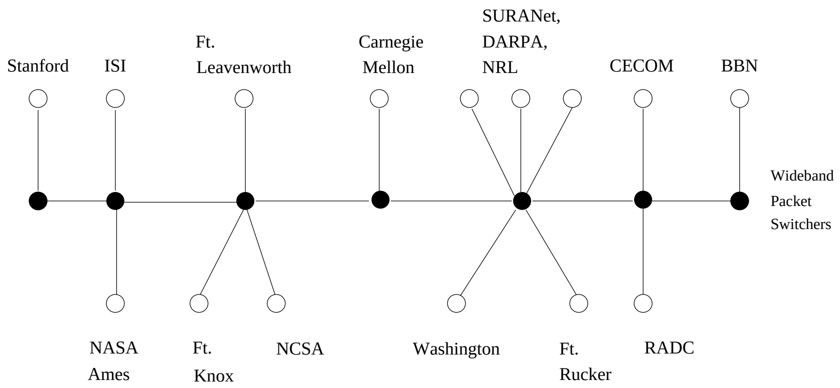

Abb. 6: Topologie des Terrestrial Wideband Networks.

Auf den TWBNet Gateways ist ein spezielles Protokoll, das *ST Stream Protocol*, installiert welches es CSCW Anwendungen erlaubt, Anforderungen an Bandbreite und Antwortzeiten den Gateways anzugeben. Die Gateways können dann die angeforderten Ressourcen allozieren und der Applikation zur Verfügung stellen.

Auf Ebene des Link-Layers (OSI Schicht 2) befindet sich das TWBNet Dual Bus Protocol (DBP) welches ähnlich zum IEEE 802.6 MAN Protokoll arbeitet. Das DBP ist dafür zuständig die von den Gateways angeforderten Bandbreiten stabil zu halten. Multicast ist im DBP ebenfalls vorgesehen.

6 Ausblick

In dieser Arbeit wurde überblicksartig auf technologische Fragen welche sich im Zusammenhang mit der Entwicklung verteilter CSCW Anwendungen ergeben eingegangen. Es wurden Anforderungen an Entwicklungsplattformen für verteilte CSCW Systeme aufgestellt, wobei sich die Wichtigkeit der Prozessgruppen-Kommunikation (reliable broadcast) und der Ordnung von CSCW Ereignissen (Kausalität) herauskristallisiert hat. Bei den meisten uns bekannten, heutigen CSCW Anwendungen wird aber meist noch auf existierenden Betriebssystem-Schnittstellen (UNIX-Sockets und Files z.Bsp.) aufgesetzt. Für anspruchsvollere CSCW Anwendungen sind diese Mechanismen sicherlich nicht mehr zielführend. Wir sind der Ansicht dass bei künftigen CSCW Anwendungen Mechanismen wie sie von Isis oder Horus (siehe Kapitel 3.7) zur Verfügung gestellt werden und vermehrt objekt-orientierte Paradigmen sowie herstellerunabhängige Standards zum Einsatz kommen müssen.

Die digitale Kommunikation ist ebenfalls ein wichtiger Aspekt des CSCW. Hier läuft die Zukunft des CSCW darauf hinaus dass vermehrt MAN- und WAN-Technologie zum Tragen kommen um global verteilte Teams in ihrer Kooperation zu unterstützen. Dort wo Echtzeit Video und Audio zum Einsatz kommt bedarf es geeigneter Kommunikations-Protokolle um die erforderlichen Bandbreiten und Antwortzeiten stabil zu halten.

Literatur

[B+90] K. P. Birman et al. *The ISIS System Manual, Version 2.1.* The ISIS Project, September 1990.

[Bey93] Torsten Beyer. Objektbörse. *IX Multiuser Multitasking Magazin*, Februar 1993.

[BG93] K. P. Birman and Bradford B. Glade. Consistent Failure Reporting in Reliable Communication Systems. Technical Report 93-1349, Cornell University, May 1993.

[Bir91] K. P. Birman. The Process Group Approach to Reliable Distributed Computing. Technical Report 91-1216, Cornell University, July 1991. To appear in Communications of the ACM.

[BJ87] K. P. Birman and T. A. Joseph. Exploiting Virtual Synchrony in Distributed Systems. *ACM Operating Systems Review*, 21(5):123–138, November 1987.

[BJ89] K. P. Birman and T. A. Joseph. Exploiting Replication in Distributed Systems. In Sape Mullender, editor, *Distributed Systems.* ACM Press, 1989.

[BR92] Dirk Bartels and Jonathan Robie. Persistent Objects and Object–Oriented Databases in C++. *C++ Report*, 4(7), September 1992.

[BSS91] K. P. Birman, A. Schiper, and P. Stephenson. Lightweight Causal and Atomic Group Multicast. *ACM Transactions on Computer Systems*, 9(3), August 1991.

[Cas92] Stephen Casner. First IETF Internet Audiocast. *ACM Computer Communication Review*, 22(3), July 1992.

[Dee89] S. Deering. Host Extensions for IP Multicasting. RFC 1112, Stanford University, August 1989.

[Dig91] Digital Equipment Corp., Hewlett-Packard Co., HyperDesk Corp., NCR Corp., Object Design Inc., SunSoft Inc. *The Common Object Request Broker: Architecture and Specification*, Revision 1.1 edition, December 1991. OMG Document Number 91.12.1.

[DR93] Prasun Dewan and John Riedl. Toward Computer-Supported Concurrent Software Engineering. *IEEE Computer*, January 1993.

[EGG91] C. Ellis, S. Gibbs, and Rein G. Groupware – Some Issues and Experiences. *Communications of the ACM*, 34(10), 1991.

[Gif79] D. K. Gifford. Weighted Voting for Replicated Data. In *Proceedings of the 7th Symposium on Operating System Principles*, December 1979.

[Gol91] Richard Andrew Golding. *Accessing Replicated Data in Large-Scale Distributed Systems.* PhD thesis, University of California, Santa Cruz, June 1991.

[Her86] Maurice. P. Herlihy. A Quorum-Consensus Replication Method for Abstract Data Types. *ACM Transactions on Computer Systems*, 4(1):32–53, February 1986.

[IEE] 1985a IEEE. Carrier Sense Multiple Access with Collision Detection.

[Keh92] Brendan P. Kehoe. Zen and the Art of the Internet, 1992.

[Kol93] Markus Kolland. GroupOS – Analyse notwendiger Funktionalitäten für Groupware Applikationen. Interner Bericht, Siemens ZFE München, May 1993.

[Kot90] Philip Kotler. Globalization – Realities and Strategies, 1990.

[KTHB] M. Frans Kaashoek, Andrew S. Tanenbaum, Susan Flynn Hummel, and Henri E. Bal. An Efficient Reliable Broadcast Protocol.

[Kur93] Jim Kurose. Open Issues and Challenges in providing Quality of Service Guarantees in High-Speed Networks. *ACM Computer Communication Review*, 23(1), January 1993.

[Lam78] Leslie Lamport. Time, Clocks and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7), July 1978.

[M+89] S. J. Mullender et al. Amoeba – A Distributed Operating System for the 1990s. Technical report, Vrije University, Amsterdam, 1989.

[Maf93] Silvano Maffeis. Object-Oriented Distributed Programming with Electra . Technical Report 93.17, CS Dept. University of Zurich, Switzerland, March 1993.

[Mul89] Sape Mullender. *Distributed Systems.* ACM Press, 1989.

[Nel81] B.J. Nelson. *Remote Procedure Call.* PhD thesis, Carnegie Mellon University, 1981. CMU-CS-81-119.

[NG92] K. Narayanaswamy and Neil Goldman. Lazy Consistency: A Basis for Cooperative Software Development. In *ACM Conference on Computer Supported Cooperative Work*, Toronto, Canada, November 1992.

[PP90] Joseph Pasquale and George Ployzos. System Support for Multimedia Applications. In *First Int. Workshop on Network and Operating System Support for Digital Audio and Video*, ICSI, Berkeley, USA, November 1990.

[RB91] Tom Rodden and Gordon Blair. CSCW and Distributed Systems: Problem and Control. In *Proceedings of the Second European Conference on Computer Supported Cooperative Work*, Amsterdam, The Netherlands, September 1991.

[Ric92] Aleta Marie Ricciardi. *The Group Membership Problem in Asynchronous Systems.* PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York, November 1992. No. 92-1313.

[S+88] M. Stefik et al. Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. In I. Greif, editor, *Computer-Supported Cooperative Work: A Book of Readings.* Morgan Kaufmann Publishers, 1988.

[SBM89] Alexander Siegel, K. P. Birman, and Keith Marzullo. Deceit: A Flexible Distributed File System. Technical Report 89–1042, Cornell University, November 1989.

[Seo90] Karen Seo. Terrestrial Wideband Network Support for Conferencing. In *First Int. Workshop on Network and Operating System Support for Digital Audio and Video*, ICSI, Berkeley, USA, November 1990.

[Tan89] Andrew S. Tanenbaum. *Computer Networks.* Prentice-Hall, second edition, 1989.

[Tan92] Andrew S. Tanenbaum. *Modern Operating Systems.* Prentice-Hall, 1992.

[TvR85] Andrew S. Tanenbaum and Robbert van Renesse. Distributed Operating Systems. *ACM Computing Surveys*, 17(4), December 1985.

[vR93] Robbert van Renesse. Causal Controversy at Le Mont St.-Michel. *ACM Operating Systems Review*, 27(2), April 1993.

[vRBC+92] Robbert van Renesse, K. P. Birman, R. Cooper, B. Glade, and P. Stephenson. Reliable Multicast between Microkernels. In *Proceedings of the USENIX Workshop of Micro-Kernels and Other Kernel Architectures*, Seattle, Washington, April 1992.