ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prof. F. Mattern / H. Vogt
Übung Verteilte Systeme
D-INFK
WS 05/06

Schriftliche Übungsserie A

Bemerkungen

Diese Übungen dienen zur Vorbereitung auf die schriftliche Prüfung. Sie sind freiwillig und werden nicht bewertet. Insgesamt gibt es zwei schriftliche Übungsserien, die den Vorlesungsteil von Prof. Mattern abdecken.

Eine Besprechung und Diskussion möglicher Lösungen der Aufgabenserie A soll im Dezember während einer Vorlesungsstunde stattfinden. Der genaue Termin wird noch bekanntgegeben. Wenn Sie Fragen oder Bemerkungen zu den schriftlichen Übungen haben, wenden Sie sich bitte an Harald Vogt (`vogt@inf.ethz.ch`).

A1. Topologien und Pfadlängen

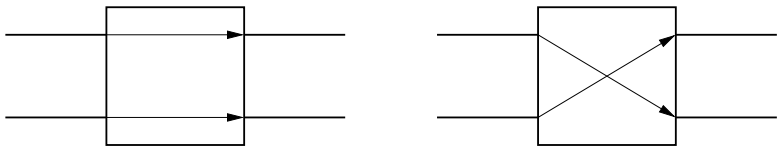
1. Gegeben sei ein 16×16-Gitter mit 256 Netzknoten. Wie gross ist die maximale Pfadlänge?
2. Wenn die 256 Knoten in einem Hypercube angeordnet werden, wie gross ist dann die maximale Pfadlänge?
3. Aus diesem Hypercube wird nun ein Cube Connected Cycle (CCC) erzeugt, indem die Ecken durch Ringe ersetzt werden wie in der Vorlesung angegeben.
   a) Wieviele Knoten enthält der CCC?
   b) Wie gross ist bei diesem CCC die maximale Pfadlänge?

A2. Pfade im Hypercube

1. Wieviele Pfade gibt es zwischen zwei Knoten im Abstand k ($1 \leq k \leq d$) eines Hypercubes der Dimension d?
2. Wieviele dieser Pfade sind knotendisjunkt (d.h. sie haben keine gemeinsamen Knoten ausser dem Anfangs- und Endknoten)? Beweis?

A3. Permutationsnetze

1. In den Vorlesungsunterlagen ist ein Permutationsnetz mit $\log n$ Stufen mit jeweils $n/2$ Schaltern abgebildet (ein Ω-Netz). Die Anzahl der Eingänge (und Ausgänge) ist dabei mit n bezeichnet. Begründen Sie, warum $\log n$ Stufen (mit jeweils $n/2$ Schaltern) notwendig sind, um alle Verbindungen herstellen zu können.
2. Typischerweise gibt es in Permutationsnetzen diese beiden Schaltelemente:



   Lässt sich damit ein Broadcast implementieren? Falls nicht, mit welchen Schaltelementen wäre das möglich?

A4. Fehlermodelle

Im Abschnitt "Kommunikation" der Vorlesung werden verschiedene Fehlermodelle beschrieben (fehlerhaftes Senden, Empfangen, Übertragen, Crash, Fail-Stop, Zeitfehler, Byzantinische Fehler).

Durch welche Fehlermodelle werden die folgenden Anwendungsfälle am besten charakterisiert? Geben Sie auch jeweils an, wen oder was Sie unter einer Nachricht und dem Empfänger bzw. Sender einer Nachricht verstehen.

1. Bei der Anfrage an einen Webserver wird ein Dokument (HTML-Seite) nicht gefunden.
2. Die Batterie eines GSM-Telefons ist leer.
3. Auf einem Rechner, der an einem Peer-to-peer-Netz teilnimmt, hat sich ein spezialisierter Virus eingenistet, der den P2P-Verkehr beobachtet und bestimmte Zugriffe sperrt.
4. Die WLAN-Verbindung eines Laptops ist instabil und bricht immer wieder für kurze Zeit ab.
5. Wegen Überlastung eines Mailservers kommt wichtige E-Mail verspätet beim Empfänger an.
6. Der Spam-Filter eines E-Mail-Clients verschiebt wichtige E-Mail in einen Spam-Ordner, wo sie der Benutzer übersieht.
7. Ein Drucker druckt den Text von Postscript-Dateien aus, statt den Postscript-Code zu interpretieren.

A5. Fehlertoleranz

Wir wollen die Verfügbarkeit eines Dienstes erhöhen. Wenn der Dienst auf einem einzigen Rechner läuft, stehe er 60 % der Zeit zur Verfügung.

1. Welche Verfügbarkeit erreichen wir, wenn wir den Dienst auf 3 Rechnern replizieren?
2. Wieviele Replikate brauchen wir, um eine Verfügbarkeit von 99 % zu erhalten?
3. Wieviele Stunden im Jahr steht der Dienst **nicht** zur Verfügung, wenn wir 99 % Verfügbarkeit erreicht haben?

A6. Implementierung eines Puffers zur asynchronen Kommunikation

Laden Sie das Programmgerüst für diese Aufgabe von der Vorlesungswebseite herunter: `http://www.vs.inf.ethz.ch/edu/WS0506/VS/`.

Es ist eine Klasse `SyncPort` gegeben, mit der zwei Threads synchron kommunizieren können, d.h. ein Aufruf auf einem Objekt dieser Klasse (send oder receive) blockiert so lange, bis das Gegenstück vom anderen Thread aufgerufen wurde.

Mit diesem Kommunikationsmechanismus soll ein Puffer implementiert werden, der asynchrone Kommunikation erlaubt. Der Puffer soll eigene send- und receive-Methoden bereitstellen. Ein send-Aufruf soll sofort zurückkehren (sofern Platz im Puffer ist), ohne auf den Empfänger zu warten. Ebenso der receive-Aufruf, auch wenn gerade kein Sender aktiv ist und sofern Daten im Puffer vorhanden sind.

Implementieren Sie folgende Varianten des Puffers:

1. Ein 1-elementiger Puffer.
2. Ein 3-fach kaskadierter Puffer aufbauend auf dem 1-elementigen Puffer.

Schreiben Sie dazu Code für die Klassen `SingleBuffer` und `CascadedBuffer`, die als (nicht-lauffähige) Gerüste vorgegeben sind. Ergänzen Sie notwendige Konstruktoren, Methoden und Interfaces. Beachten Sie, dass der 1-elementige Puffer einen eigenen Thread benötigt.

Führen Sie Ihre Implementierung mit der mitgelieferten Klasse `Test` aus. Als Ausgabe erhalten Sie eine Aufstellung darüber, wie lange Sender und Empfänger durch die Kommunikation blockiert sind. Führen Sie `Test` für die drei Fälle synchrone Kommunikation ohne Puffer, einfacher Puffer, kaskadierter Puffer aus und geben Sie die erhaltenen Zeiten an. Erläutern Sie das Ergebnis.

Achten Sie auf die Korrektheit Ihrer Implementierung. Insbesondere müssen die Daten `1,2,3,...,26,27,9999` beim Empfänger ankommen.

A7. Synchrone Kommunikation mit Mars-Rover?

Ein Gefährt wird zum Mars geschickt, das mit einer Kamera ausgestattet ist und von der Erde aus gesteuert werden kann. Die Entfernung zwischen Erde und Mars betrage 55.7 Mio km (kleinster Abstand zwischen Erde und Mars). Es steht eine Uplink-Verbindung (von der Erde zum Mars) mit einer Bandbreite von 100 bit/s zur Verfügung, sowie eine Downlink-Verbindung (vom Mars zur Erde), die pro Sekunde ein Bild liefert. Mit Steuerbefehlen der Länge 10 bit soll das Gefährt so gelenkt werden, dass es Hindernissen ausweicht.

Ein Hindernis ist auf den Bildern als solches zu erkennen, wenn es maximal 10 m entfernt ist. Die Reaktionszeit des Steuermanns auf der Erde beträgt 4 s. Auf einen Steuerbefehl reagiert das Vehikel sofort, sobald es ihn empfangen hat.

Wie schnell darf das Gefährt maximal fahren, damit einem Hinderniss zuverlässig ausgewichen werden kann?

A8. RPC

1. Ist es möglich, bei RPC-Aufrufen einen Referenztyp als Eingabeparameter zu verwenden? Als Ausgabeparameter? Begründung!

2. Wenn ein Client zu seiner Anfrage nach einem gewissen Timeout keine Bestätigung erhält, wird er die Anfrage wiederholen. Es könnte aber nur die Bestätigungsnachricht verlorengegangen sein, obwohl der Server die Anfrage bearbeitet hat. Welche Gefahr besteht hierbei und wie könnte man ihr begegnen?

3. Mit welcher RPC-Fehlersemantik-Klasse würden Sie das Verhalten eines Paares Websurfer/Webserver beschreiben, wenn der Websurfer eine GET-Anfrage (Lesen einer Webseite) stellt und, wenn nichts angezeigt wird, den "Reload"-Knopf des Browsers drückt, bis die gewünschte HTML-Seite erscheint?

A9. Broadcast

In Abb. 1 sind zwei Broadcast-Fälle dargestellt.

1. Welcher der beiden Fälle ist "atomar"?
2. Besteht eine kausale Abhängigkeit zwischen den Broadcasts von P2 und P3 im Fall A? Zwischen den Broadcasts von P1 und P2 im Fall B?
3. Sind Broadcasts, die über einen zentralen "Sequencer" gesendet werden, notwendigerweise total geordnet? Welche Voraussetzung muss dazu erfüllt sein?

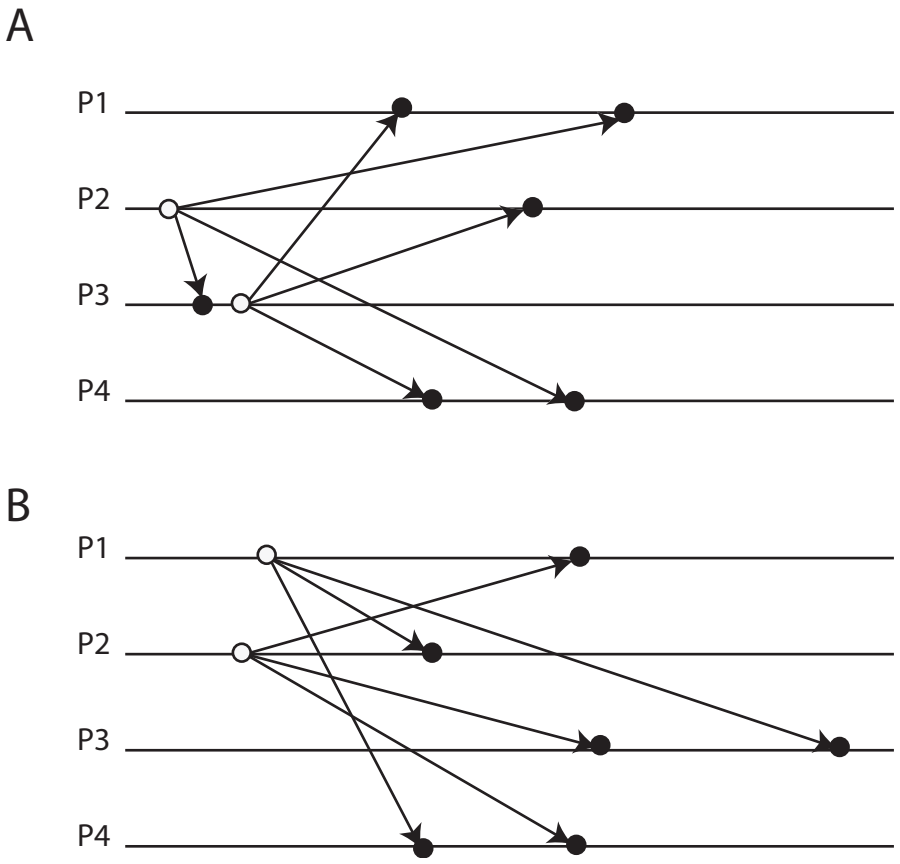


Abbildung 1: Broadcast

A10. Uhrensynchronisation I

Bei zwei Rechnern laufen die internen Uhren unterschiedlich schnell. Bei Rechner A tickt die Uhr korrekterweise 1000 mal pro Sekunde, bei Rechner B nur 990 mal, also etwas zu langsam. Jede Minute erhalten beide Rechner einen Update von einer Zentrale mit der aktuellen UTC-Zeit. Wie gross wird der Unterschied der Uhren von A und B maximal?

A11. Uhrensynchronisation II

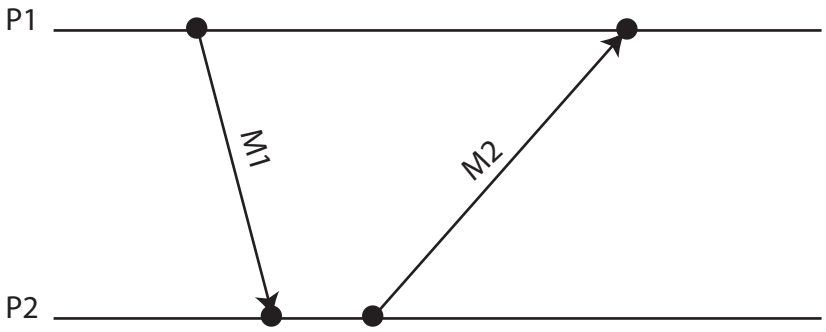


Abbildung 2: Nachrichtenaustausch zur Uhrensynchronisation

In Abb. 2 tauschen die Prozesse P1 und P2 Nachrichten aus, um ihre Uhren zu synchronisieren. Wir gehen von asynchroner Kommunikation aus. Den Prozessen ist bekannt, dass die Laufzeit von Nachrichten zwischen 10 und 35 ms beträgt.

1. Nennen Sie mögliche Gründe für die Varianz in der Nachrichtenlaufzeit.

2. P1 möchte seine Uhrzeit an die von P2 anpassen (P2 führt also die "Referenzzeit"). P1 schickt dazu die Nachricht M1 an P2 mit der Aufforderung, die aktuelle Uhrzeit zu senden. P2 schickt die Nachricht M2 mit einem Zeitstempel, der dem Absendezeitpunkt von M2 entspricht. Mit welcher Genauigkeit kann P1 seine Uhrzeit an die von P2 anpassen (im besten und schlechtesten Fall)?

3. Nun sei die Laufzeit von Nachrichten nicht genau bekannt, lediglich eine Minimallaufzeit von 10 ms. P1 kann aber die Roundtrip-Zeit RTT (mittels M1 und M2) zwischen P1 und P2 messen. Geben Sie den möglichen Wertebereich für die Uhrzeit von P2 für den Zeitpunkt an, an dem die Nachricht M2 bei P1 eintrifft (in Abhängigkeit von der Minimallaufzeit und der Roundtrip-Zeit).

   Auf welchen Wert sollte P1 seine Uhrzeit setzen? Wie gross ist die Genauigkeit (maximale Abweichung vom realen Wert) dieser Grösse?

   Geben Sie die Genauigkeit für den konkreten Fall einer Roundtrip-Zeit von $RTT = 80$ ms an.