4. Übung zur Vorlesung „Vernetzte Systeme“ WS 2001/2002

Prof. Dr. F. Mattern

Ausgabedatum: 12. Nov. 2001
Abgabedatum: 19. Nov. 2001

Hinweis: Bitte schreiben Sie immer Ihre Übungsgruppennummer und die Namen der beiden Bearbeiter auf die Lösung!

Aktuelle Informationen, Ergänzungen, sowie Korrekturen zu den einzelnen Übungsblättern finden Sie auf der Homepage der Vorlesung `http://www.inf.ethz.ch/vs/edu/WS0102/VS/`. Es lohnt sich, ab und zu auf dieser Seite vorbeizuschauen.

Aufgabe 16 (Protokollhierarchie)

a) (2 Punkte) Nennen Sie zwei wesentliche Argumente für geschichtete Protokolle. Nach welchen Kriterien sollten die Schichten gebildet werden?

b) (3 Punkte) Geschichtete Protokolle erlauben, dass eine Schicht von Aufgaben abstrahieren kann, die von den unter ihr liegenden Schichten bereits übernommen werden. Wovon abstrahieren jeweils die Sicherungsschicht (Data Link Layer), die Vermittlungsschicht (Network Layer) und die Transportschicht (Transport Layer)? Gibt es Aufgaben, die sich nicht in einer einzelnen Schicht erledigen lassen?

c) (2 Punkte) Ein Protokoll-Stack habe eine n-schichtige Hierarchie. Die Applikationen erzeugen Nachrichten von *M* Byte Länge. Auf jeder Schicht wird ein *h* Byte grosser Header hinzugefügt. Welcher Bruchteil der Netzwerkbandbreite wird für Header verwendet? Diskutieren Sie das Ergebnis im Hinblick auf die Anzahl an Schichten, Grösse der Header und Grösse von Nachrichten.

Aufgabe 17 (Stop-and-Wait-Fehlerkontrolle)

Für Zeichen-orientierte (oder Byte-orientierte) Übertragungszenarios wird oft die sogenannte *Stop-and-Wait*-Fehlerkontrolle verwendet: Nach jedem Senden eines Datenpaketes wartet der Sender erst auf eine Bestätigung des Empfängers, bevor er das nächste Datenpaket sendet. Bleibt diese Bestätigung innerhalb einer vordefinierten Wartezeit aus, wird angenommen, dass ein Fehler aufgetreten ist, und das zuletzt gesendete Datenpaket wird erneut gesendet.

Unter der Effizienz *U* einer solchen Verbindung versteht man dann das Verhältnis zwischen der reinen Datenübertragungszeit t_{send} und der Zeit t_{next} die verstreicht, bis das nächste Paket gesendet werden kann (t_{next} ist also die Summe aus Übertragungszeit und der Wartezeit auf die Bestätigung):

$$U = \frac{t_{send}}{t_{next}}$$

Da die Bestätigung normalerweise weitaus kleiner ist als das gesendete Datenpaket, wird t_{next} dabei oft mit $t_{send} + 2d$ approximiert (d ist die Leitungsverzögerung).

a) (4 Punkte) Berechnen Sie die Effizienz der folgenden Verbindungen mit *Stop-and-Wait*-Fehlerkontrolle für einen Durchsatz von jeweils (i) 10 kb/s und (ii) 10 Mb/s:[1]

1. Eine 1 km lange Kupferkabel-Verbindung.
2. Eine 200 km lange Lichtwellenleiter-Verbindung.
3. Eine 50 000 km lange Satellitenverbindung.

Die Paketgrösse sei dabei 10 000 bits, die Fehlerrate sei vernachlässigbar. Beachten Sie bitte, dass die Ausbreitungsgeschwindigkeit der Signale vom Leitungsmedium abhängig ist.

b) (4 Punkte) Eine Verbindung habe einen Durchsatz von 8 kb/s und eine Verzögerung von 30 ms. Für welche Paketgrössen (in bit) hat das *Stop-and-Wait*-Protokoll eine Effizienz von mindestens 50 Prozent?

Aufgabe 18 (URL Download in Java)

In späteren Übungen wollen wir ein wenig (!) praktische Programmier-Erfahrung in dem für vernetzte Systeme wichtigen Gebiet der *Interprozess-Kommunikation* sammeln. Wie aus dem Titel der Aufgabe zu erkennen ist, sollen Sie dazu die Programmiersprache JAVA verwenden.

In dieser Übung geht es zunächst einmal darum, ein Java-Programm erfolgreich zu kompilieren und starten zu können, sowie kleinere Modifikationen am Quelltext zu meistern. Wenn Sie schon ein wenig Erfahrung in einer anderen (prozeduralen oder objektorientierten) Programmiersprache gesammelt haben, sollte dies auch ohne vorherige Java-Kenntnisse nicht allzu schwer sein.

Laden Sie sich dazu den Quelltext (*Source*-Code) eines einfachen Java Web-Clients von der Homepage der Vorlesung herunter (zu finden im Abschnitt „Übungsaufgaben“ unter dem Eintrag für Übung 4):

```
http://www.inf.ethz.ch/vs/edu/WS0102/VS/index.html#aufgaben
```

Speichern Sie das Programm (z.B. in Netscape mit “Save as...”) als `UserAgent.java` in Ihrem Home-Verzeichnis. Bevor Sie es ausführen können, müssen den Quelltext zuerst in maschinenunabhängigen *Byte*-Code übersetzen. Dazu muss auf Ihrem System ein Java-Compiler installiert sein – auf den Rechnern in den Räumen D31 und D35 im IFW-Gebäude ist dieser beispielsweise unter `/usr/bin/javac` zu finden. War die Kompilation erfolgreich, können Sie das Programm dann mit dem Java-Interpreter `java` ausführen:

```
mustermann@rif27> javac UserAgent.java
mustermann@rif27> java UserAgent http://www.inf.ethz.ch/
```

a) (2 Punkte) Kompilieren Sie das Programm und verwenden Sie es dazu, das folgende Web-Dokument herunterzuladen (bitte mit Ausdruck belegen):[2]

```
http://www.inf.ethz.ch/vs/education/WS0102/VS/u4/
```

b) (3 Punkte) Verändern Sie das Programm so, dass anstelle des gesamten Dokuments nur dessen Grösse in Bytes ausgegeben wird, und drucken Sie den geänderten Programmtext aus.

```
mustermann@rif27> java UserAgent http://www.inf.ethz.ch/
6812 bytes
```

[1] Bitte verwenden Sie: 1 k = 10^3; 1 M = 10^6; $b = bit$

[2] Tip: Man kann die Ausgabe von Programmen auf UNIX-Systemen sehr leicht mit Hilfe des “`>`” Zeichens in eine Datei umleiten, z.B. mit “`java UserAgent http://www.inf.ethz.ch/ > test.out`” in eine Datei mit dem Namen `test.out`.