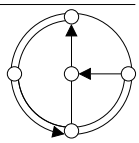
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



SS 2005 Prof. R. Wattenhofer / Keno Albrecht / Angela Nicoara

Vernetzte Systeme
Übung 3

Ausgabe: **14. April 2005**
Abgabe: **22. April 2005**

Bitte schreiben Sie immer Ihre(n) Namen auf die Lösungsblätter.

Vorbemerkung zu den praktischen Übungen

In den praktischen Übungen werden Sie die Gelegenheit haben, einige der in der Vorlesung besprochenen Aspekte selbstständig auszuprobieren und umzusetzen. Zur Implementierung sollten Sie eine der beiden Programmiersprachen Eiffel oder Java verwenden. Auf unserer Webseite finden Sie weitere Informationen, die Ihnen beim Einstieg in die Netzwerkprogrammierung helfen sollen.

Die Aufgabenstellungen geben Ihnen meist das Grundgerüst einer Klasse vor. Die dort enthaltenen leeren Prozedur-/Methodenrümpfe sollen von Ihnen ausgefüllt werden, um das jeweilige Programm zum Laufen zu bringen. Studieren und verstehen Sie aber unbedingt den kompletten Quellcode. Auch die von uns vorgegebenen Teile sind wichtig für das Verständnis und prüfungsrelevant! Als Abgabe schicken Sie bitte alle Dateien an IhrEn ÜbungsleiterIn. Vermerken Sie in der Email alle Namen der beteiligten Studierenden (max. 4)!

1 Registrierung beim Server

In dieser Aufgabe werden Sie den ersten Teil eines einfachen Instant Messengers (IM) schreiben. Der Zweck eines IMs besteht darin, Textnachrichten direkt (Peer-to-Peer) zwischen verschiedenen Benutzern auszutauschen. Im Gegensatz zu einer Email, die an eine feste Adresse (etwa `ihr.name@student.ethz.ch`) versendet wird, können sich die Adressen eines IM-Clients allerdings ändern, da der Computer z.B. bei der Einwahl über ein Modem eine dynamische IP-Adresse des jeweiligen Internetdienstleisters zugeteilt bekommt. Doch woher weiss ein IM-Client, wohin eine Nachricht gesendet werden soll? Dazu beinhaltet ein IM-System typischerweise neben den Clients auch einen Server, bei dem sich die aktiven Clients mit ihrer aktuellen IP-Adresse registrieren müssen. Möchte ein Benutzer eine Nachricht an einen Bekannten senden, so muss erst die derzeitige Adresse des Empfängers beim Server ermittelt werden.

In dieser Aufgabe geht es darum, die Registrierung des Clients beim Server zu schreiben und die Liste aller aktiven Benutzer auszugeben. In den folgenden Übungen werden Sie den IM um weitere Funktionen erweitern. Wir haben zwei IM-Server zum Testen aufgesetzt: Der eine läuft auf `dcg.ethz.ch` (Java) und der andere auf `astra.ethz.ch` (Eiffel). Der Einsatz von Java und Eiffel soll Sie darauf aufmerksam machen, dass Ihre Entwicklung unabhängig von der Wahl der Programmiersprache und der Ausführungsumgebung ist, da wir ein fest definiertes Protokoll verwenden. Testen Sie Ihre Lösung auf jeden Fall mit beiden Servern!

Laden Sie von unserer Webseite das Grundgerüst für den **RegistrationClient** entsprechend Ihrer gewählten Programmiersprache herunter, und implementieren Sie die gekennzeichneten Stellen. Der Ablauf des Programms sieht vor, dass Sie eine Nachricht an den Server schicken, um sich dort zu registrieren. Anschliessend erhalten Sie eine Liste aller vor kurzem registrierten Benutzer zurück, die Sie ausgeben sollen. Drucken Sie die Liste bitte auch aus, und reichen Sie diese bei Ihrem Übungsleiter ein (max. 20 Einträge genügen).

Da wir den IM in Zukunft noch erweitern werden, ist es notwendig, bereits jetzt ein einfaches Nachrichtenformat einzuführen. Dieses besteht aus einem einzigen Byte, einer Typenkennung, das jeder Nachricht vorausgesendet wird. Anhand dieser kann die Gegenstelle erkennen, um was für eine Nachricht es sich handelt, und entsprechend darauf reagieren. Für diese Übung benötigen wir zwei Nachrichtentypen:

- `REGISTER` hat den Wert 0x01 und wird zur Registrierung an den Server geschickt.
- `USERLIST` hat den Wert 0x03 und wird an den Client geschickt, der darauf eine Liste aller aktiven Clients erhält.

Lösen Sie nun die folgenden Teilaufgaben mit Hilfe der aufgeführten Hinweise und der Kommentare im Quellcode:

a) Implementieren Sie die Prozedur `write16`.

Datenströme arbeiten auf unterster Ebene character- bzw. byteorientiert. In dieser Teilaufgabe sollen Sie eine 16-Bit-Zahl in zwei 8-Bit-Zahlen aufteilen. Beachten Sie dabei nur die unteren (less significant) beiden Bytes und keine Vorzeichen. Übertragen Sie erst das höherwertige (more significant) der beiden Bytes und danach das niederwertige. Haben Sie also zum Beispiel eine 32-Bit-Zahl in Hexadezimalschreibweise als 0xAABBCCDD gegeben, so extrahieren Sie die Bytes 0xCC und 0xDD und übertragen zuerst 0xCC und dann 0xDD.

b) Implementieren Sie die Prozedur `read16`.

Stellen Sie die 16-Bit-Zahl, die mit `write16` in zwei 8-Bit-Zahlen aufgeteilt wurde, wieder her.

c) Implementieren Sie die Prozedur `registerUser`.

Schicken Sie eine Nachricht des Typs `REGISTER` an den Server. Anschliessend schicken Sie unter Verwendung der bereits implementierten `write16`-Prozedur Ihren als Kommandozeilen-Parameter übergebenen Port. Danach folgt die Länge Ihres Namens als ein Byte (die Länge darf also 255 Bytes nicht überschreiten), und zum Schluss folgt Ihr Name selbst. Die Nachricht sieht somit wie folgt aus:

Nachrichtentyp	Nachrichtenformat (Anzahl der Bytes)
REGISTER	Typ(1) — Port(2) — Länge des Namens(1) — Name(max. 255)

d) Implementieren Sie die Prozedur `receiveUserList`.

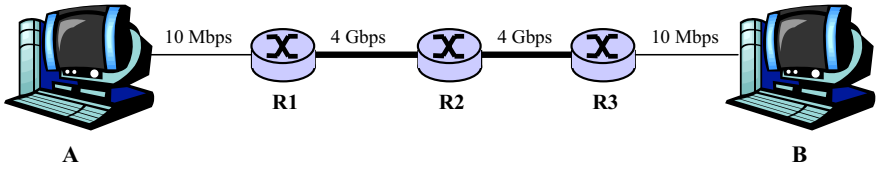
Sie empfangen eine Nachricht des Typs `USERLIST` vom Server. Als nächstes folgt die Anzahl der aktiven Benutzer und dann **für jeden Benutzer** zuerst die Länge seines Namens als ein Byte, der Name selbst und schliesslich die IP-Adresse und der Port, über welche dieser anzusprechen ist. Beachten Sie, dass die IP-Adresse wie folgt übertragen wird: Lautet die Adresse a.b.c.d, so wird zuerst a, gefolgt von b, c und d gesendet. Denken Sie auch daran, dass IP-Adressen Zahlen aus dem Bereich 0-255 enthalten, ein Byte aber u.U. (Java) vorzeichenbehaftet ist! Geben Sie die Benutzerliste aus, und reichen Sie einen Ausdruck von max. 20 Einträgen bei Ihrem Übungsleiter ein. Kompakt sieht das Nachrichtenformat wie folgt aus:

Nachrichtentyp	Nachrichtenformat (Anzahl der Bytes)
USERLIST	Typ(1) — Anzahl Benutzer(2) — { Länge des Namens(1) — Name (max. 255) — IP(4) — Port(2) }*

2 Paketvermittlung

Sie haben in der Vorlesung gehört, dass Nachrichten in Pakete aufgeteilt werden. In dieser Aufgabe sehen wir verschiedene Auswirkungen der gewählten Paketgrösse.

Nehmen wir an, wir wollen 10 MB Daten von Rechner A zu Rechner B übermitteln. Der uns zur Verfügung stehende Weg sei vorgegeben und führe über drei Router, die Pakete gemäss "store-and-forward" (siehe Kapitel 1, Folie 20) weiterleiten. Die Verbindungen am Rand haben jeweils eine Bandbreite von 10 Mbps und im Kern 4 Gbps. Wir betrachten in dieser Aufgabe nur Übertragungsverzögerungen und vernachlässigen alle anderen Verzögerungen (also Ausbreitungs-, Warteschlangen- und Knotenverzögerung).



a) Wie lange dauert die Übertragung von A nach B, wenn die Daten als ein grosses Paket übertragen werden?

b) Wie lange dauert die Übertragung von A nach B, wenn wir die Daten als 100 gleich grosse Pakete übertragen?

c) Die Übertragung in kleineren Paketen ist schneller als in grösseren. Kann man Pakete beliebig klein machen? Überlegen Sie sich, warum das in der Praxis nicht möglich ist, und notieren Sie Ihre Idee.

d) Nehmen Sie an, bei der Übertragung können Pakete verloren gehen oder Fehler auftreten. Würden Sie eher grosse oder kleine Pakete übertragen wollen? Argumentieren Sie.

3 Das HTTP-Protokoll

a) Im Internet werden viele Dienste auf Basis des **Client-Server**-Prinzips angeboten. Dabei stellt der Client eine Anfrage an den Server in Form einer Nachricht. Daraufhin bearbeitet der Server die Anfrage und schickt eine Antwort, ebenfalls in Form einer Nachricht, an den Client zurück. Ein Beispiel für einen auf dem Client-Server-Prinzip basierenden Dienst ist das WWW. Die Anfragenachricht enthält die Adresse der gewünschten Webseite, als Antwort wird die entsprechende Seite zurückgeschickt.

Ihre Aufgabe: Dokumentieren Sie die Schritte, die nötig sind, um die Homepage der Vorlesung "Vernetzte Systeme"

`http://dcg.ethz.ch/lectures/ss05/vs/index.html`

unter Verwendung von `telnet` herunterzuladen. Kopieren Sie dazu Ihre Terminal-Eingaben und die erhaltenen Antworten in eine Textdatei, die Sie ausdrucken, kommentieren und abgeben.

b) In der Vorlesung haben Sie erfahren, dass ein HTTP Server eine Anfrage mit verschiedenen Status Codes beantworten kann (Folie 2/19). Eine genaue Beschreibung dieser Codes und der Header-Zeilen finden Sie in den RFCs 1945 (HTTP 1.0) und 2616 (HTTP 1.1), z.B. unter `http://www.faqs.org`.

In dieser Aufgabe sollen Sie versuchen, möglichst viele verschiedene Status-Codes zu erzeugen. Spielen Sie dazu mit den verschiedenen Header-Zeilen herum, und probieren Sie auch fehlerhafte Angaben aus. Suchen Sie sich dazu beliebig viele HTTP-Server, da nicht unbedingt alle Codes von allen Servern erzeugt werden. (Starten Sie mit der Erzeugung eines "404 Not Found" Codes, dieser sollten Ihnen auf jeden Fall gelingen!)

Drucken Sie die von Ihnen mit Hilfe von `telnet` erzeugten Ergebnisse jeweils aus.