FOPT 5: Eigenständige Client-Server-Anwendungen (Programmierung verteilter Anwendungen in Java 1)

In dieser Kurseinheit geht es um verteilte Anwendungen, bei denen wir sowohl ein Client- als auch ein Server-Programm selbst entwickeln. Diese Art von Anwendungen bezeichnen wir als eigenständige Client-Server-Anwendungen. Zur Kommunikation werden wir dabei Sockets oder RMI verwenden.

1 Einleitung und Grundlagen von Rechnernetzen

Sollten Sie keine Kenntnisse über die Grundlagen von Rechnernetzen haben, wie sie beispielsweise im Fernstudienmodul „Rechnernetze“ vermittelt werden, können Sie sich im Folgenden das grundlegende Wissen über das in Rechnernetzen wichtige Schichtenmodell, IP-Adressen und DNS-Namen sowie die Transportprotokolle UDP und TCP erarbeiten. Das Durcharbeiten der folgenden Seiten wird auch dann empfohlen, wenn Ihr erworbenes Wissen über diese Themenbereiche bereits schon etwas verblasst ist.

Lesen Sie den Beginn des Kapitels 5 des Lehrbuchs zur Einstimmung auf die folgenden Themengebiete durch. Wenn Sie keine oder mangelnde Kenntnisse über Rechnernetze oder das darüber Gelernte weitgehend wieder vergessen haben, dann lesen Sie Abschnitt 5.1 durch.

Zusammenfassung

Die Funktionen für die Kommunikation in Rechnernetzen werden in unterschiedliche Schichten aufgeteilt, wobei die eine Schicht ihre Funktionen der darüber liegenden Schicht über eine Schnittstelle zur Verfügung stellt. Es handelt sich dabei um folgende Schichten:

1. Bitübertragungsschicht (Physical Layer)
2. Leitungsschicht (Data Link Layer)
3. Vermittlungsschicht (Network Layer)
4. Transportschicht (Transport Layer)
5. Anwendungsschicht (Application Layer)

Die ersten beiden Schichten sind abhängig von der verwendeten Netztechnologie (z.B. Ethernet oder WLAN), während die Schichten 3 bis 5 davon unabhängig sind.

Die Kommunikation zwischen zwei in derselben Schicht benachbarten Rechnern wird über Protokolle geregelt. Ein Protokoll legt fest, wie die ausgetauschten Daten formatiert sind und welche Protokollnachricht welcher anderen zu folgen hat.

Im Internet werden Rechner durch IP-Adressen auf der 3. Schicht adressiert. Das Protokoll dieser Schicht heißt IP (Internet Protocol). Da die IP-Adressen für Menschen nicht leicht zu merken sind, werden alternativ hierarchische Namen verwendet, wobei die höchste Hierarchiestufe wie z.B. de oder com am Ende des Namens steht. Durch DNS (Domain Name System) werden Namen in Adressen übersetzt.

Mit Hilfe von Transportprotokollen können nicht nur Rechner, sondern Anwendungen auf Rechnern adressiert werden. Dies geschieht durch Portnummern. Das Transportprotokoll UDP (User Datagram Protocol) hat außer dieser Adressierung keine weiteren Funktionen. Seine Charakteristiken entsprechen daher denjenigen des IP-Protokolls: verbindungslos, unzuverlässig und datagrammorientiert. Im Gegensatz dazu ist das Transportprotokoll TCP (Transmission Control Protocol) verbindungsorientiert, zuverlässig mit Fluss- und Überlastkontrolle sowie datenstromorientiert.

Übungsaufgaben

1.1 Nennen Sie Beispiele für Protokolle der Schicht 5! Geben Sie aber nicht nur den Namen des jeweiligen Protokolls an, sondern beschreiben Sie auch, wozu das Protokoll dient und was die Bedeutung einiger damit übertragener Nachrichten ist!

1.2 Anhand welchen Unterschieds aus der Kurseinheit „Fortgeschrittene Synchronisationskonzepte in Java“ kann man sich den Unterschied zwischen Datagrammorientierung und Datenstromorientierung verdeutlichen?

2 Socket-Schnittstelle

Dieser Abschnitt gibt Ihnen einen Überblick über die Socket-Schnittstelle, ohne auf Details einzugehen.

Lesen Sie Abschnitt 5.2 des Lehrbuchs.

Zusammenfassung

Die Socket-Schnittstelle ist eine Schnittstelle zwischen der Transportschicht (Schicht 4) und der Anwendungsschicht (Schicht 5) und damit in der Regel zwischen dem Betriebssystemkern und den Anwendungen. An Sockets können Portnummern gebunden werden. Alle Daten, die über diesen Socket gesendet werden, tragen diese Portnummer als Quellportnummer. Außerdem kommen an diesem Socket nur Daten an, die an die dazugehörige Portnummer als Zielportnummer adressiert sind. Bei der Kommunikation zwischen zwei Anwendungen ergreift einer der Partner die Initiative und sendet die erste Nachricht; dieser Partner wird als Client bezeichnet, der andere als Server. Aus diesem Grund muss der Client die IP-Adresse und die Portnummer des Servers kennen. Das Umgekehrte gilt nicht, da der Server durch die empfangene Nachricht die IP-Adresse und die Portnummer des Absenders erfährt.

Es gibt keine 1:1-Beziehung zwischen Sockets und Portnummern auf einem Rechner, sondern es ist im Allgemeinen möglich, dass an mehrere Sockets dieselbe Portnummer gebunden ist. Bei TCP ist dies immer dann der Fall, wenn eine zweite Verbindung von einem Verbindungs-Socket angenommen wird, bevor die erste beendet wurde. Wenn es zu einem Zeitpunkt mehrere Verbindungs-Sockets gibt, wobei diese alle von demselben Socket zur Verbindungsannahme stammen, dann besitzen alle diese Verbindungs-Sockets dieselbe Portnummer. Eingehende Nachrichten werden im TCP-Fall aber nicht nur aufgrund der Zielportnummer verteilt, sondern die IP-Adresse und die Portnummer des Absenders werden dabei ebenfalls berücksichtigt.

Die Klasse InetAddress repräsentiert eine IP-Adresse und den dazugehörigen DNS-Namen. Mit Static-Methoden dieser Klasse können in einfacher Weise DNS-Abfragen vorgenommen werden.

Übungsaufgaben

2.1 Schreiben Sie ein Programm zur Abfrage von IP-Adressen, wobei die Methode getAllByName der Klasse InetAddress benutzt werden soll! Beim Starten des Programms sollen beliebig viele Kommandozeilenargumente angegeben werden können, die der Reihe nach an getAllByName übergeben werden. Überlegen Sie bitte auch, wie Sie auf eine UnknownHostException am besten reagieren! Es ist nicht sinnvoll, das ganze Programm sofort abzubrechen. Besser wäre es, weitere Kommandozeilenargumente, falls noch welche vorhanden sind, zu bearbeiten.

2.2 Erweitern Sie Ihr Programm aus der vorigen Aufgabe so, dass Sie zu jedem abgefragten Rechner auch angeben, ob dieser erreichbar ist!

3 Kommunikation über UDP mit Java-Sockets

In diesem Abschnitt wird es konkret. Sie lernen, wie Sie Ihre erste verteilte Anwendung programmieren, die über UDP kommuniziert.

Lesen Sie Abschnitt 5.3 des Lehrbuchs.

Zusammenfassung

Mit den Klassen DatagramPacket und DatagramSocket können Anwendungen entwickelt werden, die über UDP kommunizieren. Die Klasse DatagramSocket enthält Methoden zum Senden und Empfangen von Objekten der Klasse DatagramPacket. Bei Anwendungen, die auf UDP basieren, muss man beim Programmieren die Unzuverlässigkeit von UDP mit in Betracht ziehen. Wird beim Empfangen keine Frist angegeben, kann ein Programm im Falle eines Paketverlusts hängen bleiben.

Übungsaufgaben

3.1 Können mehrere Clients gleichzeitig auf einem Rechner gestartet werden? Können mehrere Server gleichzeitig auf einem Rechner gestartet werden?

3.2 Der Client setzt zuerst den Zähler auf 0 und sendet dann N-Mal das Kommando „increment“. Trotzdem kann man nicht sicher sein, dass die letzte Antwort, die der Client vom Server erhält, N ist bzw. dass der Zählerstand auf dem Server unmittelbar danach auf N steht. Nennen Sie mehrere unterschiedliche Gründe für diesen Effekt! Geben Sie für jeden Grund an, ob der zuletzt erhaltene Wert bzw. der Stand des Zählers auf dem Server größer oder kleiner sein kann als N!