James F. Kurose
Keith W. Ross

Computernetze

Ein Top-Down-Ansatz mit Schwerpunkt Internet

Pearson Studium
ein Imprint der Pearson Education Deutschland GmbH

KAPITEL 3

Transportschicht

3.1 Dienste und Prinzipien auf der Transportschicht

Zwischen der Anwendungs- und der Vermittlungsschicht angesiedelt, bildet die Transportschicht das Kernstück der geschichteten Netzwerkarchitektur. Sie erfüllt die wichtige Rolle der direkten Bereitstellung von Kommunikationsdiensten für die Anwendungsprozesse, die auf verschiedenen Hosts laufen. In diesem Kapitel beschreiben wir die Dienste, die von einem Protokoll der Transportschicht bereitgestellt werden können, und die verschiedenen, der Bereitstellung dieser Dienste zugrunde liegenden Prinzipien. Ferner wird beschrieben, wie diese Dienste in bestehenden Protokollen implementiert sind. Wie bisher, liegt die besondere Betonung auf den Internet-Protokollen, d. h. den TCP- und UDP-Protokollen auf der Transportschicht.

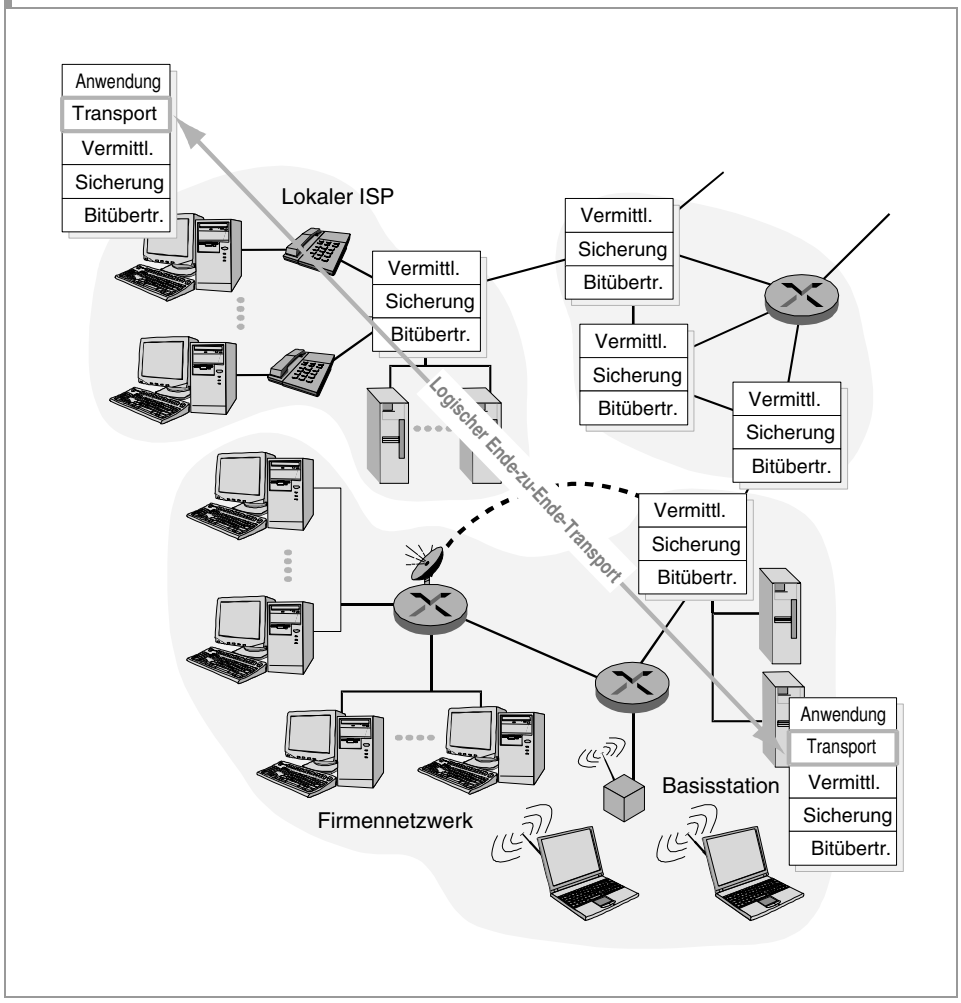
In den ersten beiden Kapiteln haben wir die Rolle der Transportschicht und die von ihr bereitgestellten Dienste kurz angesprochen. Hier folgt nun eine kurze Übersicht dessen, was Sie über die Transportschicht bisher erfahren haben.

Ein Protokoll der Transportschicht bietet eine **logische Kommunikation** zwischen Anwendungsprozessen, die auf unterschiedlichen Hosts laufen. Unter logischer Kommunikation verstehen wir, dass die kommunizierenden Anwendungsprozesse zwar *nicht physisch* miteinander verbunden sind (tatsächlich können sie sich an entgegengesetzten Stellen der Erde befinden und über zahlreiche Router und viele verschiedene Verbindungsleitungen verbunden sein), aus Sicht der Anwendung aber physikalisch verbunden erscheinen. Anwendungsprozesse verwenden die von der Transportschicht bereitgestellte logische Kommunikation, um Nachrichten miteinander auszutauschen, ohne sich um die Details der physikalischen Infrastruktur, über die diese Nachrichten fließen, kümmern zu müssen. Abbildung 3.1 stellt das Konzept der logischen Kommunikation dar.

Wie aus Abbildung 3.1 deutlich wird, werden Protokolle auf der Transportschicht in den Endsystemen und nicht in Netzwerk-Routern implementiert. Netzwerk-Router agieren lediglich in den Feldern der 3-PDUs der Vermittlungsschicht; sie agieren nicht als Felder der Transportschicht.

Auf der sendenden Seite konvertiert die Transportschicht die Nachrichten, die von einem sendenden Anwendungsprozess ankommen, in 4-PDUs (d. h. in Protokolldateneinheiten der Transportschicht). Dies wird (möglicherweise) dadurch bewerkstelligt, dass die Nachrichten einer Anwendung in kleinere Stücke aufgeteilt und jedem Stück ein Header der Transportschicht angehängt wird, um 4-PDUs zu erzeugen. Die Transportschicht gibt die 4-PDUs dann an die Vermittlungsschicht weiter, wo jede 4-PDU in einer 3-PDU verkapselt wird. Auf der empfangenden Seite

Abbildung 3.1 Die Transportschicht bietet eine logische und keine physikalische Kommunikation zwischen Anwendungen.



erhält die Transportschicht die 4-PDUs von der Vermittlungsschicht, entfernt den Transport-Header von den 4-PDUs, setzt die Nachrichten wieder zusammen und gibt sie an einen empfangenden Anwendungsprozess weiter.

Ein Computernetzwerk kann Netzwerkanwendungen mehr als ein Protokoll auf der Transportschicht zur Verfügung stellen. Das Internet hat beispielsweise zwei Protokolle: TCP und UDP. Jedes dieser Protokolle bietet andere Transportschichtdienste für die nutzende Anwendung.

Alle Protokolle der Transportschicht bieten einer Anwendung einen Multiplex/Demultiplex-Dienst. Dieser Dienst wird ausführlich im nächsten Abschnitt beschrieben. Wie in Abschnitt 2.1 erwähnt, kann ein Transportprotokoll abgesehen vom Multiplex/Demultiplex-Dienst auch weitere Dienste, z. B. zuverlässigen Datentransfer und Zusicherungen über Bandbreiten und Verzögerungen, bereitstellen.

3.1.1 Beziehung zwischen der Transport- und der Vermittlungsschicht

Die Transportschicht liegt unmittelbar oberhalb der Vermittlungsschicht im Protokollstack. Während ein Protokoll der Transportschicht eine *logische Kommunikation zwischen Prozessen* bietet, die auf unterschiedlichen Hosts laufen, stellt ein Protokoll der Vermittlungsschicht eine *logische Kommunikation zwischen Hosts* bereit. Dieser Unterschied ist subtil, aber sehr wichtig. Wir wollen ihn im Folgenden anhand einer Haushaltsanalogie erklären.

Man stelle sich zwei Häuser vor, von denen eines an der Ostküste und das andere an der Westküste (der USA) steht; in jedem der beiden Häuser wohnen ein Dutzend Kinder. Die Kids in dem Haushalt an der Ostküste sind Cousins der Kids im Haushalt an der Westküste. Die Kids der beiden Haushalte haben Spaß, sich gegenseitig zu schreiben; jedes Kind schreibt jedem Cousin jede Woche, wobei jeder Brief über die konventionelle gelbe Post in einem getrennten Umschlag befördert wird. Folglich verschickt jeder Haushalt pro Woche 144 Briefe an den anderen Haushalt. (Diese Kids würden viel Geld sparen, wenn sie E-Mail hätten!) In jedem Haushalt ist eines der Kinder – Ann im Haus an der Westküste und Bill in dem Haus an der Ostküste – für die Sammlung und die Verteilung der Post zuständig. Ann besucht jede Woche ihre Brüder und Schwestern, sammelt die Post ein und gibt sie dem Briefträger bei seiner täglichen Runde. Wenn Briefe im Haus an der Westküste ankommen, hat Ann auch die Aufgabe, die Post an ihre Brüder und Schwestern zu verteilen. An der Ostküste führt Bill diese Aufgabe aus.

In diesem Beispiel bietet der Postdienst eine logische Kommunikation zwischen den beiden Häusern: Er befördert Post von Haus zu Haus und nicht von Person zu Person. Andererseits bieten Ann und Bill eine logische Kommunikation unter den Cousins: Sie sammeln die Post von ihren Geschwistern ein und verteilen ankommende Post an sie. Aus Sicht der Cousins *sind* Ann und Bill der Postdienst, obwohl die beiden nur ein Teil (der Endsystemteil) des Ende-zu-Ende-Transportprozesses sind. Dieses Haushaltsbeispiel dient als Analogie, um den Zusammenhang zwischen der Transport- und der Vermittlungsschicht zu erklären:

Hosts (auch Endsysteme genannt) = Häuser

Prozesse = Cousins

Anwendungsnachrichten = Briefe in Umschlägen

Protokoll der Netzwerkschicht = Postdienst (mit Briefträgern)

Protokoll der Transportschicht = Ann und Bill

Wir fahren mit dieser Analogie fort und stellen fest, dass Ann und Bill ihre gesamte Arbeit innerhalb ihres jeweiligen Heims verrichten; sie haben z. B. nichts mit dem Sortieren von Post in einem Postamt oder der Beförderung von Post von einer Postverteilerstelle zu einer anderen zu tun. Ähnlich leben die Protokolle der Transportschicht in den Endsystemen. Innerhalb eines Endsystems verschiebt ein Transportprotokoll Nachrichten von Anwendungsprozessen zur Netzwerkperipherie (d. h. zur Vermittlungsschicht) und umgekehrt; es hat aber keinen Einfluss darauf, wie die Nachrichten innerhalb des Netzwerkkerns verschoben werden. Wie aus Abbildung 3.1 deutlich wird, wirken die dazwischen liegenden Router weder auf Informationen ein, die auf der Transportschicht möglicherweise an die Anwendungsnachrichten angehängt werden, noch erkennen sie solche.

Wir greifen wieder unsere Familiensaga auf und nehmen an, dass zwei andere Cousins, sagen wir Susan und Harvey, Ann und Bill ablösen, wenn diese in den Ferien sind. Zum Leidwesen der beiden Familien sammeln und verteilen Susan und Harvey die Post nicht genau auf die gleiche Weise wie Ann und Bill. Susan und Harvey sind kleinere Kinder und so passiert es schon mal, dass sie Post weniger regelmäßig abholen und verteilen und auch mal einige Briefe verlieren (die manchmal vom Haushund zerfetzt werden). Susan und Harvey bieten also nicht die gleichen Dienste (d. h. das gleiche Dienstmodell) wie Ann und Bill. Vergleichbar damit kann ein Computernetzwerk mehrere Transportprotokolle zur Verfügung stellen, die sich hinsichtlich ihres Dienstmodells für Anwendungen unterscheiden.

Die möglichen Dienste, die Ann und Bill bereitstellen können, sind deutlich auf die möglichen Dienste eingeschränkt, die der Postdienst bietet. Wenn der Postdienst beispielsweise keine Höchstgrenze festlegt, wie lange die Zustellung eines Briefs zwischen den beiden Häuser dauern darf (z. B. drei Tage), besteht für Ann und Bill keine Möglichkeit, eine maximale Verzögerung für die Postzustellung zwischen zwei Cousins zuzusichern. Ähnlich werden die Dienste, die ein Transportprotokoll bereitstellen kann, oft durch das Dienstmodell des zugrunde liegenden Protokolls auf der Vermittlungsschicht beschränkt. Wenn das Protokoll auf der Vermittlungsschicht keine Zusicherungen über Verzögerung oder Bandbreite für 4-PDUs, die zwischen zwei Hosts versendet werden, machen kann, dann kann das Protokoll auf der Transportschicht keine Zusicherungen über Verzögerung oder Bandbreite für die zwischen Prozessen ausgetauschten Nachrichten geben.

Dennoch *können* bestimmte Dienste von einem Transportprotokoll geboten werden, auch wenn das zugrunde liegende Netzwerkprotokoll auf der Vermittlungsschicht keinen entsprechenden Dienst bietet. Wie wir in diesem Kapitel noch sehen werden, kann ein Transportprotokoll einer Anwendung z. B. zuverlässigen Datentransferdienst bieten, auch wenn das zugrunde liegende Netzwerkprotokoll unzuverlässig ist, d. h. auch wenn das Netzwerkprotokoll Pakete verliert, verstümmelt und dupliziert. Als weiteres Beispiel (das wir umfassender in Kapitel 7 in Zusammenhang mit Netzwerksicherheit behandeln) kann ein Transportprotokoll Verschlüsselung anwenden, um zuzusichern, dass Nachrichten nicht von Eindringlingen gelesen werden, auch wenn die Vermittlungsschicht keine Vertraulichkeit von 4-PDUs zusichern kann.

3.1.2 Übersicht über die Transportschicht im Internet

Wir erinnern uns, dass das Internet, und allgemeiner ein TCP/IP-Netzwerk, der Anwendungsschicht zwei unterschiedliche Protokolle auf der Transportschicht zur Verfügung stellt. Eines dieser Protokolle ist **UDP** (User Datagram Protocol), das Anwendungen einen unzuverlässigen, verbindungslosen Dienst bereitstellt. Das zweite ist **TCP** (Transmission Control Protocol), das Anwendungen einen zuverlässigen, verbindungsorientierten Dienst bietet. Im Design einer Netzwerkanwendung muss der Anwendungsentwickler eines dieser beiden Transportprotokolle spezifizieren. Wir haben in den Abschnitten 2.6 und 2.7 gesehen, dass der Anwendungsentwickler entweder UDP oder TCP wählt, wenn er Sockets erstellt.

Um die Terminologie zu vereinfachen, sprechen wir in Zusammenhang mit dem Internet von einem **Segment**, um eine 4-PDU zu bezeichnen. In der Internet-Literatur (z. B. in RFCs) wird die PDU in Zusammenhang mit TCP oft als »Segment« und in Zusammenhang mit UDP als **Datagramm** bezeichnet. In der gleichen Literatur findet

man aber auch den Begriff »Datagramm« für PDUs der Vermittlungsschicht! Wir sind der Ansicht, dass in einem Fachbuch über die Grundlagen im Vernetzungsbereich wie diesem keine Verwirrung mit Begriffen gestiftet werden soll, und verwenden daher den Begriff »Segment« für TCP- *und* UDP-PDUs, während wir den Begriff »Datagramm« nur für PDUs der Vermittlungsschicht verwenden.

Bevor wir mit unserer kurzen Einführung von UDP und TCP fortfahren, sind an dieser Stelle ein paar Worte über die Vermittlungsschicht des Internets nützlich. (Die Vermittlungsschicht ist Thema von Kapitel 4.) Das Protokoll der Internet-Vermittlungsschicht ist IP (Internet Protocol). IP bietet eine logische Kommunikation zwischen Hosts. Das IP-Dienstmodell ist der **Best-Effort-Dienst**. Das bedeutet, dass IP Segmente zwischen kommunizierenden Hosts nach »bestem Bemühen« überträgt, *allerdings keine Zusicherungen macht*. Insbesondere gibt es keine Zusicherung über die Ankunft von Segmenten, über deren Ankunft in der richtigen Reihenfolge und über die Integrität der Daten in den Segmenten. Folglich bietet IP einen **unzuverlässigen Dienst**. Jeder Host hat eine IP-Adresse. IP-Adressierung wird ausführlich in Kapitel 4 behandelt; vorläufig genügt es zu wissen, dass jeder Host eine *eindeutige* IP-Adresse hat.

Nachdem wir einen Blick auf das IP-Dienstmodell geworfen haben, fassen wir das Dienstmodell von UDP und TCP zusammen. Die grundlegende Verantwortung von UDP und TCP ist im Wesentlichen die Erweiterung des IP-Übertragungsdienstes zwischen zwei Endsystemen auf einen Übertragungsdienst zwischen zwei Prozessen, die auf zwei Endsystemen laufen. Diese Erweiterung der Host-zu-Host- auf die Prozess-zu-Prozess-Übertragung wird als **Anwendungsmultiplexen** und -**demultiplexen** bezeichnet (wird im nächsten Abschnitt beschrieben). UDP und TCP bieten auch Integritätsprüfung dadurch, dass Fehlererkennungsfelder in die Header einbezogen werden. Diese beiden minimalen Transportschichtdienste – Prozess-zu-Prozess-Datenübertragung und Fehlerprüfung – sind die einzigen Dienste, die UDP bereitstellt! Wie IP ist UDP ein unzuverlässiger Dienst; es gibt keine Zusicherung, dass die von einem Prozess gesendeten Daten beim Zielprozess intakt ankommen. UDP wird ausführlich in Abschnitt 3.3 behandelt.

TCP dagegen bietet Anwendungen mehrere zusätzliche Dienste. Vor allem bietet es **zuverlässigen Datentransfer**. Mit Hilfe von Flusskontrolle, Sequenznummern, Bestätigungen (ACKs) und Timern (Techniken, die in diesem Kapitel ausführlich behandelt werden) gewährleistet TCP, dass die Daten vom sendenden zum empfangenden Prozess korrekt und in der richtigen Reihenfolge übertragen werden. TCP konvertiert folglich den unzuverlässigen Dienst von IP zwischen Endsystemen in einen zuverlässigen Datentransportdienst zwischen Prozessen. TCP nutzt auch **Überlastkontrolle**. Dabei handelt es sich nicht so sehr um einen Dienst für die aufrufende Anwendung, als vielmehr um einen für das Internet insgesamt, also einen Dienst zum allgemeinen Nutzen. Grob gesagt, hindert die TCP-Überlastkontrolle eine TCP-Verbindung daran, die Verbindungsleitungen und Switches zwischen kommunizierenden Hosts mit übermäßigem Verkehrsvolumen zu überschwemmen. Im Prinzip können TCP-Verbindungen, die ein überlastetes Netzwerk überqueren, die Bandbreite der betreffenden Verbindungsleitung gemeinsam nutzen. Dies wird dadurch bewerkstelligt, dass die Rate, in der die sendende Seite Verkehr in das Netzwerk einspeisen kann, reguliert wird. Im Gegensatz dazu wird UDP-Verkehr nicht reguliert. Eine Anwendung, die UDP-Transport nutzt, kann in jeder beliebigen Rate senden, solange sie will.

Ein Protokoll, das zuverlässigen Datentransfer und Überlastkontrolle bietet, ist natürlich komplex. Wir müssen die Prinzipien von zuverlässigem Datentransfer und Überlastkontrolle in mehreren Abschnitten beschreiben und zusätzliche Abschnitte sind notwendig, um das TCP-Protokoll selbst abzudecken. Diese Themen werden in den Abschnitten 3.4 bis 3.8 behandelt. In diesem Kapitel wechseln wir zwischen einem bestimmten Basisprinzip und dem TCP-Protokoll. Beispielsweise behandeln wir zuerst den zuverlässigen Datentransfer im Allgemeinen und anschließend die Art, wie TCP im Besonderen zuverlässigen Datentransfer bereitstellt. Ebenso behandeln wir Überlastkontrolle zuerst im allgemeinen Umfeld und anschließend spezifisch in TCP. Bevor wir uns diesen Themen zuwenden, betrachten wir im nächsten Abschnitt zunächst das Multiplexen und Demultiplexen von Anwendungen.

3.2 Multiplexen und Demultiplexen von Anwendungen

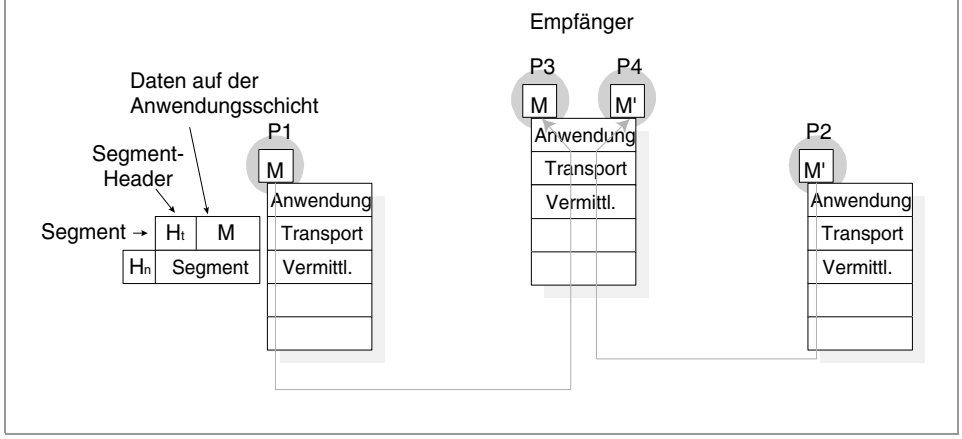
Dieser Abschnitt befasst sich mit dem Multiplexen und Demultiplexen von Netzwerkanwendungen. Um eine konkrete Beschreibung sicherzustellen, befassen wir uns mit diesem grundlegenden Transportschichtdienst in Zusammenhang mit dem Internet. Wir betonen allerdings, dass für alle Computernetzwerke ein Multiplex/Demultiplex-Dienst erforderlich ist.

Der Multiplex/Demultiplex-Dienst zählt zwar nicht zu den interessantesten Diensten, die ein Protokoll der Transportschicht bereitstellen kann, er ist jedoch absolut wichtig. Um dies zu verstehen, bedenke man die Tatsache, dass IP Daten zwischen zwei Endsystemen überträgt, wobei jedes Endsystem mit einer eindeutigen IP-Adresse identifiziert wird. IP überträgt Daten *nicht* zwischen den Anwendungsprozessen, die auf diesen Endsystemen laufen. Die Erweiterung der Host-zu-Host- auf die Prozess-zu-Prozess-Übertragung wird durch Anwendungsmultiplexen und -demultiplexen erreicht.

Auf dem Zielhost empfängt die Transportschicht Segmente (d. h. PDUs der Transportschicht) von der unmittelbar darunter liegenden Vermittlungsschicht. Die Transportschicht ist für die Übertragung der Daten in diesen Segmenten an den entsprechenden Anwendungsprozess, der auf dem Host läuft, zuständig. Wir betrachten ein Beispiel. Angenommen, Sie sitzen vor Ihrem Computer und laden Web-Seiten herunter, während Sie eine FTP-Sitzung und zwei Telnet-Sitzungen ausführen. Das heißt, es laufen momentan vier Netzwerkanwendungsprozesse (zwei Telnet-Prozesse, ein FTP-Prozess und ein HTTP-Prozess). Wenn die Transportschicht in Ihrem Computer Daten von der darunter liegenden Vermittlungsschicht empfängt, muss sie die empfangenen Daten an einen dieser vier Prozesse weiterleiten. Wie geht sie dabei vor?

Jedes Transportschichtsegment umfasst eine Reihe von Feldern, die den Prozess bestimmen, an den die Daten des Segments zu übertragen sind. Am empfangenden Ende kann die Transportschicht dann diese Felder prüfen, um den empfangenden Prozess zu ermitteln und das Segment an diesen Prozess weiterzuleiten. Die Aufgabe der Übertragung der in einem Transportschichtsegment enthaltenen Daten an den richtigen Anwendungsprozess nennt man **Demultiplexen**. Die Aufgabe des Einsammelns von Daten im Quellhost aus verschiedenen Anwendungsprozessen, die Vervollständigung der Daten mit Header-Informationen (die später beim Demultiplexen benutzt werden), um Segmente zu bilden, und die Weiterleitung der Segmente an die Vermittlungsschicht wird als **Multiplexen** bezeichnet. Multiplexen und Demultiplexen sind in Abbildung 3.2 dargestellt.

Abbildung 3.2 Multiplexen und Demultiplexen



Um Demultiplexen besser zu verstehen, betrachten wir wieder unsere Haushaltssaga aus dem vorherigen Abschnitt. Jedes der Kids wird anhand seines Namens unterschieden. Wenn Bill einen Stapel Post vom Briefträger erhält, führt er eine Demultiplexoperation durch, indem er feststellt, an wen die Briefe adressiert sind, und die Post dann an seine Brüder und Schwestern verteilt. Ann führt eine Multiplexoperation durch, wenn sie Briefe von ihren Geschwistern einsammelt und die angesammelte Post dem Briefträger übergibt.

UDP und TCP führen die Demultiplex- und Multiplexaufgaben dadurch aus, dass sie zwei spezielle Felder in die Segment-Header einbeziehen: das Feld **Portnummer Quelle** und das Feld **Portnummer Ziel**. Diese beiden Felder sind in Abbildung 3.3 dargestellt. Zusammen identifizieren die beiden Felder eindeutig einen Anwendungsprozess, der auf dem Zielhost läuft. (UDP- und TCP-Segmente haben noch weitere Felder, die in den nächsten Abschnitten dieses Kapitels beschrieben werden.)

Abbildung 3.3 Felder für die Portnummer von Quelle und Ziel in einem Segment der Transportschicht

Das Konzept von Portnummern wurde in den Abschnitten 2.6 und 2.7 in Zusammenhang mit der Anwendungsentwicklung und der Socket-Programmierung kurz vorgestellt. Die Portnummer ist eine 16-Bit-Nummer von 0 bis 65535. Die Portnummern im Bereich von 0 bis 1023 werden als **wohl bekannte (well-known) Portnummern** bezeichnet und sind eingeschränkt, was bedeutet, dass sie für wohl bekannte Anwendungsprotokolle wie HTTP und FTP reserviert sind. HTTP benutzt Portnummer 80 und FTP Portnummer 21. RFC 1700 enthält eine Liste aller wohl bekannten Portnummern. Wenn wir eine neue Anwendung (z. B. eine der Anwendungen in den Abschnitten 2.6 bis 2.8) entwickeln, müssen wir der Anwendung eine Portnummer zuweisen.

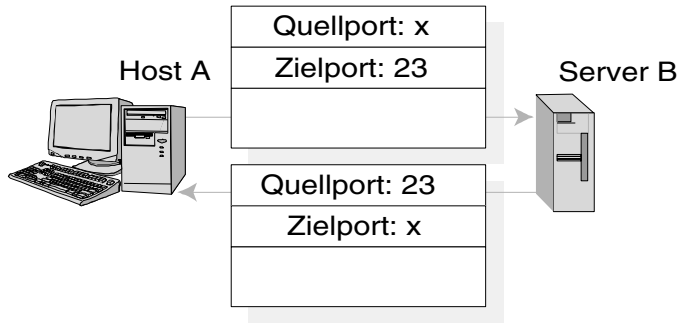
Angesichts der Tatsache, dass jeder Anwendungs*typ*, der auf einem Endsystem läuft, eine eindeutige Portnummer hat, stellt sich die Frage, warum das Transportschichtsegment Felder für zwei Portnummern – eines für die Portnummer der Quelle und eines für die des Ziels – beinhaltet? Die Antwort ist einfach: Ein Endsystem kann zwei Prozesse des gleichen Typs gleichzeitig ausführen; deswegen genügt die Zielportnummer einer Anwendung nicht immer, um einen bestimmten Prozess zu identifizieren. Dies ist z. B. der Fall, wenn ein Web-Server für jede verarbeitete Anfrage einen neuen HTTP-Prozess startet. Jedes Mal, wenn dieser Web-Server mehr als eine Anfrage bedient (was keinesfalls ungewöhnlich ist), führt der Server mehr als einen Prozess mit Portnummer 80 aus. Um den Prozess, an den Daten gerichtet sind, eindeutig zu identifizieren, ist deshalb eine zweite Portnummer erforderlich.

Wie wird diese zweite Portnummer erzeugt? Welche Portnummer wird im Feld »Portnummer Quelle« eines Segments angegeben? Welche wird im Feld »Portnummer Ziel« eines Segments angegeben? Um diese Fragen zu beantworten, rufen wir uns aus Abschnitt 2.1 wieder ins Gedächtnis, dass Netzwerkanwendungen rund um das Client/Server-Modell organisiert sind. Normalerweise ist der Host, der die Anwendung einleitet, der Client, und der andere Host ist der Server. Wir betrachten ein spezifisches Beispiel. Angenommen, die Anwendung hat Portnummer 23 (diejenige für Telnet). Ein Transportschichtsegment verlässt den Client (d. h. den Host, der die Telnet-Sitzung gestartet hat) in Richtung Server. Wie lautet bei diesem Segment die Portnummer für die Quelle und das Ziel? Die Zielportnummer dieses Segments ist die der Anwendung, nämlich 23. Für die Quellportnummer benutzt der Client eine Nummer, die noch keinem anderen Hostprozess zugewiesen wurde. (Dies erfolgt automatisch durch die Transportschichtsoftware, die auf dem Client läuft, und ist für den Anwendungsentwickler transparent.) Es sei gegeben, dass der Client Portnummer x wählt. Jedes Segment, das dieser Prozess an den Telnet-Server sendet, enthält als Quellportnummer x und als Zielportnummer 23. Wenn das Segment beim Server ankommt, ermöglichen es die beiden Portnummern im Segment dem Serverhost, die Daten des Segments an den richtigen Anwendungsprozess weiterzuleiten. Die Zielportnummer 23 identifiziert einen Telnet-Prozess und die Quellportnummer x den spezifischen Telnet-Prozess.

Die Situation ist umgekehrt bei den Segmenten, die vom Server zum Client fließen. Die Quellportnummer ist jetzt die Anwendungsportnummer, also 23, und die Zielportnummer ist jetzt x (die *gleiche* x, die als Quellportnummer für die Segmente, die vom Client zum Server gesendet wurden, benutzt wird). Wenn ein Segment beim Client ankommt, ermöglichen es die Quell- und Zielportnummern im Segment dem Clienthost, die Daten des Segments an den richtigen Anwendungsprozess weiterzugeben, der durch das Portnummernpaar identifiziert wird (siehe Abbildung 3.4).

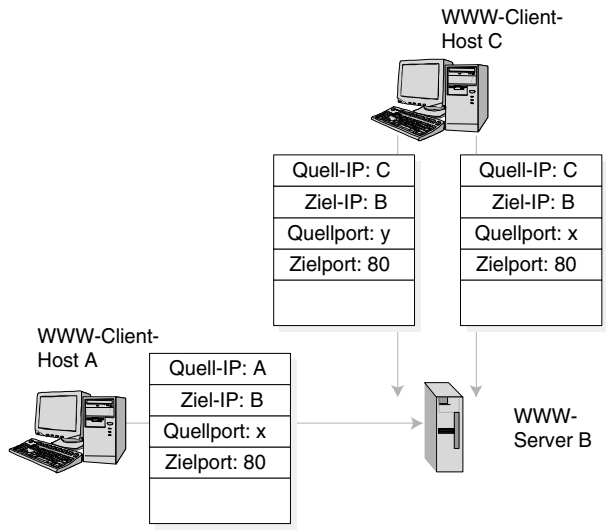
Sie fragen sich jetzt vielleicht, was passiert, wenn zwei verschiedene Clients eine Sitzung zu einem Server aufbauen und jeder der beiden Clients die gleiche Quellport-

Abbildung 3.4 Verwendung einer Quell- und Zielportnummer in einer Client/Server-Anwendung



nummer *x* wählt? Dies kann auf einem stark frequentierten WWW-Server, der viele Web-Clients gleichzeitig bedient, tatsächlich passieren. Wie kann der Server die Segmente demultiplexen, wenn die beiden Sitzungen genau das gleiche Portnummernpaar haben? Die Antwort auf diese Frage lautet, dass der Server auch die IP-Adressen in den IP-Datagrammen, in denen diese Segmente befördert werden, heranzieht. (IP-Datagramme und Adressierung werden ausführlich in Kapitel 4 behandelt.) Diese Situation ist in Abbildung 3.5 dargestellt, bei der Host C zwei HTTP-Sitzungen zu Server B und Host A eine HTTP-Sitzung zu B einleitet.

Abbildung 3.5 Zwei Clients verwenden die gleichen Portnummern, um mit der gleichen Server-Anwendung zu kommunizieren.



Die Hosts A und C und Server B haben jeweils eine eindeutige IP-Adresse – A, C bzw. B. Host C weist den beiden HTTP-Verbindungen, die von Host A ausgehen, jeweils eine unterschiedliche Quellportnummer (die SP-Nummer *x* bzw. *y*) zu. Da Host A die Quellportnummern aber unabhängig von C wählt, kann es passieren, dass er seiner HTTP-Verbindung ebenfalls SP = *x* zuweist. Server B wäre dennoch in der Lage, die beiden Verbindungen korrekt zu demultiplexen, weil die beiden Verbindungen je eine andere IP-Quelladresse haben. Zusammenfassend kann man sagen: Wenn ein Zielhost Daten von der Vermittlungsschicht empfängt, wird das Trio (IP-Quelladresse, Quellportnummer, Zielportnummer) verwendet, um die Daten an den entsprechenden Prozess weiterzuleiten.

Nachdem Sie jetzt wissen, wie die Transportschicht Netzwerkanwendungen multiplexen und demultiplexen kann, fahren wir mit der Beschreibung des Internet-Transportprotokolls UDP fort. Der nächste Abschnitt wird zeigen, dass UDP das Protokoll der Vermittlungsschicht um kaum mehr als einen Multiplex/Demultiplex-Dienst erweitert.

3.3 Verbindungsloser Transport: UDP

In diesem Abschnitt befassen wir uns mit den Merkmalen und der Funktionsweise von UDP. Wir empfehlen dem Leser, sich noch einmal Abschnitt 2.1 anzusehen, der eine Übersicht über das UDP-Dienstmodell beinhaltet, und Abschnitt 2.7, in dem Socket-Programmierung über UDP beschrieben wird.

Als Motivation für diese Beschreibung von UDP nehmen wir an, Sie sind daran interessiert, ein einfaches Transportprotokoll zu entwickeln. Wie können Sie vorgehen? Sie können sich zuerst die Verwendung eines hirnlosen Transportprotokolls überlegen. Insbesondere könnten Sie für die Sendeseite in Betracht ziehen, die Nachrichten von dem Anwendungsprozess entgegenzunehmen und sie direkt an die Vermittlungsschicht weiterzugeben. Auf der Empfangsseite können die von der Vermittlungsschicht ankommenden Nachrichten direkt an den Anwendungsprozess weitergegeben werden. Wie wir im vorherigen Abschnitt aber gelernt haben, müssen wir ein bisschen mehr als nichts tun. Zumindest muss die Transportschicht einen Multiplex/Demultiplex-Dienst bereitstellen, damit Daten zwischen der Vermittlungsschicht und dem richtigen Prozess weitergeleitet werden können.

Das in RFC 768 definierte UDP tut so wenig, wie man sich für ein Transportprotokoll nur vorstellen kann. Abgesehen von der Multiplex/Demultiplex-Funktion und einer geringen Fehlerprüfung fügt es nichts zu IP hinzu. Wenn sich der Anwendungsentwickler für UDP statt TCP entscheidet, spricht die Anwendung tatsächlich fast direkt mit IP. UDP nimmt Nachrichten vom Anwendungsprozess entgegen, hängt die Felder der Quell- und Zielportnummern für den Multiplex/Demultiplex-Dienst und einige kleinere Felder an und leitet das daraus resultierende Segment an die Vermittlungsschicht weiter. Die Vermittlungsschicht verkapselt das Segment in einem IP-Datagramm und macht dann einen Best-Effort-Versuch, um das Segment an den empfangenden Host zu übertragen. Wenn das Segment beim empfangenden Host ankommt, benutzt UDP die Zielportnummer, um die Daten des Segments an den richtigen Anwendungsprozess zu übertragen. Man beachte, dass es bei UDP kein Handshake zwischen den Einheiten der sendenden und empfangenden Transportschicht gibt, bevor ein Segment gesendet wird. Aus diesem Grund gilt UDP als *verbindungslos*.