Karlsruher Institut für Technologie

Institut für Telematik
Karlsruher Institut für Technologie (KIT)
Prof. Dr. M. Zitterbart
T. Gerhard, M. Sc.
M. Flittner, M. Sc.
M. Jung, M. Sc.
S. Friebe, M. Sc.

Einführung in Rechnernetze
Sommersemester 2017

3. Übungsblatt

Zu Beginn der dritten Übung findet die Siegerehrung zur Socket-Challenge sowie die Vorstellung der Lösung statt.

Aufgabe 1: HTTP über TCP-Sockets

Hinweis: Diese Aufgabe stand schon auf Übungsblatt 2

Auf dem 1. Übungsblatt wurde ein einfacher, HTTP-basierter Chatserver vorgestellt: Die URL `http://i72tmdjango.tm.kit.edu:8000/chat` akzeptiert sowohl GET- als auch POST-Anfragen. In einer POST-Anfrage sollten die Parameter `user` und `message` vorhanden sein, sodass man eine Chat-Nachricht an den Gruppenchat als der angegebene User senden kann. Ferner lehnt dieser Server jede HTTP-Anfrage, die einen User-Agent beinhaltet, ab.

Schreiben Sie ein Clientprogramm für diesen Chatserver in einer Programmiersprache Ihrer Wahl. Dem Benutzer sollen die Nachrichten im Gruppenchat angezeigt werden, und man soll eigene Nachrichten versenden können. Es bleibt Ihnen überlassen, ob das Programm ein Kommandozeilen-Tool ist, ein textbasiertes Programm, oder ein Programm mit graphischer Oberfläche.

Nutzen Sie zur Netzkommunikation die Socket-API, und senden Sie die Nachrichten über TCP-Sockets. Verzichten Sie dabei auf fertige HTTP-Bibliotheken!

Es ist empfehlenswert, zuerst Erfahrung mit der GET-Methode zu sammeln, indem Sie zunächst nur die Lesefunktion implementieren. Lassen Sie dabei Raum für eine Schreibfunktionalität, die sie im Anschluss nachrüsten können. Schreiben Sie dazu eine Funktion, welche einen beliebigen String als Request versenden und die Antwort des Servers zurückgeben kann, und übergeben Sie dieser Funktion wahlweise den entsprechenden GET- oder POST-Request.

Hinweis: Seien Sie fair zu Ihren Kommilitonen und überlasten Sie nicht den Server.

Aufgabe 2: ARQ-Verfahren, Auslastung, Leistungsbewertung

Hinweise zu den folgenden Aufgabenteilen: Peta = 10^{15} (Abkürzung P), Tera = 10^{12} (Abkürzung T), Giga = 10^{9} (Abkürzung G), Mega = 10^{6} (Abkürzung M), Kilo = 10^{3} (Abkürzung k); die Ausbreitungsgeschwindigkeit betrage 200.000 km/s.

Zwei Kommunikationspartner tauschen über ein Übertragungsmedium miteinander Daten aus. Die Datenrate des Mediums betrage 6 MBit/s und die Ausbreitungsverzögerung zwischen den Teilnehmern betrage 20 Millisekunden. Die Datenübertragung soll durch den Einsatz von ARQ-Verfahren gesichert werden.

Hinweise: Verarbeitungszeiten sollen vernachlässigt werden. Es treten, soweit nicht anders erwähnt, keine Übertragungsfehler auf.

(a) Wozu dienen ARQ-Verfahren? Kann durch ein ARQ-Verfahren ein zuverlässiger Dienst bereitgestellt werden?

(b) Es soll eine Datenmenge von 2 MByte mit Hilfe des Stop-and-Wait-Verfahrens übertragen werden. Die maximale Größe eine Pakets betrage 750 Byte. Wie lange dauert die gesamte Datenübertragung, von Beginn bis zum Zeitpunkt an dem der Sender sicher davon ausgehen kann, dass die Übertragung korrekt abgeschlossen wurde?

Hinweis: Quittungen seien vernachlässigbar klein.

(c) Welche Auslastung des Mediums wird in Teilaufgabe (b) erreicht?

(d) Wie wird beim Go-back-N-Verfahren die Übertragung beschleunigt?

(e) Berechnen Sie die Auslastung, die im gegebenen Beispiel aus Teilaufgabe (b) unter Verwendung des Go-back-N-Verfahrens erreicht wird, bei einer statischen Fenstergröße von 3.000 Byte und alternativ einer Größe von 120.000 Byte. Erläutern Sie den Einfluss der Fenstergröße.

(f) Nehmen Sie an, dass ein Paket nicht mehr 750 Byte groß ist, sondern dass es möglich ist, die komplette Fenstergröße von 120.000 Byte aus Teilaufgabe (e) an einem Stück, also in einem einzigen Paket, zu senden. Berechnen Sie die Auswirkungen auf die Auslastung und erklären Sie das Ergebnis.

(g) Welche Auslastung erreicht das Szenario mit Stop-and-Wait aus Teilaufgabe (b), falls bei der Übertragung der Pakete eine Verlustwahrscheinlichkeit von 10% besteht und wie sieht die Situation bei 30% Verlustwahrscheinlichkeit aus? Gehen Sie davon aus, dass Quittungen immer korrekt ankommen.

(h) Beschreiben Sie kurz, wodurch sich das Verfahren *Selective Reject* von *Go-back-N* unterscheidet.

Aufgabe 3: Sequenznummern, Flusskontrolle

(a) Sie möchten den Sliding-Window-Algorithmus (mit Selective-Repeat-Verfahren) mit Sende-/Empfangsfenstergröße 4 und maximal 5 Sequenznummern ausführen. Das N-te Paket[1] $\mathrm{P}_N[\mathrm{S} = i]$ enthält folglich $i = (N \mod 5)$ in seinem Sequenznummernfeld. Führen Sie ein Beispiel auf, bei dem der Algorithmus ein »unerwünschtes« Ergebnis liefert, d. h. ein Szenario, bei dem der Empfänger das fünfte Paket ($\mathrm{P}_5[\mathrm{S} = 0]$) erwartet, aber das nullte Paket mittels $\mathrm{P}_0[\mathrm{S} = 0]$ (das die gleiche Sequenznummer hat) entgegennimmt. Zeichnen Sie hierfür ein Weg-Zeit-Diagramm.

[1] beginnend mit Paket 0

Hinweis: S = Sende-Sequenznummer, wie in der Vorlesung. Es sollen keine Reihenfolgevertauschungen auftreten, sondern ein ungünstiges Zusammenspiel zwischen Quittung und Timeout-Mechanismus.

(b) Wieviele unterschiedliche Sequenznummern sind bei Selective Repeat abhängig von der Sende- und Empfangsfenstergröße W mindestens nötig, damit das in Teilaufgabe (a) beschriebene Problem nicht mehr auftreten kann?

(c) Bei der Verwendung des Sliding-Windows ist es möglich, dies gleichzeitig für die Ausführung der Flusskontrolle einzusetzen. Wie kann dies erreicht werden? Welchen grundlegenden Nachteil hätte ein solches Vorgehen?

Aufgabe 4: Transportschicht

Die Webseite des Instituts für Telematik (*http://www.tm.kit.edu*) wird von einem Webserver mit der IP-Adresse 141.3.70.4 bereitgestellt, der Anfragen auf TCP-Port 80 erwartet. Eine Informatik-Studentin ruft an ihrem Rechner mit der IP-Adresse 217.237.151.142 die Website in ihrem Browser auf, welcher mehrere HTTP-Requests (für Text und Bilder) zu dem Webserver sendet. Da ihre Verbindung sehr langsam ist, nutzt sie die Zeit, um sich den Vorgang auf der Transportschicht vorzustellen. Können Sie gemeinsam mit ihr die folgenden Fragen klären?

Hinweise: Es ist hilfreich, die Überlegungen mit Hilfe des Tools **Wireshark**[2] anzustellen und zu überprüfen. DNS-Abfragen seien an dieser Stelle vernachlässigt.

(a) Durch welches 5-Tupel kann ein HTTP-Request der Studentin eindeutig beschrieben werden? Gibt es mehrere richtige Antworten? Warum (nicht)?

(b) Wie sieht das 5-Tupel der zugehörigen HTTP-Response aus?

(c) Der Studentin fällt auf, dass sich in ihrem Browser mehrere Bilder parallel aufbauen. Wie ist das möglich? Was bedeutet dies für die geöffneten Sockets?

Aufgabe 5: TCP

Angenommen, es besteht eine TCP-Verbindung, über die Host A eine große Datei an Host B sendet. Geben Sie bei den folgenden Aussagen an, ob sie wahr oder falsch sind.

(a) Host B hat keine Daten um sie an Host A zu schicken. Host B wird deshalb keine ACKs zu Host A senden, da er keine Datenpakete hat, in denen er die ACKs senden kann.

(b) Die Größe des `RcvWindow` ändert sich während der Verbindung nicht.

(c) Die Anzahl der von Host A gesendeten unbestätigten Bytes kann nicht größer sein als die Größe des Empfangspuffers.

(d) Wenn die Sequenznummer für ein Segment dieser Verbindung `m` ist, dann ist sie für das nächste Paket zwangsweise `m + 1`.

(e) TCP-Segmente haben im TCP-Kopf ein Feld `RcvWindow`.

[2] *http://www.wireshark.org/*

(f) Host A sendet Host B ein Segment mit der Sequenznummer 38, welches 4 Bytes Daten enthält. Die Quittungsnummer in diesem Segment ist zwangsweise 42.

Aufgabe 6: Internet-Prüfsumme

(a) TCP und UDP benutzen das Einerkomplement der Summe für ihre Prüfsummen. Angenommen, Sie möchten die folgenden drei Bytes senden: `01010011`, `01100110`, `01110100`. Wie ist die Prüfsumme? *Hinweis: Normalerweise summieren TCP und UDP 16-Bit-Werte, während Sie in dieser Aufgabe 8-Bit-Werte aufsummieren sollen.*

(b) Warum wird das Einerkomplement der Summe verwendet und nicht die Summe selbst? Wie erkennt der Empfänger mithilfe des Einerkomplements Fehler?

(c) Kann ein 1-Bit Fehler unbemerkt bleiben? Was ist mit einem 2-Bit Fehler?

(d) Angenommen ein Host empfängt ein Segment mit der Prüfsumme `01011101 11110010`. Der Host addiert die 16 Bit Wörter über alle notwendigen Felder *außer der Prüfsumme* und erhält den Wert `00110010 00001101`. Wird dieses Segment als korrekt oder fehlerhaft betrachtet? Was wird der Empfänger tun?

Aufgabe 7: Hamming Code

Aus der Vorlesung ist der (7,4) Hamming Code bekannt. Im Rahmen dieser Aufgabe soll nun ein Hamming Code entwickelt werden, der 11 Bit an Nutzdaten prüfen kann.

(a) Wie viele Paritätsbits werden für den Code benötigt?

(b) Erstellen Sie eine Hamming-Tabelle wie aus der Vorlesung bekannt. Markieren Sie, welche Zellen *nicht* zum Prüfen von empfangenen Daten verwenden werden.

(c) Geben Sie die Gleichungen für die Paritätsbits p_1, p_2, ... an.

(d) Gegeben ist das Codewort 100110001101011. Wurde dieses Codewort korrekt empfangen? Verwenden Sie die in der vorherigen Teilaufgabe erstellte Tabelle. Geben Sie die korrekten Nutzdaten an.