Zeitschrift: Technische Mitteilungen / Schweizerische Post-, Telefon- und Telegrafenbetriebe = Bulletin technique / Entreprise des postes, téléphones et télégraphes suisses = Bollettino tecnico / Azienda delle poste, dei telefoni e dei telegrafi svizzeri

Herausgeber: Schweizerische Post-, Telefon- und Telegrafenbetriebe

Band: 73 (1995)

Heft: 3

Artikel: Offene verteilte Systeme. Teil 1, Einführung

Autor: Zweiacker, Marc

DOI: https://doi.org/10.5169/seals-875927

Offene verteilte Systeme Teil 1: Einführung

Marc ZWEIACKER, Bern

Zusammenfassung

Offene verteilte Systeme
Teil 1: Einführung

Verteilte Systeme trifft man überall dort an, wo mehrere vernetzte Rechner miteinander kommunizieren und dadurch gemeinsam ein Problem bearbeiten. Der besondere Nutzen dieser Zusammenarbeit kann darin liegen, dass das Problem in kürzerer Zeit gelöst wird, als dies mit einem einzelnen Rechner der Fall gewesen wäre. Andere Vorteile sind die Redundanz von Funktionalitäten und Daten sowie die erhöhte Sicherheit vor physikalischer Einwirkung auf die Hardware. Zudem existieren Anwendungen, die anders gar nicht lösbar wären, wie z. B. elektronische Post oder Bulletin-Boards. Die kooperierenden Teile eines verteilten Systems, die funktionalen Einheiten, können auf unterschiedlichen Prozessorknoten aktiv sein. Die dazu notwendige Eigenschaft der Ortsunabhängigkeit von funktionalen Einheiten wird durch objektorientierte Techniken unterstützt. Objektorientierung hilft zusätzlich, die Offenheit des verteilten Systems zu gewährleisten.

Résumé

Systèmes distribués ouverts
Première partie: Introduction

Les systèmes distribués se rencontrent dans tous les cas d'intercommunication d'ordinateurs maillés appelés à résoudre des problèmes communs. Le principal avantage de cet interfonctionnement réside dans le fait que des problèmes peuvent être résolus plus vite qu'avec un seul ordinateur. D'autres avantages consistent dans la redondance de fonctions et de données ainsi que dans la sécurité accrue contre des influences physiques sur le matériel. Il existe en outre des applications qui ne pourraient être résolues d'autre manière, notamment la poste électronique ou les tableaux d'affichage. Les éléments contribuant au fonctionnement d'un système distribué, les unités fonctionnelles, peuvent être actifs sur divers nœuds de processeurs. Des techniques orientées objet soutiennent à cet effet la nécessité d'implanter en des sites différents les éléments fonctionnels. L'orientation objet assure mieux encore l'ouverture du système distribué.

Riassunto

Sistemi distribuiti aperti
1a parte: introduzione

Si parla di sistemi distribuiti quando più calcolatori interconnessi comunicano l'uno con l'altro ed elaborano insieme un problema. Questa collaborazione è di particolare utilità in quanto il tempo impiegato per risolvere il problema è inferiore a quello che verrebbe impiegato da un singolo calcolatore. Altri vantaggi sono la ridondanza delle funzioni e dei dati come pure la maggiore sicurezza contro influssi fisici sull'hardware. Esistono inoltre soluzioni che non sarebbero altrimenti applicabili, come la posta elettronica o il Bulletin-Boards. Le unità funzionali, ossia le parti cooperanti di un sistema distribuito, possono essere attive su diversi nodi di processori. La caratteristica necessaria a tal fine cioè l'indipendenza delle unità funzionali dal luogo viene supportata da tecniche orientate agli oggetti. L'orientamento agli oggetti aiuta inoltre a garantire l'apertura del sistema distribuito.

Summary

Open Distributed Systems
Part 1: Introduction

Distributed systems are to be found wherever a number of cross-linked processing units communicate with each other and thus work on a problem jointly. The main advantage of this type of cooperation is that it takes less time to solve a problem than would be the case with just a single processing unit. Other advantages are that fewer functions and data are required and that there is a greater degree of protection against physical effects on the hardware. Moreover, there are applications for which there would be no other solution — for instance electronic mail or bulletin boards. The cooperating parts of a distributed system, the functional units, can be active on different processor nodes. The local independence of functional units which this requires is supported by object-oriented technologies. Object-orientation also helps to ensure the openness of the distributed system.

Einleitung

Die rasante technologische Entwicklung der vergangenen Jahre hat dazu geführt, dass die Telekommunikation heute praktisch in alle Bereiche des täglichen Lebens einfliesst. Es ist zur Selbstverständlichkeit geworden, Dokumente in Sekundenschnelle via Fax über den Erdball zu versenden, Computeranlagen über grosse Entfernungen Daten austauschen zu lassen oder Konferenzen via Satellit abzuhalten. Der Ruf nach immer höheren Übertragungsraten von Telekommunikationsnetzen folgt logisch dem Bedürfnis, immer mehr Daten in noch kürzerer Zeit auszutauschen. Als Beispiel sei das Bildtelefon angeführt, mit dem nebst Sprache auch Bilder in Echtzeit übertragen werden sollen, Stichwort: Multimedia.

Was sind verteilte Systeme?

Der Austausch von Daten über ein Netzwerk setzt voraus, dass die beteiligten Parteien (Gesprächspartner, Fax-Geräte, Computer usw.) diesen Datenverkehr in einem gemeinsamen Kontext ansiedeln. Das bedeutet, dass ihnen der Sinn dieser Datenübertragung klar ist. Diese Tatsache ist an sich einleuchtend, hat jedoch eine wichtige Konsequenz: Es entsteht eine Kooperation, man kann auch sagen, die Parteien arbeiten gemeinsam an der Lösung eines Problems. Dies soll an einem Beispiel aus der Büroautomation erläutert werden: Die Daten, die ein PC an einen Drucker im lokalen Netzwerk sendet, definieren Inhalt und Form eines Dokumentes, das auf Papier ausgedruckt werden soll. Dies ist beiden Parteien (Drucker und PC) bekannt, wodurch ihr gemeinsamer Kontext definiert ist. Eine Vielzahl weiterer Beispiele aus den Bereichen Fabrikautomatisation, Bankenwesen, Unterhaltung usw. könnten hier angeführt werden. Ihnen wäre allen gemeinsam, dass Daten zum Zwecke der Kooperation über grosse Entfernungen ausgetauscht werden. Diese Eigenschaften beschreiben ein *verteiltes System*.

Schwerpunkte

Dem Verwendungszweck entsprechend, können verteilte Systeme klassifiziert werden. Im allgemeinen lässt sich ein bestimmtes System als Mischform der hier genannten Schwerpunkte einordnen *(Fig. 1)*.

- Rechenintensive Anwendungen in den Bereichen Architektur (CAD), Fabrikation (Echtzeit-Systeme) oder Mechanik (Finite-Elemente-Methode) erfordern hohe *Rechenleistungen*. Diese können einerseits durch schnelle Prozessoren erreicht werden. Anderseits lässt sich die Rechenbelastung auf mehrere Prozessoren eines verteilten Systems aufteilen, was die Datenverarbeitung beschleunigt.
- Daten können mehrfach auf verschiedenen Rechenanlagen abgelegt werden. Die Verteilung von Datenkopien garantiert besseren Schutz vor Datenverlust. Diese und andere Massnahmen zur *Datensicherheit* sind beispielsweise für Datenbank-Anwendungen von Bedeutung. Vertreter hiervon sind Flugreservationssysteme, Systeme für den elektronischen Zahlungsverkehr und Directory-Services (wie X.500).
- Das *Netzwerkmanagement* ist von seiner Natur her ein verteiltes System. Hier liegt das Schwergewicht auf der Beobachtung netzspezifischer Lastdaten und den davon abhängigen Massnahmen, um diese Lasten zu meistern. Das Netzwerk, welches anderswo als Voraussetzung für die Existenz eines verteilten Systems gilt, bildet hier das Hauptinteresse desselben. Das Netzwerkmanagement ist eine herausfordernde Disziplin, die nicht nur von technischem Interesse ist; Netzwerke und Netzwerkdienste sind nicht zuletzt Teil des Telecom-Geschäfts.

Nutzen und Anwendungen

Der Nutzen verteilter Systeme gegenüber herkömmlichen ist teilweise bereits angesprochen worden:

- *Verarbeitungsgeschwindigkeit*. Die Aufgabenteilung kann so vorgenommen werden, dass ein besonders schneller Rechner mit CPU-intensiven Teilproblemen versorgt wird. Zusätzlich treten durch die Verteilung Parallelitäten auf, d. h. es entstehen gleichzeitige Verarbeitungsschritte im verteilten System.
- *Zuverlässigkeit und Verfügbarkeit*. Ein verteiltes System bringt eine höhere Zuverlässigkeit mit sich, da es Daten wie auch Funktionalitäten redundant zur Verfügung stellen kann. Die Fehlfunktion einer Rechenanlage (Datenverlust, Hardware-Probleme) kann durch andere Rechner aufgefangen werden.
- *Sicherheit*. Die Verteilung von Daten und Funktionalitäten auf unterschiedliche Rechner erhöht nebst der Verfügbarkeit auch den Schutz vor physikalischer Zerstörung, da die Rechner eines verteilten Systems im allgemeinen geographisch weit genug entfernt sind, um nicht derselben Katastrophe (z. B. Feuer) zum Opfer zu fallen.

Es gibt eine Vielzahl von Anwendungen für verteilte Systeme. Jede Applikation, die Netzwerkdienste in Anspruch nimmt, kann dazugezählt werden. Die Forderung nach *offenen* Systemen kommt mit dem Bedürfnis, diese dynamisch erweiterbar zu machen. Unter Erweiterung verstehen wir den Zusammenschluss von Systemen, die unabhängig voneinander gebaut worden sind. Diese Erweiterungen haben oft nicht primär technische, sondern wirtschaftliche Gründe. Man denke an die Bankenfusionen der letzten Zeit. Wenn die beteiligten Banken ihre EDV-Systeme von einem Augenblick auf den anderen zusammenschalten wollen, so ist das nur mit offenen Systemen möglich.

Ein weiteres Beispiel sind Multimedia- und Konferenzsysteme. Sollen diese dereinst sinnvoll genutzt werden, so dürfen wir uns in keine Herstellerabhängigkeit begeben. Dem Benutzer ist es im allgemeinen gleichgültig, welches Produkt sein Gegenüber benutzt; für ihn zählt allein, dass die beiden Anlagen einwandfrei zusammenarbeiten. Das kann nur durch konsequente Durchsetzung der Systemoffenheit erreicht werden. Eine der grossen Herausforderungen für die Telekommunikation sind die *intelligenten Netze* (Intelligent Networks). Darunter versteht man die Modernisierung der öffentlichen Netze mit neuen, zusätzlichen Leistungen. Die Netze der Zukunft werden dem Kunden grösstmögliche Flexibilität bieten. Er wird beispielsweise die Kosten eines Anrufes von seiner Kreditkarte abbuchen können oder seinen Apparat irgendwo in der Welt anschliessen und davon ausgehen dürfen, dass er nach wie vor unter derselben

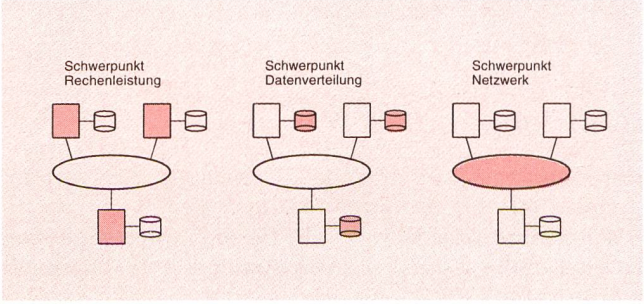


Fig. 1 Schwerpunkte verteilter Systeme

Nummer erreichbar ist. Solche und andere Dienste werden bald als verteilte Systeme realisiert sein und dem Kommunikationsmarkt noch stärkere Impulse verleihen.

Verteiltes System und Offenheit

Verteiltes System

Allgemein versteht man unter einem verteilten System die Aufteilung einer Aufgabe auf verschiedene *funktionale Einheiten*, die mittels Datenaustausch kooperieren. Den Vorgang der Aufgabenteilung nennt man *Verteilung*. Ein Problem gilt als *verteilt*, wenn es mit Hilfe eines verteilten Systems lösbar ist.

In der Informatik verstehen wir unter einem verteilten System die Gesamtheit der Hard- und Softwaremittel, die eine Verteilung von Aufgaben ermöglicht. Rechenanlagen wie Workstations, PCs usw. sind die Träger von funktionalen Einheiten. Damit diese untereinander Informationen austauschen können, werden sie über ein Netzwerk miteinander verbunden. Die Software für die Lösung eines verteilten Problems nennt man auch verteilte Anwendung oder *verteilte Applikation:*

Ein verteiltes System ist eine Konfiguration von zwei oder mehr Prozessorknoten in einem Netzwerk, die gemeinsam eine verteilte Applikation ausführen.

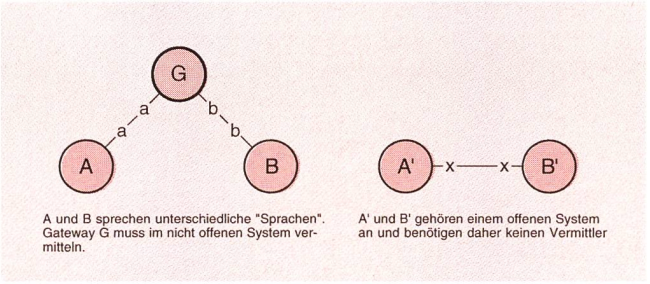
Durch die Verteilung arbeiten die Prozessorknoten weitgehend autonom und gleichzeitig an den ihnen zugewiesenen Teilproblemen; man spricht von *Parallelität.*

Offenheit

Ein verteiltes System ist in der Regel ein dynamisches, sich veränderndes System. In der Büroautomation ist es beispielsweise üblich, dass man unter verschiedenen Druckern eines Netzwerkes eine Auswahl treffen kann, was zur Folge hat, dass sich die Konfiguration von zusammenwirkenden Parteien ändert. Ebenso kann es vorkommen, dass sich zwei unterschiedliche verteilte Systeme zu einem neuen, grösseren System zusammenfinden. In beiden Fällen ist es von grosser Wichtigkeit, dass sich die neu hinzukommenden Parteien nahtlos in das bestehende System einfügen. Die Antwort auf diese Anforderung heisst *Offenheit.* Sie besagt, dass alle Parteien die gleiche Sprache sprechen müssen, damit sie sich ohne weitere Umstände in das verteilte System integrieren können. Ein verteiltes System, das dieses Kriterium erfüllt, heisst *offenes verteiltes System:*

Offenheit ist die Eigenschaft eines Systems, seine Teilsysteme über eine systemunabhängige Verständigungsbasis zu koppeln.

Eine so definierte Offenheit entspricht einer Menge von Regeln, nach denen sich die Teilsysteme zu richten haben. Diese Regeln beschreiben *System-Schnittstellen,* deren Kenntnis Voraussetzung genug ist, um eine weitere Komponente in ein bestehendes System einzugliedern. Der OSI-Protokollstack ist ein Beispiel eines offenen Systems, was sich auch in seiner Bezeichnung niederschlägt (OSI = Open Systems Interconnection). Die Protokoll-Layer sind durch klar definierte Schnittstellen beschrieben. Die Implementation der Layer selbst ist durch diese Definitionen weder vorgeschrieben noch eingeschränkt. Im Gegenteil: Offenheit ermöglicht die Beschreibung eines Systems, ohne auf Implementationsdetails einzugehen.



A und B sprechen unterschiedliche "Sprachen". Gateway G muss im nicht offenen System vermitteln.

A' und B' gehören einem offenen System an und benötigen daher keinen Vermittler

Fig. 2 Offenes und nicht offenes System

Hier treffen wir auf die enge Verwandtschaft zwischen Offenheit und *Hersteller-Unabhängigkeit:* Wer ein System baut, das nicht dem Kriterium der Offenheit entspricht, wird nur mit erheblichem Mehraufwand eine Erweiterung seines Systems mit Produkten anderer Hersteller zustande bringen. Es ist heute noch ein stark verbreitetes Phänomen in der Telekommunikation, dass viele Applikationen herstellerspezifisch sind und deshalb zusätzliche Hard- und Software benötigen, um mit Fremdsystemen einen Verbund eingehen zu können. Typische Beispiele dafür sind Formatkonverter zwischen Textverarbeitungsprogrammen und Rechner mit Vermittlerfunktionen zwischen Systemen unterschiedlicher Hersteller (sog. Gateways). *Figur 2* illustriert den Unterschied zwischen einem System, das aus herstellerspezifischen Komponenten aufgebaut ist, und einem offenen System. Im ersten Fall kooperieren die Teilsysteme A und B mit Hilfe eines Vermittlers G, der die Rolle eines Übersetzers (zwischen A und B) einnimmt. Demgegenüber bilden die Systeme A' und B' zusammen ein offenes System, d. h. sie benötigen zu ihrer Interaktion keinen weiteren Aufwand, weil die gemeinsame Verständigungsbasis von Anfang an berücksichtigt worden ist (Kleinbuchstaben in Fig. 2 bezeichnen Verständigungsbasen).

Bezogen auf die verteilten Systeme bedeutet Offenheit u. a., dass sich die funktionalen Einheiten an bestimmte Regeln halten müssen, was ihre Beziehung untereinander betrifft. Sind diese Regeln bekannt, so kann das verteilte System erweitert werden oder mit anderen verteilten Systemen zusammenarbeiten, ohne dass diese beiden Erweiterungsvarianten Ausnahmecharakter bekommen. Schliesslich sind die bestehenden Teilsysteme nach genau denselben Regeln aufgebaut worden, wie die neu hinzugekommenen. Dies verdeutlicht ein Hauptmerkmal der Offenheit: Sie muss von Anfang an integraler Bestandteil eines Systems sein. Offenheit lässt sich nicht nachträglich in ein System einbringen.

Ohne Offenheit sind Integrationen verschiedener Systeme nicht oder nur schwer möglich. Dies wird in Zukunft vermehrt ein gewichtiges Kriterium in der Beurteilung von Software-Lösungen für die Telekom-

munikation sein. Offene Systeme werden die Konkurrenz fördern. Sie werden die Hürden der Hersteller-Abhängigkeit überwinden und dem Kunden eine grössere Freiheit bei der Wahl seines Dienstanbieters geben.

Objektorientierte Techniken

Die Prinzipien der *Kapselung* und *Abstraktion* machen objektorientierte Techniken zu einem unabdingbaren Hilfsmittel für die Analyse und den Bau verteilter Applikationen.

Unterstützung der Ortsunabhängigkeit

Objektorientierte Techniken erleichtern die Modellierung eines Problems in funktionalen Einheiten, was bereits im Frühstadium einer Entwicklung zu klaren Systemstrukturen führt. Damit wird die Komplexität eines Systems auf eine Menge von Objekten und ihre Schnittstellen reduziert. Die Zerlegung in Teilprobleme ist ein weitgehend kreativer Prozess und kann in der Regel nicht auf Korrektheit geprüft werden. Der Analytiker kann sich zuweilen an Rezepte halten, die ihn bei dieser Arbeit unterstützen.

Für eine verteilte Applikation ist das Zusammenwirken von Objekten, d. h. ihr Verhalten und ihre Schnittstellen, von Bedeutung. Diese Eigenschaften sind in objektorientierten Sprachen durch das Konzept der *Objektklassen* unterstützt.

Nun kommt die wahre Stärke des objektorientierten Ansatzes zum Tragen: Die Objekte werden als unabhängige Einheiten programmiert und repräsentieren schliesslich die funktionalen Einheiten. Der Informationsaustausch zwischen funktionalen Einheiten wird zum Datenaustausch zwischen Objektschnittstellen und erfolgt in der Regel über das Netzwerk. Darum ist es für die Objekte nicht von Belang, an welcher Stelle im Raum sie sich befinden, da sie von überall auf dem Netz erreichbar sind. Dies ist ein grundlegendes Merkmal objektorientierter Techniken: Sie unterstützen die *Ortsunabhängigkeit* der funktionalen Einheiten eines verteilten Systems. Man kann auch sagen: Es ist nicht nötig, eine Abbildung von funktionalen Einheiten auf die Prozessorknoten nach einem starren Muster festzulegen: Die *Objektkonfiguration* (die Allozierung von Objekten auf Prozessorknoten) kann *dynamisch* vorgenommen werden.

Die Ortsunabhängigkeit der Objekte ermöglicht eine beliebige Abbildung derselben auf die Prozessorknoten *(Fig. 3)*. Es ist einerseits möglich, dass alle Objekte auf demselben Knoten ausgeführt werden; aus der verteilten Applikation wird eine zentralisierte. Anderseits kann jedes Objekt auf einem anderen Prozessorknoten «beheimatet» sein. Diese Palette von möglichen Allozierungen ist typisch für ein objektorientiertes Design. Der Informationsaustausch zwischen Objekten wird, abhängig von der Allozierung, entweder lokal, d. h. innerhalb eines Prozessorknotens, oder über das Netzwerk realisiert. Es existieren aber Betriebssysteme für den Bau und Betrieb verteilter Applikationen, die diesen Unterschied in der Objektkonfiguration bemerken und automatisch entsprechende Massnahmen für die Realisierung des Datenverkehrs treffen.

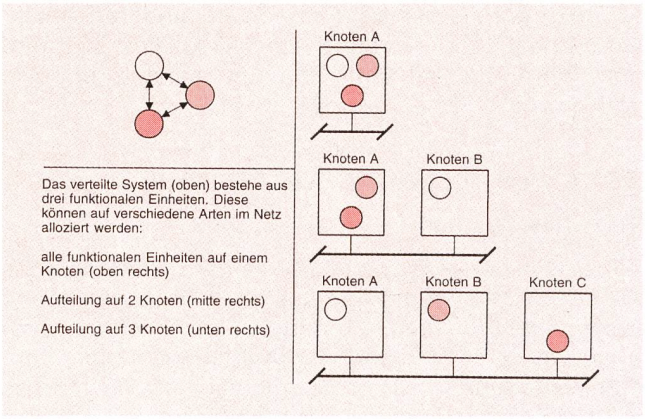


Fig. 3 Ortsunabhängigkeit von funktionalen Einheiten

Die Tatsache, dass funktionale Einheiten nicht ortsgebunden sind, ermöglicht erst die Schaffung der funktionalen Redundanz. Fällt eine funktionale Einheit auf einem Prozessorknoten aus, so kann eine Ersatzfunktion auf einem anderen Knoten im Netz aktiviert werden.

Der Gebrauch der Objektorientierung bedeutet, dass Objektschnittstellen unabhängig von der Applikation und der Programmiersprache beschrieben werden, womit gleichzeitig die Offenheit des Systems verwirklicht wird. Jedes Objekt der *offenen verteilten Applikation* versteht diese Beschreibung von Schnittstellen und kann sich dadurch ohne weiteren Aufwand in das System integrieren, da es die gemeinsame Verständigungsbasis kennt, die für die Offenheit unerlässlich ist.

Client-Server-Prinzip

Um eine verteilte Applikation erstellen zu können, müssen die funktionalen Einheiten (sie repräsentieren die Teilaufgaben) auf Objekte abgebildet werden. Diese nehmen, ihrer zugewiesenen Aufgabe entsprechend, eine bestimmte Rolle im Gesamtsystem ein. Wird ein Objekt von anderen Objekten für die Ausführung eines Dienstes angefragt, so verhält sich dieses als Dienstanbieter, und man spricht von *Server* oder *Server-Objekt*. Die anfragenden Objekte nehmen die Rolle eines Kunden ein, daher der Ausdruck *Client* oder *Client-Objekt*.

Jedes Objekt kann sowohl als Server wie auch als Client agieren, d. h. ein Server-Objekt kann selber andere Server-Objekte um deren Dienste anfragen und schlüpft damit in die Rolle des Client. Server und Client sind nicht starre Verhaltensmuster von Objekten, sondern Rollen, die dynamisch, nach Bedarf, eingenommen werden. Dies ist das Hauptmerkmal des *Client-Server-Prinzips*, welches übrigens nicht auf objektorientierte Systeme beschränkt ist.

Verteilte Plattform

Um eine verteilte Applikation überhaupt erst zu ermöglichen, müssen Objekte über betriebssystemnahe Sonderfunktionen verfügen. Diese werden zusammengefasst und als Spezial-Software auf den Rechnern installiert, womit ein erweitertes Betriebssystem, genannt *verteilte Plattform*, entsteht.

Sonderfunktionen von Objekten

Die Objekte einer verteilten Applikation verfügen über zwei grundlegende Eigenschaften. Zum einen sind es in sich abgeschlossene Einheiten, welche die funktionalen Einheiten eines Systems repräsentieren. Zum anderen müssen sie zusätzliche Funktionen erfüllen können, die von der Art der verteilten Applikation *unabhängig* sind. Dazu gehören u. a.:

- *Kommunikation.* Sie ist die Voraussetzung für die Kooperation unter den funktionalen Einheiten des Systems.
- *Synchronisations-Mechanismen.* Darunter fallen alle Funktionen, die nötig sind, um fehlerhafte Systemzustände zu verhindern (z. B. Deadlocks). Sobald mehrere Objekte auf dieselben Daten zugreifen können, müssen Massnahmen ergriffen werden, die eine Datenkollision verhindern. Ohne derartige Mechanismen sind zuverlässige verteilte Applikationen nicht realisierbar.
- *Transparenzfunktionen.* Diese hängen mit der Ortsunabhängigkeit eines Objektes und mit der Fehlertoleranz des verteilten Systems zusammen. Dank der sogenannten Verschiebungs-Transparenz (Relocation Transparency) können beispielsweise Server-Objekte zur Laufzeit durch andere Server ersetzt werden, ohne dass die Applikation aus Sicht des Benutzers davon Kenntnis erhält.
- *Anschluss an eine Objekt-Datenbank.* Server-Objekte müssen die Gelegenheit bekommen, ihren Dienst im Netz anzubieten, was mittels einer systemweiten Datenbank realisiert wird. Diese unterstützt das sogenannte *Trading*, d. h. die Vermittlung von Diensten eines Servers an potentielle Clients.

Diese Sonderfunktionen werden von allen Objekten eines verteilten Systems benötigt. Anstatt sie in jedes Objekt einzeln zu integrieren, fasst man sie zu sogenannten *Infrastruktur-Diensten* zusammen, die als Software auf jedem Prozessorknoten installiert werden. Dadurch wird ein erweitertes Betriebssystem erzeugt, das mit der Natur der Verteilung von Objekten umgehen kann und den Anwendungsprogrammierer mit Infrastruktur-Diensten versorgt; man spricht von einer *verteilten Plattform*. Der Programmierer kann sich allein auf die applikationsspezifische Funktion eines Objektes beschränken, was letztlich hilft, die Fehlerquote der Applikation zu verringern. Die verteilte Plattform befindet sich somit zwischen dem rechnereigenen Betriebssystem und den Objekten einer verteilten Applikation, weshalb in der Literatur oft der Begriff *Middleware* auftaucht *(Fig. 4)*.

Das Netzwerk als Randbedingung

Die Schwerpunkte verteilter Systeme mögen je nach Applikation unterschiedlich sein (Fig. 1), eines ist ihnen jedoch gemeinsam: Sie sind von einem Netzwerk abhängig. Die Netzwerk-Technologie ist entscheidend für die Effizienz einer verteilten Applikation.

Liegt das Schwergewicht auf der Rechenleistung der Prozessoren, so muss ein entsprechend bemessenes Netzwerk diese Kooperation unterstützen. Man denke sich eine Applikation, die aufwendige Berechnungen an Bilddaten vornimmt. Wenn die benötigte Transferzeit der Bilddaten durch das Netzwerk in die Grössenordnung der Berechnungszeit der Bilder gerät, dann wird aus dem Netzwerk ein «Flaschenhals» für die Applikation. Dasselbe gilt für die Verteilung von Daten: Eine Video-Datenbank kann nur dann sinnvoll betrieben werden, wenn die Transferzeit in für die Kundschaft akzeptablen Grenzen liegt.

Man spricht in diesem Zusammenhang oft von Quality of Service (QoS, Dienstqualität). Eine Definition dieses Begriffs ist ohne Kenntnis der entsprechenden verteilten Applikation schwierig. Eine Umschreibung könnte lauten:

Dienstqualität (QoS) ist eine Menge von Kriterien, deren Beobachtung dazu dient, die Zufriedenheit eines Kunden bezüglich eines Dienstes beurteilen zu können.

Diese Erklärung zielt auf die *Ansprüche* an ein System und hat allgemeingültigen Charakter. Sind die Kriterien der Kundschaft einmal bekannt, so kann die Menge der QoS-Elemente genauer spezifiziert werden, ebenso die Zusammenhänge unter denselben.

Hier tritt das *Netzwerk* oft in den Vordergrund. Die Effizienz einer verteilten Applikation ist eines der Hauptkriterien für eine hohe Dienstqualität. Diese wiederum ist stark von Netzwerk-Kennwerten wie Übertragungsrate, Bitfehlerrate usw. abhängig. Das Netzwerk kann also den Ablauf einer Applikation beeinflussen und beispielsweise den Gebrauch niedrigauflösender Grafiken erzwingen, weil nur diese in «vernünftiger» Zeit übertragbar sind.

Der Bau einer verteilten Applikation darf nicht ohne Berücksichtigung der verwendeten Technologien vorgenommen werden. Das gilt besonders im Hinblick auf Netzwerke.

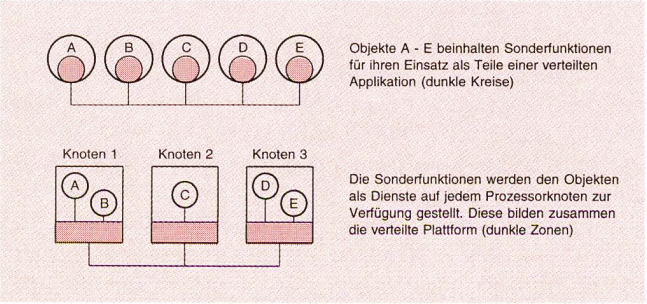


Fig. 4 Verteilte Plattform