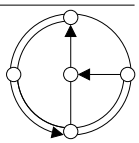
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



SS 2006 Prof. R. Wattenhofer / Michael Kuhn / Yves Weber

Vernetzte Systeme
Übung 7

Ausgabe: **3. Juni 2006**
Abgabe: **Aufgabe 1: 24. Juni 2006**
Aufgaben 2-5: 10. Juni 2006

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

1 IM–Service

In den Übungen 3 und 6 haben Sie einen einfachen Instant Messenger geschrieben. Die Arbeit umfasste die Registrierung beim Server sowie den Austausch von Nachrichten untereinander. In dieser Übung sollen Sie sich einen beliebigen Dienst ausdenken und implementieren. Wir haben auf unseren Servern z.B. einen einfachen ECHO-Dienst bereitgestellt, der an ihn gesendete Nachrichten an den Absender zurückschickt.

Der Dienst kann als *Computerbenutzer* angesehen werden, der sich wie gehabt beim Server registriert, aber auf Anfragen automatisch Antworten generiert und zurücksendet. Es ist ein eigenständiges Programm, das unabhängig von Ihrem Instant Messenger lauffähig sein soll. Natürlich kann der Dienst auf den bisherigen Quellcode aufgebaut werden. Wir geben daher in dieser Aufgabe kein neues Rahmengerüst für Sie vor! Sollten Sie mit Ihrer eigenen Lösung der Übung 6 nicht zufrieden sein, so nehmen Sie die von uns bereitgestellte Beispiellösung als Vorlage.

Für einen *menschlichen* Benutzer erscheint der Dienst als ganz normaler Eintrag in der Benutzerliste. Dem Dienst können wie gewohnt Textnachrichten (als `MESSAGE`!) in einem von ihm spezifizierten Format (siehe c)) gesendet werden. Diese Nachrichten werden dann vom Dienst bearbeitet und eine Textnachricht als Antwort zurückgesendet.

Bearbeiten Sie nun die folgenden Teilaufgaben. Reichen Sie Ihre Lösungen per Email oder ausgedruckt bei Ihrem Assistenten ein.

a) Denken Sie sich einen beliebigen Dienst aus, und spezifizieren Sie seine Aufgabe. Welche Eingabe(n) (Parameter) benötigt er? Welche Ausgabe produziert er?

b) Nehmen Sie sich Ihre Lösung oder unsere Beispiellösung als Vorlage, und entwickeln Sie daraus ein Programm, das sich wie in Übung 4 wiederholt beim Server registriert (TCP) und Nachrichten empfangen und senden kann (UDP). Entfernen Sie unnötigen "Ballast" – eine GUI oder ein Menü ist wahrscheinlich nicht mehr notwendig! Der Dienst soll sich beim Server mit dem Benutzernamen `SERVICE:<service-name>` anmelden. Dabei steht `<service-name>` für einen aussagekräftigen, aber kurzen Namen des Dienstes, der so gewählt werden sollte, dass er unter allen Lösungen eindeutig ist.

c) Sehen Sie in Ihrem Code vor, dass Sie bei Empfang einer Nachricht mit dem Text "help" eine kurze Beschreibung Ihres Dienstes zurücksenden. Geben Sie auch an, welche Nachricht(en) Ihr Dienst erwartet, um seine Aufgabe(n) auszuführen und welche Parameter benötigt werden.

d) Implementieren Sie Ihren Dienst! Wenn Sie die von Ihnen erwartete Nachricht mit gültigen Parametern empfangen, dann bearbeiten Sie diese und schicken das Ergebnis als Nachricht an den Absender zurück. Empfangen Sie eine ungültige Nachricht, so senden Sie einfach die Beschreibung des Dienstes zurück. Seien Sie kreativ!

2 Flusssteuerung

Wenn ein Sender schneller sendet, als der Empfänger die empfangenen Daten verarbeiten kann, werden diese meistens auf Empfängerseite gepuffert. Pufferspeicher sind jedoch begrenzt. Um ein Überlaufen des Puffers zu vermeiden, beinhalten Transportprotokolle eine (typischerweise auf der Sicherungsschicht angesiedelte) **Flusssteuerung**. Die Aufgabe der Flusssteuerung ist es, den Empfänger vor einem zu grossen Zufluss von Paketen eines Senders und damit auch vor dem Überlaufen des Puffers zu schützen. Eine einfache Flusssteuerung kann mittels **Halt**- und **Weiter**-Nachrichten realisiert werden, die der Empfänger dem Sender in entgegengesetzter Richtung des Datenflusses übermittelt.

a) Welcher Zusammenhang besteht zwischen dem Produkt von Verzögerung V und Bandbreite R einer Verbindung und der benötigten Puffergrösse?

b) Angenommen, der Empfänger schickt dem Sender sofort eine Halt-Nachricht, sobald er merkt, dass er mit der Verarbeitung der Daten nicht mehr nachkommt. Wie gross muss der Puffer des Empfängers mindestens dimensioniert sein, damit keine Pakete verloren gehen?

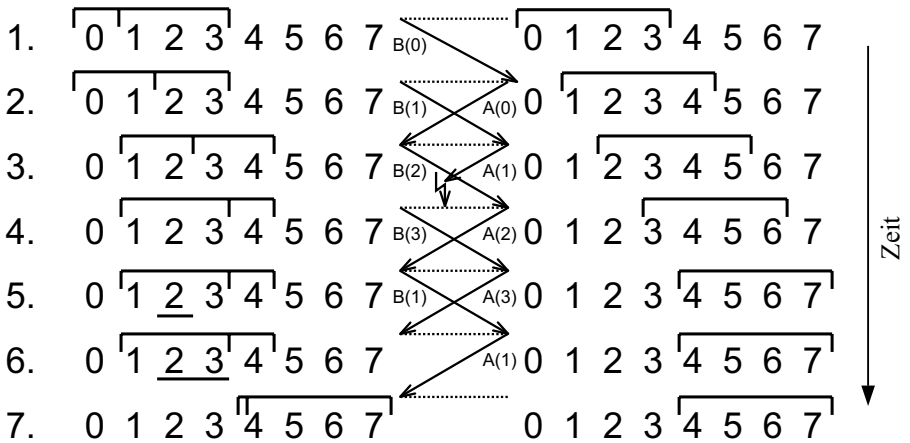
c) Im Internet können die verschiedenen Teilstrecken (Hops) einer Verbindung von A nach B mit verschiedenen Technologien wie Modem- oder Satellitenverbindungen realisiert sein. Welche Probleme ergeben sich, wenn die Teilstrecken unterschiedliche Bandbreiten haben?

3 Schiebefenster und Sequenznummern

In der Vorlesung haben Sie *go-Back-N* und *selective repeat* als zwei Formen für zuverlässige Datentransferprotokolle mit Pipelining kennen gelernt. Pipelining bedeutet dabei, dass der Sender nacheinander mehrere Pakete verschicken kann und diese erst nach und nach vom Empfänger bestätigt (acknowledged) werden können. Damit dies funktioniert, verwenden diese Protokolle auf Sender- und Empfängerseite Puffer, sogenannte Schiebefenster (sliding windows). Für zuverlässigen Datentransfer (damit die Reihenfolge der Pakete erhalten bleibt und kein Paket verloren geht) werden Sequenznummern (sequence numbers) verwendet.

a) In dieser Aufgabe spielen Sie selber einmal das Selective-Repeat-Protokoll durch. Folgendes Beispiel zeigt den Zeitablauf für das Sliding-Window-Protokoll mit Sende- und Empfangspuffergrösse 4, bei dem die Bestätigung des Blocks B(1) verloren geht. Pro Zeiteinheit wird maximal ein Paket versandt. Im Sendefenster sind zusätzlich die Blöcke markiert, die schon bestätigt wurden. Als Fehlerbehandlungsverfahren wird *selective repeat* mit einem Timeout von 3 Zeiteinheiten verwendet, d.h. falls ein gesendeter Block nach drei Zeiteinheiten nicht bestätigt ist, wird er nochmals gesendet.

In jedem Schritt (zum Zeitpunkt der gepunkteten Linien) wird zuerst ein allfälliges Paket empfangen, dann entschieden, ob und wenn ja welches Paket versandt wird, dann dieses ggf. versandt und der neue Zustand dargestellt. Im Beispiel läuft die Zeit von oben nach unten. Dargestellt ist jeweils der Zustand des Sende- bzw. Empfangsfensters kurz nach dem Zeitpunkt der gepunkteten Linien. Der Abstand zwischen zwei gepunkteten Linien entspricht der Zeiteinheit T. Nachrichten sind jeweils genau eine Zeiteinheit (also T) unterwegs.



Nun zu Ihrer Aufgabe: Es sollen 3 Blöcke mit den Sequenznummern 0-2 übertragen werden. Dazu wird das Sliding-Window-Protokoll mit Sendefenstergrösse 3 und Empfangsfenstergrösse 2 und ein Sequenznummernbereich von [0, 5] verwendet. Als Fehlerbehandlungsverfahren verwenden wir *selective repeat* mit einem Timeout von 3 Zeiteinheiten. Block B(0) geht bei der ersten Übertragung verloren. Auch die Bestätigung A(0) der erneuten Übertragung von B(0) geht verloren. Alle anderen Übertragungen verlaufen erfolgreich.

Zeigen Sie mit Hilfe eines Diagramms analog zum Beispiel oben, wie die Fensterstellung nach jedem Erhalt eines Blocks bzw. einer Bestätigung aussieht. Zeichnen Sie nach jeder Übertragung bzw. Bestätigung den Zustand des Sende- und Empfangsfensters in die Schablone a) auf dem Zusatzblatt zum Übungsblatt ein. Markieren Sie auf Empfängerseite auch bereits empfangene, aber noch nicht weitergereichte Pakete.

b) Angenommen, der Sequenznummernbereich für Aufgabenteil a) sei [0, 3], ansonsten bleibe alles gleich. Zeichnen Sie den Ablauf in die Schablone b) ein. An welcher Stelle (in welcher Zeile) scheitert der Algorithmus? Warum scheitert er?

c) Wie viele Sequenznummern (k) sind mindestens nötig, so dass Probleme wie auf Folie 3/41 nicht vorkommen können? Begründen Sie ihr Resultat. Überlegen Sie sich dazu, wie gross der mögliche Bereich von Sequenznummern im Sende- und Empfangsfenster sein kann. Gehen Sie davon aus, dass Sende- und Empfangsfenster gleich gross sind, beide haben Grösse w. Nehmen Sie weiter an, dass der Empfänger Pakete, die in der richtigen Reihenfolge empfangen werden, direkt der Anwendungsschicht weiter reicht.

4 Minimale Fenstergrösse

Über eine Glasfaserstrecke von 5000 km mit einer Bandbreite von 2 Gbps werden Datenblöcke der Grösse 1000 Byte mit dem Sliding-Window-Protokoll gesendet.

a) Wieviele Blöcke sollte das Sendefenster mindestens fassen, um einen kontinuierlichen Datenstrom zu gewährleisten? Geben Sie zu Ihrer Berechnung die getroffenen Annahmen an.

b) Ist es sinnvoll, das Sendefenster grösser als den in a) gefundenen Wert zu machen?

5 TCP Fenstergrösse und Effizienz

Auf Folie 3/43 ist der TCP-Header dargestellt. In dieser Aufgabe beschäftigen wir uns genauer mit einem Feld davon.

a) Die gewünschte Fenstergrösse wird dem Kommunikationspartner im **Window**-Feld (rcvr-window-size) des TCP-Headers mitgeteilt. Überlegen Sie, ob die Grösse dieses Feldes mit 16 Bits genügend und optimal ist, besonders unter Berücksichtigung grosser Verzögerungen und hoher Bandbreiten. Erkennen Sie ein mögliches Problem? Notieren Sie Ihre Idee.

b) Berechnen Sie die Effizienz einer TCP-Verbindung mit einer Bandbreite von 100 Mbps über einen geostationären Satelliten. Nehmen Sie an, dass immer so viele Bytes gesendet werden, wie es das Window-Feld erlaubt. Wie gross können der tatsächliche Durchsatz und die Effizienz (Utilization) dieser Verbindung maximal sein? Treffen Sie vereinfachende Annahmen und notieren Sie diese.

c) Wie könnte man das Problem in einer Weise lösen, die mit dem ursprünglichen TCP-Protokoll, so wie es in RFC 793 definiert wurde, interoperabel ist.[1]

[1] Hinweis: Der TCP-Header lässt sich um **Options** der Form (Optionstyp, Optionslänge, Parameter) erweitern.