WissenHeute Jg. 58 3/2005

Besonderheiten der Anwendungsentwicklung für die mobile Nutzung – ein Leitfaden, Teil 3.

Die Autoren

Dipl.-Ing. Stefanus Römer arbeitet als Produktmanager bei T-Mobile International, wo er insbesondere für das Produkt Mobile IP VPN sowie mobile Intranet-Access-Lösungen zuständig ist.

Dipl.-Ing. Marcus Freitag arbeitet als Senior Consultant in der DIALOGS Software GmbH in München.

Der in Heft 11/2004 begonnene Beitrag wird mit dem dritten Teil zum Systemdesign fortgesetzt. Unter Systemdesign versteht man die Unterteilung der Anwendung in einzelne Komponenten, die Festlegung der Interaktion sowie der Abläufe zwischen diesen Komponenten. Das Systemdesign legt somit die Architektur fest, die als Grundlage für die Implementierung dient. Spätere Änderungen an diesem Entwurf sind in der Regel mit hohem Aufwand verbunden. Der Designphase kommt daher besondere Bedeutung zu. In Anlehnung an den Application Configuration & Developer Guide (ACDG)[18] der T-Mobile werden nachfolgend Empfehlungen für das Systemdesign mobiler Anwendungen beschrieben.

Einleitung

In der Designphase werden Architektur und Abläufe der Anwendung bis ins Detail spezifiziert. Das Ergebnis dieser Phase ist ein Systementwurf, der den Entwicklern bei der Implementierung als Bauplan dient. Dieses stellt allerdings nicht immer einen vollständigen Entwurf dar. Meist ist der Grund dafür bereits in der Analysephase zu finden, denn oft sind die Anforderungen unscharf und unvollständig formuliert. Ein Mangel an Information und technischem Wissen führt in vielen Projekten dazu, dass erst während

[18] Der ACDG der T-Mobile bietet detaillierte Informationen und Empfehlungen sowie viele praxisbezogene Beispiele und kann unter www.t-mobile.de/entwickler kostenlos angefordert werden.

Das Thema im Überblick

Software-Entwicklung ist ein komplexes Thema. Besonders für die Nutzung im Mobilfunkbereich besteht derzeit ein großer Bedarf. Neue Projekte werden häufig unter hohem Zeit- und Kostendruck ins Leben gerufen, wodurch sich im Projektverlauf oft Risikofaktoren zeigen, die in der Anfangsphase unberücksichtigt geblieben sind. Fehler, die in der Analysephase einer Software-Entwicklung gemacht werden, wirken sich in der Designphase meist negativ aus. Im Beitrag werden Empfehlungen für die Designphase beschrieben, um eine stabile Systemarchitektur zu entwerfen, die die Grundlage der Implementierung bildet.

der Implementierungs- oder der Testphase erkannt wird, dass wichtige Anforderungen im Entwurf nicht berücksichtigt worden sind.

Bei der Software-Entwicklung wird deshalb immer häufiger vom strikten Phasenmodell abgewichen, bei dem jede Phase vollständig abgeschlossen wird und ein endgültiges Ergebnis besitzt (s. Bild 1). Stattdessen wird ein inkrementeller[19] iterativer[20] Entwicklungsprozess eingesetzt, bei dem alle Phasen mehrere Male in so genannten Stufen durchlaufen werden (s. Bild 2). Jede Stufe erweitert den Funktionsumfang der Software inkrementell. Ergebnis eines jeden Zyklus ist somit jeweils eine Softwareversion, die mit jedem Inkrement immer mehr den Gesamtanforderungen entspricht, bis die endgültige Version den Funktionsumfang schließlich vollständig abdeckt und vom Kunden akzeptiert wird. Diese Vorgehensweise berücksichtigt, dass sich Anforderungen im Projektverlauf ändern oder neu hinzukommen können und dass der Kunde oftmals zu Projektbeginn selbst keine ganz klare Vorstellung von dem hat, was er benötigt.

Bei der inkrementellen Entwicklung liegt es in Projekten oft nahe, die Phasen willkürlich zu verkürzen. Um möglichst schnell ein Ergebnis vorweisen zu können, bleiben Anforderungen zunächst unberücksichtigt, die schon in der Analyse- und Designphase hätten ermittelt werden können. Stattdessen wird alles, was die Realisierung verkomplizieren könnte, auf eine nachgelagerte Stufe verschoben. Dabei verlängert jeder zusätzliche Zyklus von Analyse, Design, Implementierung und abschließendem Test die Projektlaufzeit deutlich stärker, als eine gründliche Analyse- und Designphase.

Ist die Architektur einmal festgelegt und eine erste Softwareversion erstellt, so sind nachträgliche, tief greifende Änderungen nur schwer oder gar nicht mehr möglich. Die Konsequenz daraus könnte sein: Die erste Version wird als „Prototyp“ bezeichnet und nicht mehr weiterverwendet. Die in diese Version eingeflossene Arbeit kann man in einigen Fällen mit dem „Sammeln von Erfahrungen“ rechtfertigen, doch der Preis hierfür ist hoch.

Bei der Entwicklung mobiler Lösungen tritt dieses Problem leicht auf, denn die mobile Datenübertragung erfordert ein spezielles Anwendungsdesign. Berücksichtigt man dies nicht schon zu Beginn ausreichend, so endet die erste Version mit hoher Wahrscheinlichkeit als unbrauchbarer Prototyp. Die folgenden Abschnitte verdeutlichen, was beim Entwurf einer mobilen Lösung wichtig ist, um dies zu vermeiden und stattdessen eine tragfähige Architektur zu entwerfen.

Architektur mobiler Anwendungen

Eine universelle Architektur für eine mobile Lösung lässt sich nur schwer festlegen. Mobile Kommunikation ist nur ein Baustein von vielen in einem Projekt. Es wäre sicher falsch, sich beim Design zunächst ausschließlich auf die Kommunikation zu konzentrieren und dabei die eigentliche Applikationsarchitektur zu vernachlässigen. Vielmehr sollten die Anregungen aus den nächsten Abschnitten in den eigenen Software-Entwurf einfließen. Ob alle Aufgaben von einer Komponente übernommen werden, ob die Anwendung auch für die Kommunikation zuständig ist, oder ob eine Aufteilung in mehrere Prozesse empfehlenswert ist, hängt von der gesamten Systemarchitektur ab. Der hier verwendete Komponentenbegriff lässt diesen Spielraum zu und soll lediglich die Aufgaben in einer Einheit umfassen. Zunächst werden die Aufgabenbereiche einer Anwendung definiert. Aus diesen werden die Komponenten abgeleitet und die spezifischen Eigenschaften bestimmt. Die hier vorgestellten Komponenten dienen im Folgenden der Zuordnung weiterer Themen.

Komponenten

Aus den bisher ermittelten Anforderungen lassen sich drei Aufgabenbereiche identifizieren, die in jeder mobilen Anwendung auftreten. Zunächst gibt es die eigentliche Anwendung, die bei der Software-Entwicklung die größte Rolle spielt. Dahinter verbergen sich unter anderem

- die Benutzeroberfläche,
- die gesamte Anwendungslogik und
- das Datenmodell.

Die Anwendung nutzt zur Übermittlung der Daten die Kommunikation. Unter Kommunikation werden sämtliche Abläufe zwischen den Kommunikationspartnern verstanden. Der Aufgabenbereich Kommunikation ermöglicht zum Beispiel

- die An- und Abmeldeprozedur,
- die Durchführung von Datenübertragungen und
- die Transformation der Daten (z. B. durch Verschlüsselung).

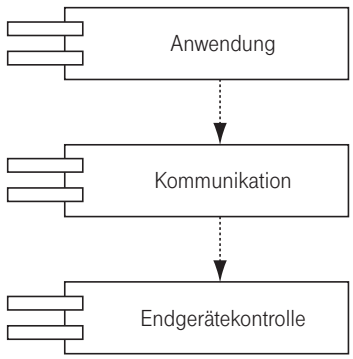
Der dritte und letzte Aufgabenbereich ist die Endgerätekontrolle. Eine Kontrolle des Endgerätes ist ein eher untypischer Aufgabenbereich und für viele Software-Entwickler Neuland. Unter Kontrolle wird hier Überwachung und Steuerung verstanden: Die Endgeräteüberwachung ist notwendig, um Kommunikationsprobleme schnell isolieren zu können. Die Endgerätesteuerung wiederum ermög-

[19] **inkrementell:** (Inkrement=Zuwachs) schrittweise um einen bestimmten Betrag zunehmend.

[20] **iterativ:** sich schrittweise in wiederholten Rechengängen der exakten Lösung annähernd.

licht der Anwendung, Probleme ohne Eingriff des Benutzers automatisch zu beheben. Die Endgerätekontrolle kommt normalerweise nur bei der Client-Komponente einer Lösung zum Einsatz. Aus den definierten Aufgaben lassen sich direkt die notwendigen Komponenten für die Architektur ableiten. Um den Sachverhalt zu veranschaulichen, werden für die Komponenten dieselben Bezeichnungen gewählt wie für die Aufgabenbereiche. Es ergibt sich ein Komponentenmodell wie es in Bild 12 dargestellt ist.

Bild 12 **Komponenten und Abhängigkeiten**



Die Ähnlichkeit diese Modells zum OSI-Modell ist offensichtlich: Die Komponenten bauen wie die Kommunikationsschichten des OSI-Modells aufeinander auf. Jedoch werden in der vorgestellten Architektur keine Kommunikationsschichten dargestellt, sondern Komponenten, die bestimmte Kommunikationsebenen verwenden oder kontrollieren. Ergänzt man das Komponentenmodell um die OSI-Schichten, so ergibt sich eine Zuordnung wie sie Bild 13 zeigt.

Anwendung

Die Anwendungskomponente realisiert nach dem OSI-Modell die drei oberen Schichten Anwendungs-, Darstellungs- und Sitzungsschicht. Nicht immer ist es sinnvoll, alle Protokolle für alle Schichten zu implementieren, so wie es im OSI-Modell spezifiziert ist. Bei einer mobilen Anwendung spielt die Sitzungsschicht jedoch eine bedeutende Rolle. Sie wird deshalb ausführlich erläutert.

Kommunikation

Die Kommunikationskomponente verwendet für ihre Aufgaben die Transportschicht, üblicherweise kommt hier das TCP-Protokoll zum Einsatz. Die Abläufe in den darunter liegenden Schichten müssen durch die Anwendung nicht berücksichtigt werden. Hardware-Treiber und Betriebssystem der Client-Plattform nehmen dem Entwickler diese Arbeit ab. Die Kommunikationskomponente ist zuständig für die Aufbereitung des Datenstroms und für die mobile Übertragung sowie deren Kontrolle. Die Kommunikationskomponente ist in der Lage, Verbindungsfehler auf ihrer Protokollebene zu erkennen, beispielsweise typische TCP-Fehler. Diese werden beispielsweise beim Einsatz von GPRS nicht nur durch Funkstörungen verursacht, sondern können sich auch aus Fehlfunktionen von Routern oder falsch konfigurierten Firewalls ergeben.

Endgerätekontrolle

Die Endgerätekontrolle überwacht die Funktion der unteren drei Kommunikationsschichten: Vermittlungs-, Sicherungs- und Bitübertragungsschicht. Sie prüft unter anderem, ob das Endgerät funktioniert, ob es im Funknetz eingebucht ist, welche Feldstärke vorliegt oder ob ihm eine IP-Adresse zugewiesen wurde. Die Kommunikationskomponente verwendet die Endgerätekontrolle, um frühzeitig über mögliche Fehler auf ihrer Protokollebene informiert zu sein.

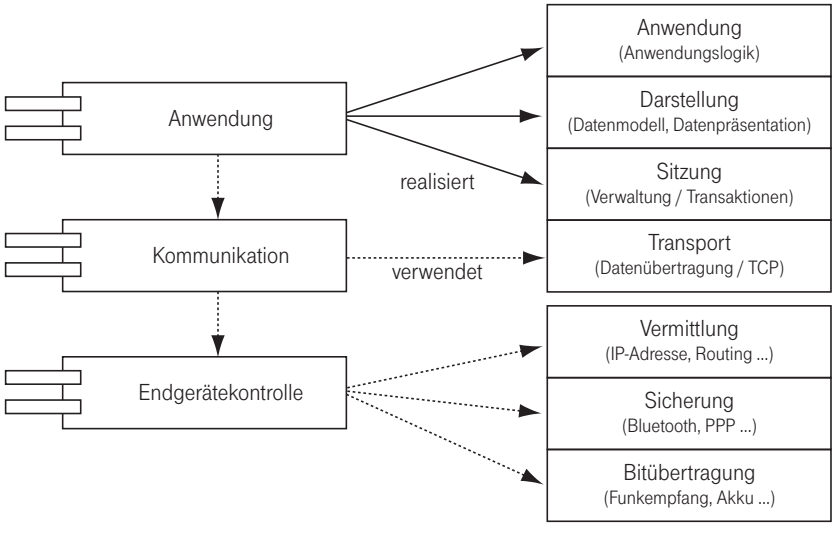
Nachdem die Zuständigkeiten der Komponenten festgelegt wurden, ist zu definieren, wie die Komponenten in einer mobilen Anwendung zusammenarbeiten sollen. Dabei sind folgende Prinzipien zu beachten:

- Hierarchie
- Unabhängigkeit
- Wiederverwendbarkeit

Hierarchie

Wie das Modell (s. Bild 12) zeigt, besteht eine Hierarchie zwischen den Komponenten. Dies bedeutet, dass jede Komponente nur die Dienste der direkt untergeordneten Komponente verwenden darf. Jede Komponente verfügt über eine definierte Schnittstelle, über die sie angesprochen werden kann. Sie kennt zudem die Schnittstelle der ihr untergeordneten Komponente. Der Kontrollfluss durchläuft also die Komponenten von der Anwendung über die Kommunikation bis in die Endgerätekontrolle. Eine klare Hierarchie erlaubt eine effektive Fehlerbehandlung in der Anwendung und eine klare Trennung zwischen den Zuständigkeiten der Kompo-

Bild 13 **Komponenten und Schichten im OSI-Modell**



IP Internet Protocol
OSI Open Systems Interconnection
PPP Point-to-Point Protocol
TCP Transmission Control Protocol

nenten. Im Idealfall ist jede Komponente unabhängig.

Unabhängigkeit
Jede Komponente ist so zu entwickeln, dass sie trotz eines Ausfalls der untergeordneten Komponente stabil arbeiten kann. Die Komponenten benötigen deshalb ein hohes Maß an Unabhängigkeit. Tritt im System ein Fehler auf, so hat die Komponente, die den Fehler entdeckt, die Aufgabe, diesen Fehler von der übergeordneten Komponente fernzuhalten und selbstständig zu beheben. Wenn die Endgerätekontrolle einen Fehler signalisiert, darf dies nicht dazu führen, dass die gesamte Kommunikation zusammenbricht. Deshalb muss die Stabilität der Kommunikationskomponente unabhängig von der Funktionsfähigkeit der Endgeräteüberwachung sein. Kann eine Komponente den Fehler nicht beheben, leitet sie diesen an die nächsthöhere Komponente weiter, die wiederum unabhängig davon versucht, eine Lösung zu finden.

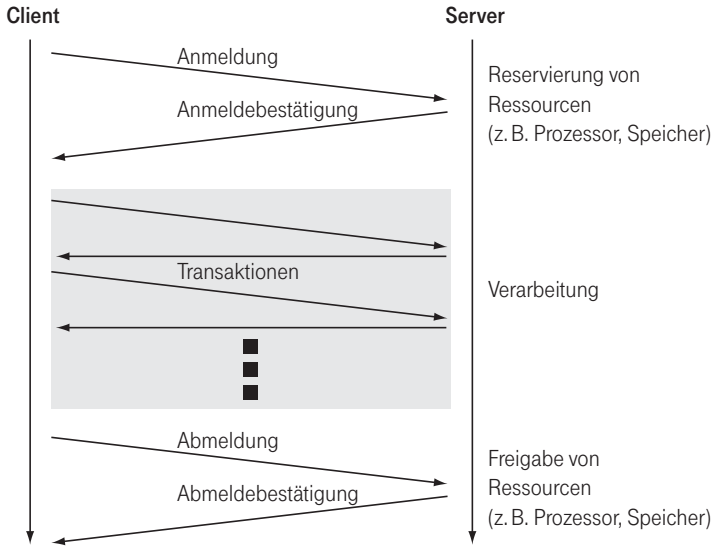
Wiederverwendbarkeit
Eine klare Hierarchie und Unabhängigkeit fördert die Wiederverwendbarkeit der Komponenten. Diese Eigenschaft ist nicht nur bei späteren Projekten von Bedeutung, sondern schon bei der Entwicklung der ersten Lösung wichtig. So ist es beispielsweise sinnvoll, die Kommunikationskomponente so zu entwickeln, dass sie sowohl auf dem Client als auch auf dem Server eingesetzt werden kann. Idealerweise nutzt der Server dieselben Komponenten wie der Client. So lässt sich nicht nur der Entwicklungsaufwand verringern, sondern auch die Fehlersuche deutlich vereinfachen.

Sitzungsmanagement

Das Sitzungsmanagement ist die Aufgabe der Anwendungskomponente. Es bildet die Schnittstelle zur Kommunikationskomponente und verwendet diese, um Daten zu übertragen. Ein „robustes“ Sitzungsmanagement stellt eine stabile Anwendung sicher. Je besser die Fehlerbehandlung und die Intelligenz des Sitzungsmanagements sind, desto unabhängiger ist die Anwendungskomponente von der Kommunikationskomponente. Im ACDG der T-Mobile wird, basierend auf einem realen Szenario, ein vollständiges Sitzungsmanagement beschrieben, das die besonderen Anforderungen an eine mobile Anwendung berücksichtigt.

Im Folgenden wird der Begriff des Sitzungsmanagements anhand von allgemeinen Definitionen vorgestellt und mit Hilfe eines alltäglichen Beispiels veranschaulicht. Abschließend werden spezielle Anforderungen an ein mobiles Sitzungsmanagements beschrieben. Eine detaillierte Modellierung ist im ACDG der T-Mobile zu finden.

Bild 14 Ablauf einer Sitzung



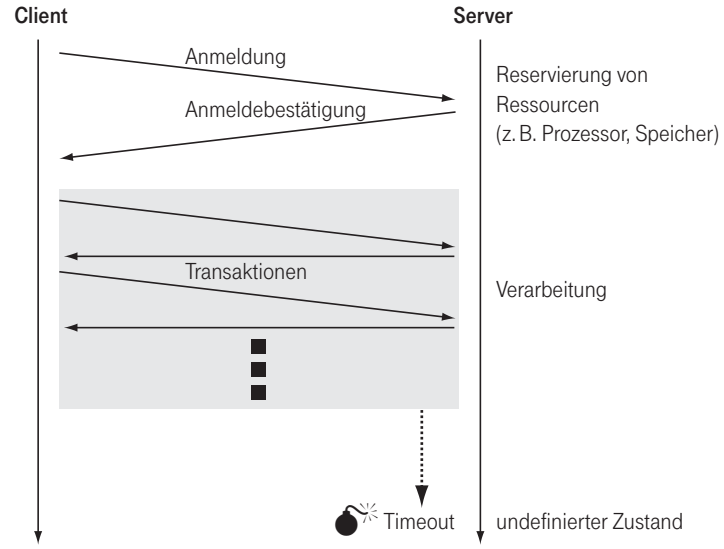
Sitzung
Eine Sitzung (Session) beschreibt eine zeitlich begrenzte Beziehung zwischen einer Client- und einer Server-Komponente. Eine Sitzung beginnt mit der Anmeldung des Clients beim Server. Der Server reserviert für den Client Ressourcen, die er erst nach der Abmeldung wieder freigibt. Der Ablauf einer Sitzung ist in Bild 14 dargestellt. Damit eine Sitzung durchführbar ist, benötigen Client und Server die Möglichkeit miteinander zu kommunizieren. Das bedeutet, dass sie eine physikalische Verbindung und eine gemeinsame Sprache brauchen, die als Protokoll bezeichnet wird. Dieses Protokoll wird als Sitzungsprotokoll bezeichnet.

Transaktion
Während einer Sitzung kann der Client die Dienste des Servers nutzen. Die Nutzung eines Dienstes wird als Transaktion bezeichnet. Eine Transaktion besteht aus einer Anfrage und einer Antwort. Im Allgemeinen wird eine Transaktionsanfrage durch den Client gestellt. Dieser übermittelt bei der Anfrage Daten, die der Server zur Bearbeitung verwendet. Für die Bearbeitung der Transaktion werden Ressourcen (z. B. Arbeitsspeicher, Prozessorzeit) auf dem Server genutzt. Ist die Bearbeitung abgeschlossen, so erstellt der Server eine Transaktionsantwort und sendet diese an den Client. Um eine Transaktion durchzuführen, wird ein gemeinsames Protokoll verwendet, das festlegt, wie im Falle einer Störung auf Transaktionsebene zu verfahren ist. Eine Transaktion ist gewissermaßen eine Sitzung in der Sitzung.

Beispiel für eine Sitzung
Ein Kunde besucht seine Bank, um einige Überweisungen zu tätigen und Bargeld abzuholen. Er betritt die Bank und meldet sich an einem Schalter bei einem Kundenberater. Der Kundenberater begrüßt den Kunden und bedient ihn.

Der Kunde übergibt dem Kundenberater drei Überweisungsträger. Der Berater nimmt die

Bild 15 Offene Sitzung



Überweisungen entgegen und händigt dem Kunden die Quittungen aus. Im Anschluss daran hebt der Kunde Bargeld vom Konto ab. Kurz darauf verabschiedet sich der Kunde beim Kundenberater und verlässt die Bank. Übertragen auf die Begriffe des Sitzungsmanagements bedeutet das:

Der Client (Kunde) besitzt eine physikalische Verbindung (persönlicher Besuch) zum Server (Bank). Über ein gemeinsames Protokoll (menschliche Sprache) meldet er sich für eine Sitzung (Bedienung durch Kundenberater) beim Server an. Der Server reserviert Ressourcen für die Sitzung (Kundenberater). Die Sitzung beginnt.

Der Client (Kunde) nutzt einen Dienst (Überweisung) des Servers (Bank). Dazu beginnt er im Rahmen der Sitzung eine Transaktion, in dem er die Transaktionsanfrage übermittelt (Übergabe eines Überweisungsträgers). Die Transaktion wird durch die Rückgabe der Transaktionsantwort (Quittung) abgeschlossen. Bei dieser Transaktion wird ein spezielles Protokoll (Aufbau der Überweisungsträger) verwendet, welches vorher vom Server (Bank) definiert wurde. Anschließend nutzt der Client (Kunde) einen weiteren Dienst (Bargeldauszahlung) des Servers (Bank). Er beginnt eine neue Transaktion (Kunde teilt dem Berater den gewünschten Betrag mit), die mit der Transaktionsantwort (Bargeldauszahlung) abgeschlossen wird. Der Client meldet sich beim Server ab (verabschiedet sich vom Kundenberater) und die verwendeten Ressourcen stehen dem Server für neue Clients wieder zur Verfügung (Kundenberater wartet auf den nächsten Kunden).

Spezielle Anforderungen an ein mobiles Sitzungsmanagement

Bei dem einfachen Szenario eines Bankbesuches sind Unterbrechungen der Sitzung eher unwahrscheinlich. Ein Bankräuber könnte zwar die Kundenbetreuung stören oder ein Feuer könnte ausbrechen, doch im Normalfall ist die Kommunikation zwischen dem Kunden und seinem Kundenberater ungestört.

Möchte man ein Sitzungsmanagement für den mobilen Einsatz entwerfen, so muss es mit einer Unterbrechung der Verbindung umgehen können, weil Störungen des vorgesehenen Ablaufs sehr viel wahrscheinlicher sind.

Eine Verbindungsunterbrechung kann unterschiedliche Auswirkungen auf das Verhalten einer Client-/Server-Anwendung haben. Im Folgenden sind einige typische Problemsituationen beispielhaft aufgezeigt, die bei der Modellierung eines mobilen Sitzungsmanagements zu berücksichtigen sind.

Offene Sitzungen

Wird eine Sitzung ohne korrekte Abmeldung abgebrochen, so erkennt der Session Manager des Servers kein definiertes Ende. Die Sitzung bleibt dann weiterhin geöffnet und der Server wartet auf Daten (Bild 15). Dies kostet unnötig Systemressourcen des Servers. Werden diese Ressourcen nicht regelmäßig freigegeben (beispielsweise nach einem gewissen Zeitintervall), so sammeln sich die offenen Sitzungen und blockieren schließlich das System. Offene Sitzungen bilden zudem Angriffspunkte für unerwünschte Eindringlinge, die eine bestehende authentifizierte Verbindung ausnutzen. Um diesen Fall zu verhindern, lässt sich die Kommunikation zwischen Client und Server durch Verschlüsselungsmechanismen sichern.

Fehlermeldungen und Verzögerungen

Die Ausführung von Befehlen ist in den meisten Anwendungen mit einer zeitlichen Begrenzung (auch „Timeout“ genannt) kombiniert. Wird eine Anfrage nach diesem Zeitpunkt nicht beantwortet, so bricht die Anwendung mit einer Fehlermeldung ab. Je nach Konfiguration kann es sich dabei um ein relativ großes Zeitintervall handeln. So kann z. B. die Anmeldung erst nach mehreren Minuten mit einem Fehler abgebrochen werden, obwohl schon früher feststeht, dass keine Verbindung mehr besteht. Ein solches Verhalten ist oft bei Anwendungen anzutreffen, die für den Einsatz in lokalen Netzwerken gedacht sind.

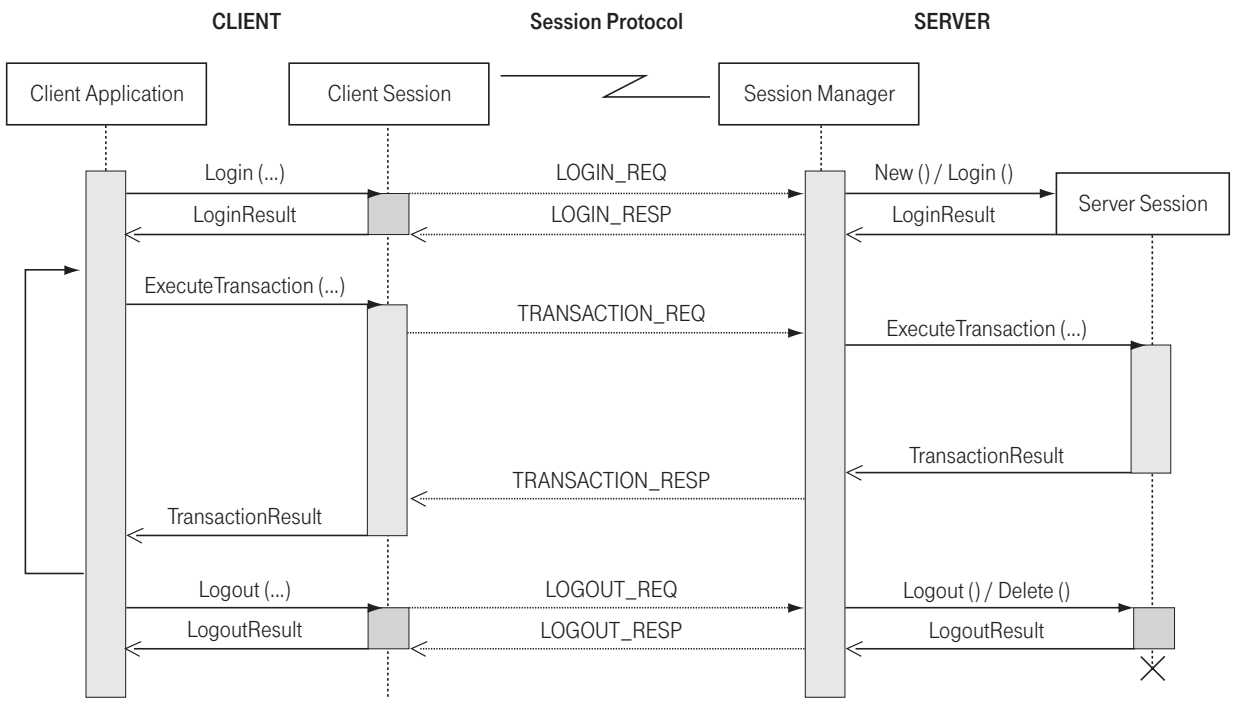
Abbrüche von Transaktionen

Abhängig von der Anwendung können bei der Unterbrechung einer Transaktion verschiedene Verhaltensweisen auftreten. Der ungünstigste Fall ist die „undefinierte Verhaltensweise“.

Undefinierte Verhaltensweise

Die Anwendung reagiert in diesem Fall mit einer Fehlermeldung und startet die Sitzung erneut. Dabei gehen Informationen über alle durchgeführten Transaktionen verloren. Es

Bild 16 **Ablauf einer einfachen Sitzung unter Berücksitigung der Kommunikationskomponente**



REQ Request RESP Response

ist unklar, ob die letzte Transaktion durchgeführt oder unterbrochen wurde. In dieser Situation ist ebenfalls unklar, ob alle Transaktionen verloren gegangen sind oder nur die letzte unterbrochen wurde. Besonders kritisch ist ein solches Verhalten beispielsweise bei Finanztransaktionen oder Bestellvorgängen. Das Auftreten undefinierter Verhaltensweisen ist ein Zeichen für unsauberes Design im Sitzungsmanagement. Die Funktion ist zwar im Normalbetrieb sichergestellt, die Fehlerbehandlung ist allerdings unzureichend entwickelt.

Rollback aller Transaktionen

Ein Rollback macht durchgeführte Transaktionen wieder rückgängig. Wurde zum Beispiel durch die Transaktionen einer Sitzung der Datenbestand einer Datenbank manipuliert, so führt ein Rollback aller Transaktionen dazu, dass der Datenbankbestand wiederhergestellt wird, wie zum Zeitpunkt vor der Sitzung. Für den Anwender ist der komplette Rollback oft aufwendig, weil er alle wichtigen Transaktionen erneut durchführen muss. Das Verfahren sorgt allerdings für einen stabilen Datenbestand.

Rollback der letzten Transaktion

In diesem Fall wird lediglich die letzte Transaktion rückgängig gemacht. Der undefinierte Zustand besteht nicht und der Anwender erkennt, dass er nur den letzten Schritt erneut durchführen muss. Dazu benötigt entweder er oder die Anwendung selbst eine Liste der durchgeführten Transaktionen, um zu kontrollieren, welche Transaktionen in der Sitzung erfolgreich abgeschlossen wurden und welche nicht.

Wiederherstellung verloren gegangener Transaktionen

Wird die Verbindung unterbrochen, so prüft die Anwendung bei einem erneuten Verbindungsaufbau den Zustand der Sitzung und wiederholt automatisch verloren gegangene Transaktionen. Der Anwender merkt in diesem Fall lediglich eine kurze Unterbrechung durch den Wiederaufbau der Verbindung und kann mit seiner Arbeit fortfahren.

Schlankes Transportprotokoll

Das Transportprotokoll ist Bestandteil der Kommunikationskomponente. Es hat die Aufgabe, Befehle und Informationen des Sitzungsmanagements für die Übertragung vorzubereiten und legt fest, wie die Kommunikation zwischen Client und Server auf der Datenübertragungsebene abläuft. Bild 16 zeigt den Ablauf einer einfachen Sitzung unter Berücksichtigung der Kommunikationskomponente. Während das Sitzungsprotokoll noch einen Bezug zur Anwendung besitzt, definiert das Transportprotokoll (Session Protocol) nur die Befehle für die Übermittlung der Informationen. Es ist vergleichbar mit einem Briefumschlag, in dem ein Brief (der Befehl des Sitzungsprotokolls) steckt, der von der Post (dem Übertragungsmedium) übermittelt wird. Auch beim Versenden eines Briefes müssen bestimmte Vorgaben (Protokoll) eingehalten werden, damit die Informationsübermittlung funktioniert: Es ist notwendig, dass auf dem Umschlag die Empfängeradresse richtig angegeben ist, und die Sendung muss an einer zulässigen Stelle an das Transportnetz übergeben werden, nämlich bei einem Postamt oder Briefkasten.

Das Transportprotokoll verwendet für die Übertragung wiederum ein standardisiertes

Basisprotokoll. Das Transport Control Protocol oder das User Datagram Protocol (UDP) können als Basisprotokolle zur Übermittlung des Transportprotokolls verwendet werden. Diese Basisprotokolle erlauben dem Entwickler, die vorhandenen Kommunikationsdienste des Betriebssystems zu verwenden. Theoretisch wäre es möglich, das Sitzungsprotokoll direkt mit dem Basisprotokoll zu übertragen und sich die Implementierung eines zusätzlichen Transportprotokolls zu sparen. Dies funktioniert noch bei einfachen Anwendungen. Bei der Entwicklung einer mobilen Anwendung allerdings bestehen folgende Gefahren:

- Zusätzliche Komplexität des Sitzungsprotokolls
  Das mobile Sitzungsprotokoll besitzt, wie im vorangegangenen Abschnitt dargestellt, bereits eine hohe Komplexität. Werden nun zusätzliche Anforderungen wie zum Beispiel Kompression der Daten, Verschlüsselung oder Übertragungskontrolle in dieses Protokoll integriert, lassen sich Programmfehler schwerer isolieren und damit auch schlechter beheben.
- Keine Trennung zwischen Anwendung und Kommunikation
  Die geforderte Trennung zwischen Anwendungs- und Kommunikationskomponente ist nicht mehr gegeben. Das bedeutet, dass die Anwendung Aufgaben wahrnimmt, die eigentlich nicht in ihren Zuständigkeitsbereich fallen. Die Portabilität[21] der Anwendung auf ein anderes Übertragungsprotokoll, die Wiederverwendung von Anwendungs- oder Kommunikationskomponenten und eine automatische Fehlerbehebung lassen sich damit nur mit hohem Aufwand umsetzen.
- Unzulänglichkeiten des Basisprotokolls
  Protokolle wie TCP oder UDP wurden nicht als mobile Übertragungsprotokolle entwickelt[22]. Sie besitzen einige Eigenschaften, die die Anwendung bei der kabellosen Übertragung negativ beeinflussen können. Verwendet das Sitzungsmanagement direkt das Basisprotokoll, so ist es diesen Unzulänglichkeiten in vollem Umfang „ausgesetzt". Zur Behebung dieser Probleme ist nicht nur die Erweiterung des Sitzungsprotokolls, sondern auch die des Sitzungsmanagements nötig.

Der Einsatz eines Transportprotokolls bietet dem Entwickler Freiräume. Er kann zunächst eine einfache Variante des Transportprotokolls implementieren, welche die Daten unverändert übermittelt. Damit wird das Sitzungsprotokoll direkt auf Korrektheit geprüft. Läuft die Anwendung stabil, so können im nächsten Schritt zusätzliche Funktionen und Optimierungen wie z. B. Datenkomprimierung im Transportprotokoll vorgenommen werden ohne dadurch die Stabilität des Sitzungsmanagements zu gefährden. Zeigt sich im Feldbetrieb, dass die Übertragung Schwächen aufweist oder gar ein anderes Basisprotokoll eingesetzt werden muss, so kann dies auch unabhängig von der Anwendung geschehen.

Anforderungen an ein schlankes Transportprotokoll

Die Auswirkungen der Funkübertragung wurden bereits in den Unterrichtsblättern Nr. 11/2001, S. 626 ff ausführlich beschrieben. Für die Datenübermittlung ergeben sich zusammengefasst folgende Rahmenbedingungen:

- hohe Paketlaufzeiten,
- geringe Datenrate
- Asymmetrie und
- unerwartete Verbindungsabbrüche.

Ein schlankes Transportprotokoll muss seine Aufgabe mit möglichst geringem Daten- und Paketvolumen umsetzen, um diese Rahmenbedingungen einhalten zu können. Im Einzelnen ergeben sich die folgenden Anforderungen.

Geringer Protokoll-Overhead

Ein Protokollblock setzt sich aus einem Kopf (Header) und einem Datenteil zusammen. Der Header ermöglicht die Interpretation des Protokollblocks. Im Header befinden sich Protokollbefehl, Länge des Datenteils, Adressinformationen, Prüfsumme und Sequenzzähler. Diese Informationen stellen den Protokoll-Overhead[23] dar und sind auf das Notwendige begrenzt, denn sie vergrößern das Datenvolumen bei der Übertragung.

Effizienter Protokollablauf

Ein Protokollablauf baut sich aus Anfragen und Antworten auf. Der Client stellt eine Anfrage an den Server, der Server bearbeitet die Anfrage und sendet eine Antwort. Jede Kombination aus Anfrage und Antwort bremst den Programmablauf auf Grund der hohen Paketlaufzeit im General Paket Radio Service zusätzlich. Deshalb ist es notwendig, die Protokollabläufe mit so wenig Anfrage- und Antwort-Paaren wie möglich umzusetzen:

- Anfragen und Antworten gebündelt übermitteln (Bild 17)
  Werden Informationen gesammelt, bevor sie übertragen werden, so spart dies deutlich Übertragungszeit. Ein effizientes Protokoll unterstützt deshalb das gebündelte Senden und Empfangen von Befehlen.
- Wenige Informationen senden, viele Informationen empfangen
  Die Asymmetrie von GPRS erfordert einen sparsamen Umgang mit dem Sendekanal. Anfragen sollten deshalb so klein wie möglich gehalten werden.

Push statt Polling

Wenn ein Client in zyklischen Abständen neue Informationen beim Server anfordert, so spricht man von „Polling". Polling hat zwei Nachteile: Zum einen erhält der Client nur dann die aktuellen Informationen (z. B. Mails, Aufträge, Alarme), wenn er sie abfragt, und zum anderen vergrößert es das Datenvolumen. Eine Alternative stellt die Integration eines so genannten „Push"-Mechanismus in das Transportprotokoll dar. Der Client abonniert dabei durch eine einzelne Anfrage einen Dienst beim Server. Der Server sendet darauf-

[21] **Portabilität:** Eigenschaft von Programmen ohne größere Änderung auf anderen Rechenanlagen ausgeführt werden zu können. Dies wird häufig durch standardisierte Programmiersprachen oder eine Zweiteilung von Programmen in einen maschinenunabhängigen und -abhängigen Teil erreicht. Der maschinenabhängige Teil wird bei der Übertragung auf ein anderes Rechensystem neu geschrieben.

[22] Siehe hierzu den Beitrag „Mit GPRS ins Intranet – Optimierungsmaßnahmen für den praktischen Einsatz in Unternehmensnetzen", Unterrichtsblätter Nr. 11/2001, S. 626 ff.

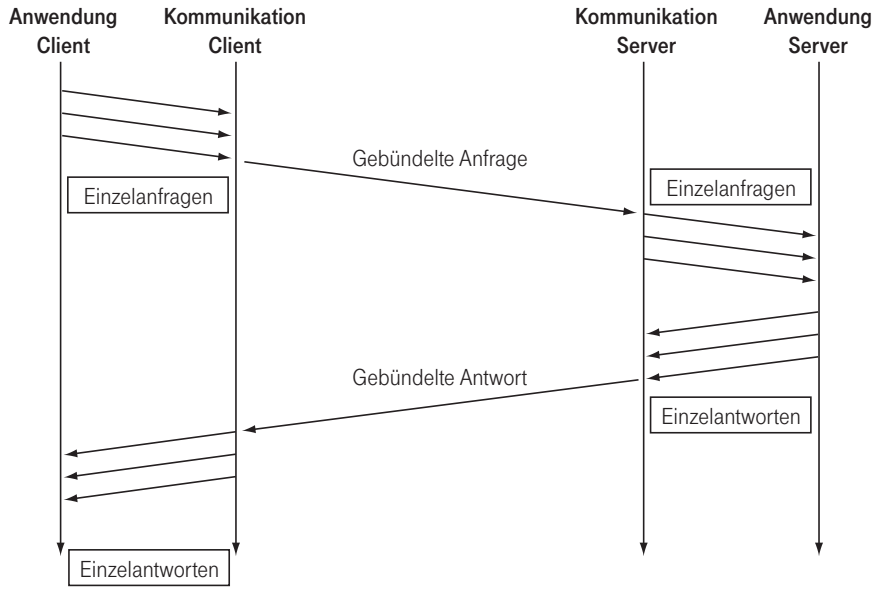
[23] Unter einem **Protokoll-Overhead** versteht man Steuerinformationen, die vom Protokoll zusätzlich zu den Rohdaten hinzugefügt werden, damit das Protokoll seine Aufgabe erfüllen kann.

hin automatisch neue verfügbare Informationen gebündelt als Antwortblock, ohne dass vorher eine erneute Anfrage notwendig ist.

Verbindungsüberwachung

Die Verbindungsqualität kann sich bei einer Funkübertragung verändern, gegebenenfalls bricht die Verbindung sogar vollständig ab. Um diese Situationen auch auf der Ebene der Datenübermittlung möglichst frühzeitig zu erkennen, benötigt das Protokoll eine eigene Verbindungsüberwachung. Die Verbindungsüberwachung kann zum Beispiel aus einer einfachen Anfrage- und Antwort-Kombination bestehen. Ein solcher Mechanismus wird häufig auch als „Keep-Alive" (lebend/offen halten) bezeichnet. Der Client sendet dazu eine Keep-Alive-Anfrage an den Server, um zu ermitteln, ob der Server noch erreichbar ist. Der Server sendet bei Erhalt sofort eine Keep-Alive-Antwort. Der Client erkennt anhand der Antwort, dass die Verbindung noch korrekt arbeitet. Im Gegensatz zu anderen Protokollbefehlen können bei Keep-Alives kurze Zeitintervalle verwendet werden, um auf diese Weise schnell eine Unterbrechung feststellen zu können. Keep-Alives vergrößern das übertragene Datenvolumen, so dass hier besonders auf die Minimierung des Protokoll-Overheads geachtet und ein möglichst guter Kompromiss zwischen Effizienz[24] und Kontrolle gefunden werden muss.

Bild 17 **Gebündelte Übermittlung von Anfragen durch ein optimiertes Transportprotokoll**



Integration von weiteren Optimierungen

Es ist ratsam, ein Protokoll schon zu Beginn so auszulegen, dass es Optimierungen des Datenstroms erlaubt. Dies wird dann nötig, wenn sich Anforderungen an die Kommunikationsabläufe ändern. Hier können beispielsweise Datenströme zusätzlich komprimiert oder verschlüsselt werden. Die Themen Kompression und Verschlüsselung werden im Application Configuration & Developer Guide der T-Mobile mit konkreten Code-Beispielen und Quellenangaben ausführlich behandelt. *(Br)*

Der Beitrag wird fortgesetzt.

[24] **Effizienz:** Wirksamkeit und Wirtschaftlichkeit.