ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prof. F. Mattern
Übung Verteilte Systeme
D-INFK
HS 2009

Verteilte Systeme

Theoretische Übungen mit Lösungsvorschlägen

Bemerkungen

Diese Übungen dienen zur Vorbereitung auf die schriftliche Prüfung. Sie sind freiwillig und werden nicht korrigiert. Wenn Sie Fragen oder Bemerkungen zu den schriftlichen Übungen haben, wenden Sie sich bitte an einen der verantwortlichen Assistenten.

Matthias Kovatsch (kovatsch@inf.ethz.ch)
Iulia Ion (iion@inf.ethz.ch)

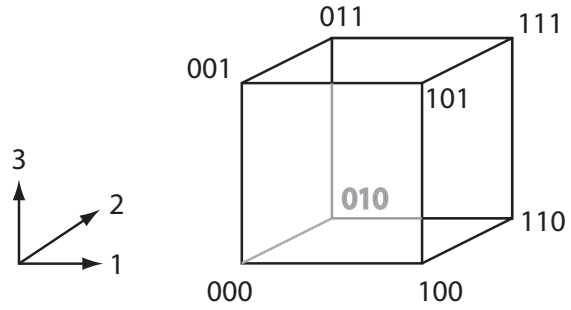
1 Topologien und Pfadlängen

1. Gegeben sei ein 16×16-Gitter mit 256 Netzknoten. Wie gross ist die maximale Pfadlänge?

   Lösungsvorschlag: *30 Schritte (von einer Ecke zur diagonal gegenüberliegenden)*

2. Wenn die 256 Knoten in einem Hypercube angeordnet werden, wie gross ist dann die maximale Pfadlänge?

   Lösungsvorschlag: *Dimension des Hypercubes:* $d = log_2(256) = 8$
   Die maximale Pfadlänge entspricht der maximalen Anzahl an Bitkippern der Adresse
   $\Rightarrow$ *8 Schritte*

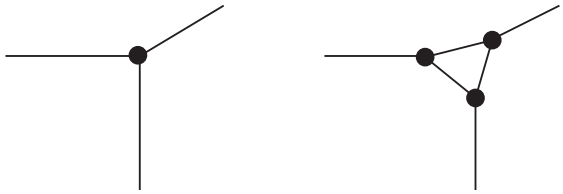


Adressenbeispiel im Hypercube der Dimension 3

3. Aus diesem Hypercube wird nun ein Cube Connected Cycle (CCC) erzeugt, indem wie in der Vorlesung angegeben die Ecken durch Ringe ersetzt werden.

   a) Wieviele Knoten enthält der CCC?

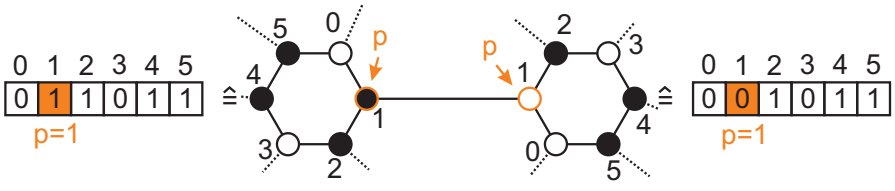
   Lösungsvorschlag: *Ein Hypercube wird zum CCC, indem jede Ecke durch einen Ring mit d Knoten ersetzt wird.*



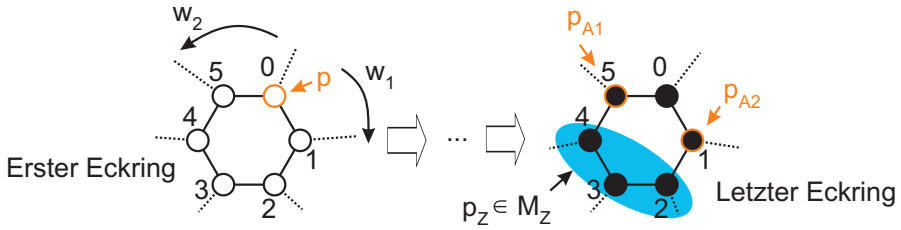
   2^d *Knoten werden zu* $d \cdot 2^d$ *Knoten:* $n = 8 \cdot 256 = 2048$ *Knoten.*

   b) Wie gross ist bei diesem CCC die maximale Pfadlänge?

   Lösungsvorschlag: *Jeder Knoten hat eine CCC-Adresse, bestehend aus Hypercube-Anteil (Adresse des Eckrings) und Position p innerhalb dieses Eckrings. Aus Gründen der Übersichtlichkeit ist hier ein CCC der Dimension 6 dargestellt.*



Pro Schritt hat man nun drei Möglichkeiten: Dimensionswechsel an der Position p, Wechsel zum linken Nachbarn im Ring ($(p-1) \bmod d$) und Wechsel zum rechten ($(p+1) \bmod d$).
Der grösstmögliche Abstand zwischen zwei Knoten ergibt sich wie folgt:
Maximaler Unterschied beim Hypercube-Adressteil, d.h. d verschiedene Bitkipper.
Jeder Dimensionswechsel, ausser der erste, benötigt einen zusätzlichen Schritt im Ring, d.h. $d-1$ Schritte.
Im letzten Eckring muss die Position des Zielknotens p_Z maximal von der Ankunftsposition entfernt sein. Je nachdem, ob man sich in den Ringen im Uhrzeigersinn (w_1) oder entgg. dem Uhrzeigersinn (w_2) bewegt hat, ergeben sich zwei mögliche Ankunftspositionen, p_{A1} bzw. p_{A2}.



p_Z muss in der Menge M_Z enthalten sein, um maximal entfernt zu sein. Die maximale Weglänge im letzten Ring ist also $\left\lfloor \frac{d-2}{2} \right\rfloor = \left\lfloor \frac{d}{2} \right\rfloor - 1$.
Dies entspricht der Anzahl Schritte bis zur Mitte des Kreisbogens M_Z.
Insgesamt erhält man also: $d + (d-1) + \left\lfloor \frac{d}{2} \right\rfloor - 1 = 2d + \left\lfloor \frac{d}{2} \right\rfloor - 2$

Anmerkung: Die *maximale Pfadlänge* bezeichnet die Länge des längsten Pfades aus der Menge aller kürzesten Pfade (keine Umwege).

2 Pfade im Hypercube

1. Welchen Abstand haben die beiden Knoten $a = (0,1,1,0,1,0)$ und $b = (1,1,0,0,1,1)$ eines 6-dimensionalen Hypercubes?

   Lösungsvorschlag: *Zwei Knoten mit Abstand k unterscheiden sich in k Bits ihrer Adresse. Er lässt sich wie folgt berechnen:*
   $\delta = \sum_i |a_i - b_i| = |0-1| + |1-1| + |1-0| + |0-0| + |1-1| + |0-1| = 3$

2. Wieviele Pfade gibt es zwischen zwei Knoten im Abstand k $(1 \leq k \leq d)$ eines Hypercubes der Dimension d?

   Lösungsvorschlag: *Um vom Start- zum Zielknoten zu gelangen, müssen nacheinander die Bits gekippt werden, in denen sich ihre Adressen unterscheiden. Der Pfad entsteht entlang der durchlaufenen Knoten. Im ersten Schritt gibt es k Möglichkeiten, im zweiten (k - 1), usw.*
   Das ergibt $k!$ Pfade.

3. Wieviele dieser Pfade sind knotendisjunkt (d.h. sie haben keine gemeinsamen Knoten ausser dem Anfangs- und Endknoten)? Beweis?

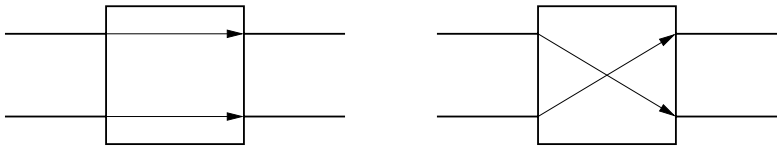
Lösungsvorschlag: *Maximal kann es nur k knotendisjunkte Pfade geben, denn der Startknoten kann im ersten Schritt zum Ziel nur zwischen k Nachbarn wählen.*
Es gibt aber auch mindestens k Pfade. Ein Pfad ist gegeben durch eine Folge von Dimensionswechseln, z.B. 1,4,3,2 (zuerst 1. Bit kippen, dann 4., dann 3. und dann das 2.). Die Anordnung dieser Dimensionswechsel muss so erfolgen, dass sich keine gemeinsamen Zwischenknoten auf verschiedenen Pfaden ergeben. Eine passende Anordnung erhält man durch Rotation der Dimensionswechsel:
Start mit: $n_1, n_2, ..., n_k$
Rotieren um eine Stelle nach links: $n_2, n_3, ..., n_k, n_1$
Um zwei Stellen: $n_3, ..., n_k, n_1, n_2$
Usw. bis zu k-1 Stellen: $n_k, n_1, ..., n_{k-1}$
Dies sind k knotendisjunkte Pfade.
Andersgesagt, die Präfixe der Pfade dürfen nicht aus den gleichen Elementen bestehen. Die Folgen 1,4,3 und 4,1,3 würden einen gemeinsamen Zwischenknoten ergeben, da nach zwei Schritten die gleichen Bits gekippt wurden.

3 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.

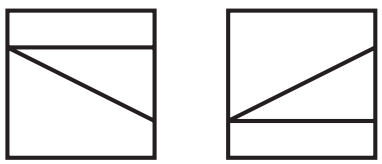
   Lösungsvorschlag: *Pro Stufe verdoppelt sich die Anzahl der Nachfolger, die durch Verschalten erreicht werden können, so dass nach m Stufen 2^m Positionen erreichbar sind. Mit $m = \log_2 n$ Stufen erreicht man folglich $2^{\log_2 n} = n$ Ausgänge.*

2. Typischerweise gibt es in Permutationsnetzen diese beiden Schaltelemente:



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

   Lösungsvorschlag: *Ein Broadcast ist mit Standard-Schaltelementen nicht möglich, denn dort ist pro Eingang nur ein Ausgang erreichbar. Die folgenden Elemente erlauben, einen Eingang mit beiden Ausgängen zu verknüpfen, wodurch Broadcast möglich wird:*



4 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.

   Lösungsvorschlag: *Fail-Stop. Sender: Browser, Empfänger: Server. Der Server antwortet mit einer Fehlermeldung, d.h. der Sender erfährt, dass ein Fehler aufgetreten ist und die Anfrage nicht bearbeitet werden kann.*

2. Die Batterie eines GSM-Telefons ist leer.

   Lösungsvorschlag: *Crash/Fail-Stop. Sender: GSM-Netz/anderer Teilnehmer, Empfänger: Mobiltelefon. Das Netz erkennt den "Verlust" durch einen Timeout.*

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.

   Lösungsvorschlag: *Byzantinischer Fehler. Das Verhalten des Virus ist beliebig. Dass ein Fehler auftritt, kann nicht in jedem Fall festgestellt werden.*

4. Die WLAN-Verbindung eines Laptops ist instabil und bricht immer wieder für kurze Zeit ab.

   Lösungsvorschlag: *Fehlerhaftes Senden und/oder Empfangen. Hier ist der Link selbst betroffen, nicht einer der Kommunikationsteilnehmer.*

5. Wegen Überlastung eines Mailservers kommen wichtige E-Mails verspätet beim Empfänger an.

   Lösungsvorschlag: *Zeitfehler. Der Empfänger (Benutzer) erhält die Nachricht zu spät.*

6. Der Spam-Filter eines E-Mail-Clients verschiebt wichtige E-Mails in einen Spam-Ordner, wo sie der Benutzer übersieht.

   Lösungsvorschlag: *Fehlerhaftes Empfangen. Die Nachricht (Mail) wird vom Empfänger (Spam-Filter) falsch behandelt. Der Sender merkt davon nichts.*

7. Ein Drucker druckt den Text von Postscript-Dateien aus, statt den Postscript-Code zu interpretieren.

   Lösungsvorschlag: *Byzantinischer Fehler oder fehlerhaftes Empfangen. Entweder hat der Druckertreiber einen Bug (byz. Fehler) oder bei der Übertragung zum Drucker gehen Daten verloren, so dass der Drucker die Postscript-Datei nicht als solche erkennt.*

5 Kommunikation

1. Wie kann es bei synchroner Kommunikation zwischen zwei Prozessen zu einem Deadlock kommen?

   Lösungsvorschlag: *Durch gleichzeitiges, gegenseitiges Warten, z.B. wenn sich zwei Teilnehmer gleichzeitig einen Auftrag schicken.*

2. Bei welchem Kommunikationsmechanismus besteht nur eine geringe Gefahr für Deadlocks? Begründen Sie Ihre Antwort.

   Lösungsvorschlag: *Bei mitteilungsorientierter, asynchroner Kommunikation wird nie gewartet, also kann es prinzipiell zu keinem gleichzeitigen, gegenseitigen Warten kommen.*

3. Warum bevorzugen Programmierer trotzdem RPC?

   Lösungsvorschlag: *Bei RPC macht es im Code keinen Unterschied, ob ein Objekt lokal oder entfernt ist. Programmierer können also ihre gewohnten, prozeduralen/objektorientierten Lösungsmuster verwenden.*

4. Was ist der Unterschied zwischen synchroner, mitteilungsbasierter und synchroner, auftragsorientierter Kommunikation ohne Rückgabewert?

   Lösungsvorschlag: *Im ersten Fall wartet der Sender nur, bis eine Bestätigung des Eingangs der Mitteilung vorliegt. Im zweiten Fall wartet er, bis auf der Empfängerseite der Auftrag tatsächlich abgearbeitet wurde.*

6 RPC

1. Ist es möglich, bei RPC-Aufrufen einen Zeiger als Eingabeparameter zu verwenden (“call by reference”)? Als Ausgabeparameter? Begründung!

   Lösungsvorschlag: *Referenzen sind grundsätzlich nicht erlaubt, da eine Referenz (Pointer) von Rechner A auf Rechner B keine Bedeutung hat. Das gilt sowohl für Eingabe- wie Ausgabeparameter. Man könnte jedoch die Datenstruktur hinter der Referenz sequentialisieren und in dieser Form übertragen.*

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?

   Lösungsvorschlag: *Es besteht die Gefahr, dass eine Anfrage mehrfach bearbeitet wird. Sequenznummern helfen, allerdings nicht falls der Grund für die ausbleibende Bestätigung ein Server-Absturz war. Eine andere Möglichkeit wären idempotente Funktionen.*

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? ***Lösungsvorschlag:*** *At-least-once*

7 Broadcast

In Abbildung 1 sind zwei Broadcast-Fälle dargestellt.

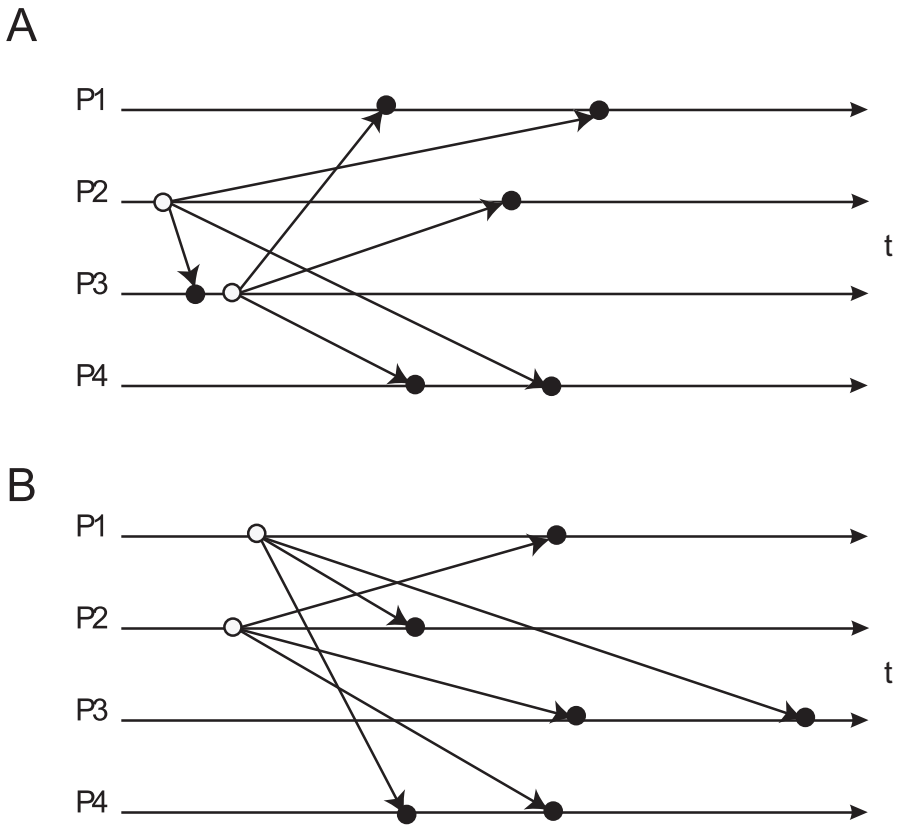


Abbildung 1: Broadcast

1. Welcher der beiden Fälle ist "atomar"?

   Lösungsvorschlag: *Def. atomarer (totaler) Broadcast: Wenn P_i und P_j die Nachrichten N,M erhalten, ist die Empfangsreihenfolge bei beiden Prozessen die gleiche.*
   Beachte: Das Senden einer Nachricht zählt nicht als Empfang!

   A: Empfangsreihenfolge bei P1 und P4 gleich $\Rightarrow$ atomarer Broadcast
   B: Empfangsreihenfolge bei P3 und P4 unterschiedlich $\Rightarrow$ kein atomarer Broadcast

2. Welchen Vorteil haben atomare Broadcasts gegenüber nicht-atomaren?

   Lösungsvorschlag: *Nicht atomare Broadcasts gefährden die Konsistenz von replizierten Variablen. Die Illusion einer speicherbasierten Kommunikation kann nicht realisiert werden.*

3. Besteht eine kausale Abhängigkeit zwischen den Broadcasts von P2 und P3 im Fall A? Zwischen den Broadcasts von P1 und P2 im Fall B?

Lösungsvorschlag: *Simpler Test auf kausale Abhängigkeit: Gibt es einen Pfad (von links nach rechts), der vom Sendeereignis von X zum Sendeereignis von Y führt?*

A: Ja! Aber: Kausale Abhängigkeit ist nicht gewahrt, denn die Reihenfolge bei den Empfängern P1 und P4 entspricht nicht der Kausalität!
B: Nein, keine kausale Abhängigkeit der Broadcasts

4. Sind Broadcasts, die über einen zentralen "Sequencer" gesendet werden, notwendigerweise total geordnet? Welche Voraussetzung muss dazu erfüllt sein?

   Lösungsvorschlag: *Falls die Kanäle vom Sequencer zu den Empfängern die FIFO-Eigenschaft besitzen, dann gilt totale Ordnung, denn: Wenn Nachricht X vor Y gesendet wird, dann kommt X vor Y an (Nachrichten überholen sich im FIFO-Kanal nicht).*
   Die FIFO-Eigenschaft kann z.B. über Acknowledgements hergestellt werden (Sequencer braucht aber ein ACK von allen Empfängern).

8 Lamport-Zeit

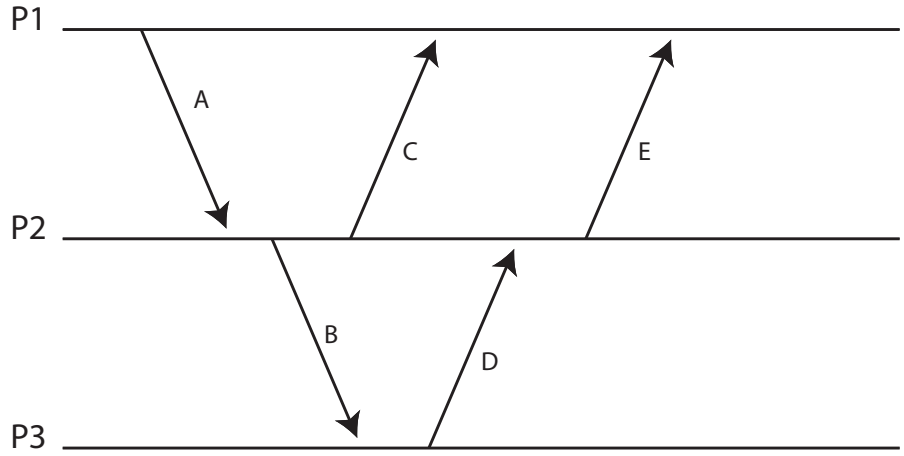


Abbildung 2: Zeitdiagramm

Im folgenden bezeichnet $\prec$ die Kausalrelation auf Ereignissen ("happened before"), C ist die Abbildung von Ereignissen auf Zeitstempel (die durch natürliche Zahlen repräsentiert werden).

1. Geben Sie ein Paar von Ereignissen aus Abb. 2 an, über deren kausale Abhängigkeit keine Aussage getroffen werden kann.

   Lösungsvorschlag: *Beispiele sind: (B.receive, C.receive), (C.receive, E.send)*

2. Fügen Sie in Abb. 2 eine Nachricht N ein, für die gilt

   $$\text{A.receive} \prec \text{N.send} \ \wedge C(\text{N.receive}) < C(\text{E.send})$$

   wobei Sie Absender und Empfänger der Nachricht (die unterschiedlich sein sollen) frei wählen können, sofern die Bedingung erfüllt ist.

   Lösungsvorschlag: *Es gibt eine Reihe von Lösungsmöglichkeiten. Beispiel 1: N von P2 an P3, N.send zwischen B.send und C.send, N.receive zwischen B.receive und D.send Beispiel 2: N von P2 an P1, N.send zwischen C.send und D.send, N.receive nach C.receive*

3. Fügen Sie auf ähnliche Art eine Nachricht M ein, für die gilt

   $$\text{M.send} \prec \text{C.send} \ \wedge C(\text{B.receive}) < C(\text{M.receive})$$

   und die von P1 gesendet und von P2 empfangen wird.

   Lösungsvorschlag: *Analog zum ersten Beispiel aus 2.)*

Bemerkung: Die Notation X.send bzw. X.receive bezeichnet das send- bzw. receive-Ereignis der Nachricht X.

9 Lamport-Zeit – Wechselseitiger Ausschluss

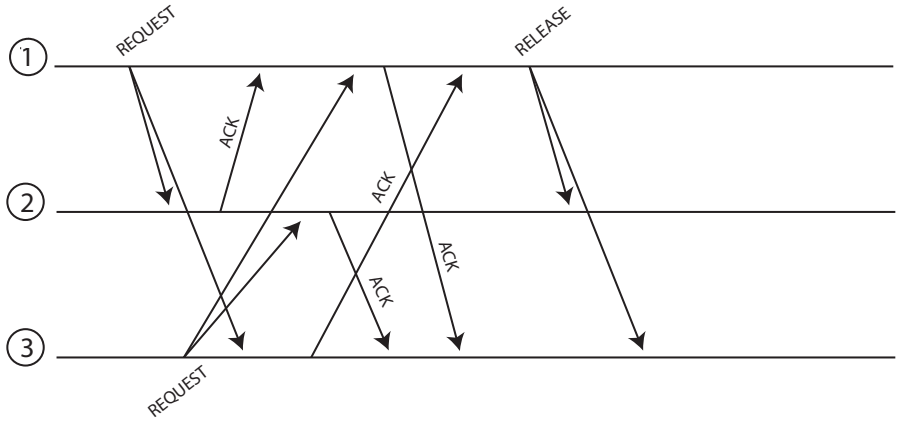
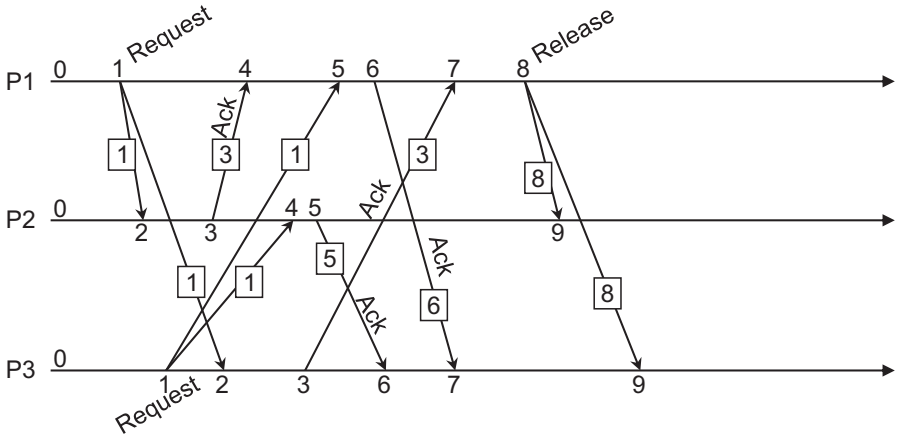


Abbildung 3: Wechselseitiger Ausschluss mit Lamport-Zeit

In Abb. 3 ist ein Zeitdiagramm dargestellt mit Nachrichten von drei Prozessen. Prozesse 1 und 3 bewerben sich um den exklusiven Zugriff auf eine gemeinsame Ressource. Die Prozesse wenden das aus der Vorlesung bekannte Verfahren zum wechselseitigen Ausschluss an, das Lamport-Zeit und verteilte Warteschlangen benutzt.

1. Geben Sie die Sende- und Empfangszeitstempel für jedes Ereignis an.

   Lösungsvorschlag:



2. Geben Sie für die Prozesse 1 und 3 an, wie die Warteschlange des jeweiligen Prozesses nach jedem Sende- bzw. Empfangsereignis aussieht.

   Lösungsvorschlag: *Q1: Request(P1,1); Ack(P2,3); Request(P3,1); Ack(P3,3);*
   Q2: Request(P1,1); Request(P3,1); Release(P1,8);
   Q3: Request(P3,1); Request(P1,1); Ack(P2,5); Ack(P1,6); Release(P1,8);

3. Sind beim Einreihen in die Warteschlange der Sende- oder der Empfangszeitstempel zu verwenden? Warum?

   Lösungsvorschlag: *Die Sendezeitstempel, da sonst keine global-konsistente Sicht garantiert werden kann. Zum Test die Warteschlangen geordnet nach Empfangszeitstempel:*
   Q1: Request(P1,1); Ack(P2,4); Request(P3,5); Ack(P3,7);
   Q2: Request(P1,2); Request(P3,4); Release(P1,9);
   Q3: Request(P3,1); Request(P1,2); Ack(P2,6); Ack(P1,7); Release(P1,9)

4. Welche Bedingung muss erfüllt sein, damit Prozess 1 auf die Ressource zugreifen kann?

   Lösungsvorschlag: *Eigener Request hat niedrigsten Zeitstempel, ACKs (oder andere Nachrichten) von allem anderen Teilnehmern mit höheren Zeitstempeln erhalten.*

5. Markieren Sie den Zeitpunkt im Zeitdiagramm, zu dem Prozess 1 bzw. Prozess 3 auf die Ressource zugreifen kann.

   Lösungsvorschlag: *P1: Nach dem ACK von P3. P3: Nach RELEASE von P1.*

10 Namen

1. Was versteht man unter dem Binden und dem Auflösen von Namen? Nennen Sie ein Beispiel für einen Dienst, der diese Operationen zur Verfügung stellt.

   Lösungsvorschlag: *Binden: Name wird einer Adresse zugewiesen. Auflösen: Adresse eines Namens wird "geholt". Beispiele: DNS oder Telefonbuch.*

2. Was ist der Unterschied zwischen iterativer und rekursiver Namensauflösung?

   Lösungsvorschlag: *Iterativ: Client bekommt Adresse des (nächsten) zuständigen Nameservers von seinem Nameserver mitgeteilt und muss diesen selbst kontaktieren.*
   Rekursiv: Client kontaktiert nur seinen Nameserver, welcher den Namen direkt auflöst, indem er ggf. andere Nameserver kontaktiert.

3. Warum bietet es sich an, Caching bei der Auflösung von Namen einzusetzen und was wird dadurch erreicht?

   Lösungsvorschlag: *Namensbindungen ändern sich selten. Durch Caching wird das Auflösen von häufig angefragten Namen effizienter, da nicht jedes Mal der Nameserver kontaktiert werden muss.*

4. Nehmen Sie an, die Abbildung von Namen auf Objektadressen wird in einem lokalen Cache eines Clients gespeichert. Ein bereits gebundener und im Cache aufgrund einer früheren Anfrage enthaltener Name wird nun an eine andere Adresse gebunden, das alte Objekt bleibt aber weiterhin aktiv.

   a) Welches Problem tritt nun beim Client auf? Kann der Client dieses Problem erkennen?

      Lösungsvorschlag:*Client kommuniziert mit ursprünglichem Objekt und erkennt das Problem nicht.*

   b) Tritt das Problem auch bei solchen Clients auf, die statt der Adresse den zuständigen Nameserver im Cache halten?

      Lösungsvorschlag: *Nein*

5. Zur Erhöhung von Effizienz und Fehlertoleranz werden Nameserver repliziert. Für welche Nameserver ist dies besonders relevant? Begründen Sie Ihre Antwort.

   Lösungsvorschlag: *Server die die höhere Hierarchieebene abdecken sind kritischer, da es hier viel mehr Anfragen gibt und ein Ausfall ganze Teilbereiche betreffen würde.*

11 Client-/Server

1. Bei Web-basierten Diensten wird oft ein Bezeichner in Links codiert ("URL rewriting"), um die aktuelle Transaktion zu identifizieren. Wie könnte ein Unbefugter eine laufende Transaktion "übernehmen" und was kann man gegen diese Gefahr tun?

   Lösungsvorschlag: *Es muss 'nur' eine URL mit einer gültigen Transaktions-ID erzeugt werden. Timeouts erhöhen die Sicherheit.*

2. Welches Problem entsteht bei einem zustandsbehafteten Server, wenn viele Clients abstürzen, bevor sie ihre Transaktionen beendet haben?

   Lösungsvorschlag: *Die Ressourcen auf dem Server werden blockiert, obwohl sie nicht mehr verwendet werden.*

3. Erläutern Sie kurz ein paar Vorteile von zustandslosen gegenüber zustandsbehafteten Client/Server-Protokollen und umgekehrt.

   Lösungsvorschlag: *Vorteile zustandslos: Effizienz, Robustheit des Servers gegen eigenen Crash und Client-Crash*
   Vorteile zustandsbehaftet: Sitzung über mehrere Interaktionen halten, potentiell Reduktion der übertragenen Datenmenge

12 Middleware

1. Wie werden in CORBA verschiedene Programmiersprachen zur Implementierung der Anwendung unterstützt? Welchen Vorteil hat die Unterstützung mehrerer Sprachen?

   Lösungsvorschlag: *Mit Hilfe der Interface Description Language (IDL) können passende Client- und Server-Stubs erzeugt werden. Dadurch können heterogene Systeme miteinander kooperieren (man denke auch an Legacy-Code).*

2. Welche Funktionen übernimmt ein Stub in CORBA?

   Lösungsvorschlag: *Verpacken und Entpacken der Parameter (Marshalling) sowie Abschicken bzw. Entgegennehmen von Methodenaufruf-Mitteilungen über den ORB-Core.*

3. Was ist der ORB? Wo wird er ausgeführt?

   Lösungsvorschlag: *Der Object Request Broker ist das Rückgrat einer CORBA-Implementierung. Er dient als Vermittlungsschicht zwischen verschiedenen CORBA-Objekten, d.h. Weiterleitung von Methodenaufrufen, etc. Der ORB bildet eine Kommunikationsschicht und ist logisch über alle Teilnehmer verteilt.*

4. Warum wird CORBA heutzutage nicht mehr weiterentwickelt?

   Lösungsvorschlag: *Im Skript sind verschiedene Gründe aufgezählt, z.B. die zu weitreichende Anforderungen, fehlende Unterstutzung durch Microsoft, aufkommende Konkurrenzsysteme, die z.T. besser an die neuen Anforderungen angepasst sind.*

5. Was versteht man unter Objekt-Serialisierung? Für was wird sie benötigt? Nennen Sie ebenfalls Beispiele, wie die Serialisierung in Middlewaresystemen realisiert wird.

   Lösungsvorschlag: *Unter Serialisierung versteht man die Konvertierung von Objekten in Byteströme und Rückkonvertierung in eine identische Kopie des Ausgangsobjekts. So können Objekte zwischen verschiedenen Komponenten eines verteilten Systems ausgetauscht werden. Java RMI bietet das Interface "Serializable", wodurch Objekte per ObjectInputStream/ObjectOutputStream (de-)serialisiert werden können. Bei .NET können Klassen mit einem Attribut versehen werden und deren Instanzen so binär oder als XML übertragen werden.*

13 Jini

1. Erläutern Sie kurz die Funktion von Leases.

   Lösungsvorschlag: *Zeitlich befristeter Vertrag über einen Service zwischen zwei Parteien.*

2. Eine Besonderheit von Jini ist das Ausnutzen der Mobilität von Java-Code. Welche Code-Teile werden übertragen und welche Möglichkeiten ergeben sich dadurch?

   Lösungsvorschlag: *Proxy-Objekte werden bei der Registrierung des entsprechenden Services beim Lookup-Service hinterlegt und bei der Nutzung des Services zum Client übertragen. Dies ermöglicht die Verwendung spezieller Protokolle sowie "smart proxies", d.h. zusätzliche Intelligenz in den stubs, z.B. Vorverarbeitung, Datenkompression.*

14 Sicherheit

1. Auf welche Herausforderungen trifft man bei der Schlüsselverteilung in verteilten Systemen?

   Lösungsvorschlag: *Verteilte Systeme sollen "offen" sein, was aber Angriffe fördert, eine zentrale Sicherheitsautorität skaliert nicht und die Heterogenität der Systeme sorgt für zusätzliche Schwachstellen.*

2. Beschreiben Sie zwei möglichen Lösungsansätze zur Schlüsselverteilung.

   Lösungsvorschlag: *Zum einen Schlüsselvergabe über einen sicheren Kanal oder per (aufwändigem) Public-Key-Verfahren, zum anderen direkte Schlüsselvereinbarung per Sicherheitsprotokoll (z.B. Diffie-Hellman).*

3. In der Vorlesung wurde der Diffie-Hellman-Algorithmus besprochen.

   a) Für was wird er verwendet?

      Lösungsvorschlag: *Diffie-Hellman stellt ein kryptografisches Protokoll dar. Es dient zur Erstellung eines geheimen Schlüssels zwischen Kommunikationspartnern über einen unsicheren Kanal.*

   b) Beschreiben Sie kurz das Verfahren.

      Lösungsvorschlag: *Zwei Kommunikationspartner (A und B) kennen beide eine (grosse) Primzahl p und eine Primitivwurzel $c \bmod p$ (mit $2 \leq c \leq p-2$). Diese können wie bei Sun-RPC vorgegeben sein, oder auch über den unsicheren Kanal ausgetauscht werden.*
      A und B wählen je eine Zufallszahl a bzw. b (aus der Menge zwischen 1 und $p-2$), die geheimgehalten werden muss. Aus den gegebenen Werten berechnen die Kommunikationspartner $\alpha = c^a \bmod p$ bzw. $\beta = c^b \bmod p$. Diese werden ausgetauscht, d.h. A sendet α an B und B β an A.
      Jetzt können A und B jeweils den gemeinsamen, geheimen Schlüssel berechnen: $G_A = \beta^a \bmod p$ und $G_B = \alpha^b \bmod p$.
      Da $(c^b \bmod p)^a \bmod p = (c^a \bmod p)^b \bmod p$ gilt, gilt auch $G_A = G_B$.

   c) Was ist ein möglicher Angriff und wie könnte man sich dagegen verteidigen?

      Lösungsvorschlag: *Als "man in the middle" könnte man in den Kanal zwischen zwei Kommunikationspartnern eindringen und sich jeweils als Gegenstelle ausgeben. So werden für beide Teilstrecken eigene Schlüssel ausgehandelt und der Angreifer kann die Nachrichten transparent weiterleiten, sie dabei aber mitlesen und auch verändern. Dieser Angriff kann durch das Interlock-Protokoll (siehe Skript S.311) erkannt werden.*

4. Wie funktioniert zertifikatsbasierte Authentifizierung? Von welchem weitverbreiteten System wird sie verwendet?

   Lösungsvorschlag: *Eine vertrauenswürdige Autorität signiert mittels asymmetrischer Verschlüsselung ein Zertifikat, welches die Identität einer Person oder Ressource zusammen*

mit dessen Public-Key enthält. Mittels dem Challenge-Response-Verfahren prüft die Gegenseite dann, ob die Person/Resource wirklich die ist, die sie vorgibt zu sein. Der Zertifikatsinhaber authentifiziert sich, indem er mit seinem Private-Key eine bestimmte Nachricht signiert, welche mit dem Public-Key aus dem Zertifikat validiert werden kann.
SSL/TLS verwendet zertifikatsbasierte Authentifizierung.

5. Wenn One-Time-Pads ein perfektes Verschlüsselungssystem darstellen, warum werden diese dann heutzutage nicht global eingesetzt?

   Lösungsvorschlag: *One-Time-Pads sind nur unter der Annahme perfekt, dass die beteiligten Parteien bereits im Vorraus genügend Pads ausgetauscht haben. Sie müssen jeweils die gleiche Länge wie die Nachrichten haben und dürfen nur ein Mal verwendet werden. Diese Voraussetzungen sind kaum zu bewältigen.*

6. Mit Einwegfunktionen lassen sich Einmalpasswörter erzeugen und leicht überprüfen. f sei eine Einwegfunktion und x_1 ein initiales Passwort, aus dem eine Passwortkette erzeugt wird:

   $$x_1 \rightarrow^f x_2 \rightarrow^f \ldots \rightarrow^f x_{n-1} \rightarrow^f x_n$$

   a) Um die Passwörter zur Authentisierung nutzen zu können, muss x_n zunächst zum Server S übertragen werden. Welche der folgenden Anforderungen müssen erfüllt sein:

      i. Ein Angreifer darf nichts über x_n erfahren, die Übertragung muss also geheimnisbewahrend erfolgen.

      ii. Es muss sichergestellt sein, dass x_n bei der Übertragung nicht verändert wird.

      Lösungsvorschlag: *i. nicht erforderlich, ii. erforderlich*

   b) Wir nehmen an, es sei $n = 100$. Dem Server S wird x_{100} bekanntgemacht. Ein Client C schreibt die Werte $x_1, x_2, \ldots, x_{99}$ in eine Liste. Bei der ersten Anmeldung an S verwendet er x_{99} und streicht diesen Wert von der Liste. Beim zweiten Mal verwendet C aus Versehen x_{89} (statt x_{98}). Welche Gefahr besteht, wenn dieser Wert von einem Angreifer abgehört wird und S den Anmeldeversuch einfach ignoriert, weil $f(x_{89}) \neq x_{99}$?

      Lösungsvorschlag: *Man setzt normalerweise voraus, dass die Hashfunktion bekannt ist. Ein Angreifer könnte daher x_{89}, x_{90}, $\ldots x_{98}$ berechnen und einsetzen, d.h. er könnte sich bis zu 11 mal anmelden.*

7. Im Kerberos-Protokoll erhält ein Client vom KDC (Key Distribution Center) ein verschlüsseltes TGT (Ticket Granting Ticket). Kann dieses TGT von einem anderen Client verwendet werden, um vom TGS (Ticket Granting Service) ein ST (Service Ticket) anzufordern? Begründung!

   Lösungsvorschlag: *Kann nicht benutzt werden, da der Schlüssel K nur dem ursprünglichen Client bekannt ist. $TGT = \{Name, Client - ID, t, \Delta t, K, ...\}_{KC}$ Um TGT zu verwenden, ist ein Authentizitätsnachweis erforderlich! Dieser muss mit K verschlüsselt sein. Kenntnis von TGT reicht nicht aus.*

8. Nennen Sie zwei Gründe, warum in Kerberos KDC und TGS getrennt sind.

   Lösungsvorschlag: *Skalierbarkeit, Performance, konzeptionelle Trennung von Akkreditierung und Servicenutzung*