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.

Leyna Sadamori (leyna.sadamori@inf.ethz.ch)

1 Verteilte Systeme

Beschreiben Sie ein konzeptionelles Problem, das nur in verteilten Systemen auftreten kann.

Lösungsvorschlag: *Beispiel Uhrensynchronisation: Die Uhren verschiedener Systeme laufen nicht unbedingt gleich schnell und können auch über die Zeit driften. Somit ist es nicht trivial genaue Zeitpunkte oder zeitliche Ordnung in einem verteilten System zu bestimmen.*

2 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. 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.*

3. 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.*

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

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

5. 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.*

6. 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.*

3 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.*

4 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*

5 Synchrone Kommunikation mit einem 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?

Lösungsvorschlag: *Signallaufzeit Erde-Mars:* $55.7Miokm/300'000km/s = 186s$
Benötigte Zeit um auf ein Hindernis zu reagieren: $2 \cdot Signallaufzeit + 4s Reaktionszeit = 376 Sekunden$
Max. erkennbare Distanz zum Hindernis: 10m
In 376 Sekunden darf der Rover max. 10m zurücklegen, d.h. $vmax = 10m/376s = 0.0266m/s = 96m/h$
Bemerkung: ein echter Mars-Rover ist auf maximal 5 cm/s ausgelegt und bewegt sich real mit ca. 1 cm/s.

6 Modellierung von Web-Schnittstellen

In der Vorlesung wurden zwei verschiedene Paradigmen vorgestellt, nach denen man Web- Schnittstellen modellieren kann: serviceorientiert mittels WS-* und ressourcenorientiert direkt mittels HTTP 1.1. In dieser Aufgabe sollen Sie für beide Paradigmen die Web-Schnittstelle eines internetfähigen digitalen Bilderrahmens modellieren. Der digitale Bilderrahmen kann auf seinem Display sowohl Text als auch Bilder darstellen. Ausserdem unterstützt er die Darstellung von Diashows im Atom[1]- und RSS[2]-Format. Die Darstellung von Slideshows kann weiter konfiguriert werden. So kann man die Darstellungsdauer eines Bildes festlegen, ebenso wie die Darstellungsreihenfolge der Bilder (zufällig/ursprüngliche Reihenfolge). Der Rahmen verfügt ausserdem über drei verschiedene Übergangsmodi zwischen den einzelnen Bildern. Das Display selbst kann in 16 verschiedenen Helligkeitsstufen gedimmt werden.Weiterhin sind im Bilderrahmen noch ein binärer Bewegungsmelder, welcher die Anwesenheit von Personen feststellen kann, und ein Helligkeitssensor, welcher in 256 verschiedenen Stufen auflöst, verbaut.

Modellieren Sie nun für beide Paradigmen die entsprechenden Schnittstellen, welche die nachfolgend beschriebene Funktionalität zur Verfügung stellen:

1. Darstellung eines Textes auf dem Bildschirm (unformatierter Text)
2. Darstellung eines JPEG-Bildes auf dem Bildschirm
3. Darstellung einer Diashow, welche im Atom- bzw. RSS-Format übertragen wird
4. Abfrage der aktuellen Darstellungskonfiguration
5. Abfrage des momentanen Bildschirminhalts (z.B. gerenderter Text oder aktuelles Bild des Feeds)
6. Setzen/Auslesen der Konfiguration der Diashow
   - Anzeigedauer pro Bild in Sekunden
   - Übergangsanimation zwischen den Bildern (keine, überblenden, umblättern)
   - Reihenfolge der Bilder (wie im Feed spezifiziert oder zufällig)
7. Setzen/Auslesen der Bildschirmhelligkeit
8. Auslesen des Bewegungssensors
9. Auslesen des Helligkeitssensors

Beispiel: Nehmen wir an, dass es eine Funktion gibt, mittels derer man den Bildschirm des Bilderrahmens ein- und ausschalten und dessen aktuellen Zustand (ein- oder ausgeschaltet) abfragen kann. Nachfolgend ein Modellierungsvorschlag für beide Paradigmen.

[1] `http://de.wikipedia.org/wiki/Atom_(Format)`
[2] `http://de.wikipedia.org/wiki/RSS`

- Serviceorientiert: (Pseudocode ausreichend)
  - `setDisplayPower( [xsd:boolean] )`
    (`[xsd:boolean]`, `[xsd:integer]` und `[xsd:string]` sind verwendbare Basistypen; wird `[xsd:complexType]` benötigt, muss dieser nicht weiter spezifiziert werden)
  - `[xsd:boolean] getDisplayPower()`
- Ressourcenorientiert:
  - Pfad: `/display/power`
  - Methoden: `GET`, `PUT`
  - Repräsentation: `text/plain`
  - Erlaubte Werte:
    - Display ist eingeschaltet: `true`
    - Display ist ausgeschaltet: `false`

Lösungsvorschlag:

1. *Darstellung eines unformatierten Textes auf dem Bildschirm*
   - *Serviceorientiert:* `showText([xsd:string])`
   - *Ressourcenorientiert:*
     - *Pfad:* `/display/presentation`
     - *Methoden:* `PUT`
     - *Repräsentation:* `text/plain`
     - *Erlaubte Werte: (Beliebiger Text)*
2. *Darstellung eines JPEG-Bildes auf dem Bildschirm*
   - *Serviceorientiert:* `showImage([xsd:complexType image])`
   - *Ressourcenorientiert:*
     - *Pfad:* `/display/presentation`
     - *Methoden:* `PUT`
     - *Repräsentation:* `image/jpeg`
     - *Erlaubte Werte: JPEG-kodiertes Bitmap*
3. *Darstellung einer Diashow, welche im Atom- bzw. RSS-Format übertragen wird*
   - *Serviceorientiert:*
     - `showRSS([xsd:string])`
     - `showATOM([xsd:string])`
     - *Alternativ:* `showFeed([xsd:string])`
   - *Ressourcenorientiert:*
     - *Pfad:* `/display/presentation`
     - *Methoden:* `PUT`
     - *Repräsentationen:* `application/atom+xml`, `application/rss+xml`
     - *Erlaubte Werte: RSS-Feed, ATOM-Feed*

4. *Abfrage der aktuellen Darstellungskonfiguration*
   - *Serviceorientiert:*
     - `[xsd:integer] getShowType()`
     - `[xsd:string] getText()`
     - `[image] getImage()`
     - `[xsd:string] getFeed()`
     - *Alternativ:* `[xsd:complexType presentation] getPresentation()`. `[xsd:complexType presentation]` *kapselt entsprechend die verschiedenen Präsentationsmodi.*
   - *Ressourcenorientiert:*
     - *Pfad:* `/display/presentation`
     - *Methoden:* `GET` *(oder* `HEAD`*, um nur den Modus zu bestimmen)*
     - *Repräsentationen: Hier muss im Request der Accept-Header weggelassen oder auf* `*/*` *gesetzt werden*
     - *Rückgabe: Aktueller Stand, welcher mittels PUT auf derselben Ressource gesetzt wurde*
     - *Enstprechender Content-Type*
     - *Entsprechender Inhalt (Text, JPEG-Bild, RSS-Feed, . . .)*
5. *Abfrage des momentanen Bildschirminhalts*
   - *Serviceorientiert:* `[image] getCurrentImage()`
   - *Ressourcenorientiert:*
     - *Pfad:* `/display/currentimage`
     - *Method:* `GET`
     - *Repräsentation:* `image/jpeg`
     - *Rückgabe: JPEG-Bild des aktuellen Bildschirminhalts*
6. *Setzen/Auslesen der Konfiguration der Diashow*
   - *Serviceorientiert:*
     - `setShowConfiguration([configuration])`
     - `[configuration] getShowConfiguration()`
     - `[configuration]:`
       - `[xsd:integer] delay` *// 0-65535*
       - `[xsd:string] animation` *// none, fade, turn*
       - `[xsd:string] ordering` *// feed, random*
   - *Ressourcenorientiert:*
     - *Pfad:* `/configuration/delay`
     - *Method:* `GET`, `PUT`
     - *Repräsentation:* `text/plain`
     - *Erlaubte Werte: 0-65535*
     - *Pfad:* `/configuration/animation`
     - *Method:* `GET`, `PUT`

  - *Repräsentation:* `text/plain`
  - *Erlaubte Werte: none, fade, turn*

  - *Pfad:* `/configuration/ordering`
  - *Method:* `GET`, `PUT`
  - *Repräsentation:* `text/plain`
  - *Erlaubte Werte: feed, random*

7. *Setzen/Auslesen der Bildschirmhelligkeit*
   - *Serviceorientiert:*
     - `setDisplayBrightness([xsd:integer])`
     - `[xsd:integer] getDisplayBrightness()`
   - *Ressourcenorientiert:*
     - *Pfad:* `/display/brightness`
     - *Method:* `GET`, `PUT`
     - *Repräsentation:* `text/plain`
     - *Erlaubte Werte: 1, 2, 3, ... 16*
8. *Auslesen des Bewegungssensors*
   - *Serviceorientiert:* `[xsd:boolean] isMotion()`
   - *Ressourcenorientiert:*
     - *Pfad:* `/sensors/motion`
     - *Method:* `GET`
     - *Repräsentation:* `text/plain`
     - *Erlaubte Werte: false, true*
9. *Auslesen des Helligkeitssensors*
   - *Serviceorientiert:* `[xsd:integer] getBrightness()`
   - *Ressourcenorientiert:*
     - *Pfad:* `/sensors/brightness`
     - *Method:* `GET`
     - *Repräsentation:* `text/plain`
     - *Erlaubte Werte: 0, 1, 2, ... 255*

7 REST

Ist das ressourcenorientierte Beispiel (und Ihre Lösungen) aus Aufgabe 6 bereits RESTful? Geben Sie jeweils ein Beispiel mit Erklärung, welche Eigenschaften von REST von dieser Web-Schnittstelle erfüllt werden und welche nicht.

Lösungsvorschlag:

- *Erfüllt:*
  - *Client/Server: Der Bilderrahmen ist ein Server, der von Clients gesteuert werden kann.*

  - *Zustandslosigkeit: Der Bilderrahmen speichert nur seine Konfiguration in den Ressourcen und keinerlei Sitzungsinformationen zu seinen Clients. Diese können also beliebige Requests in beliebiger Reihenfolge senden, ohne unerlaubte oder fehlerhafte Konfigurationen zu erzeugen.*
  - *Caching: Es werden vollständige Repräsentation der Ressource übertragen, welche von Caches im Client oder auf dem Pfad zum Server zwischengespeichert werden können. (Siehe auch alternative Argumentation unter "Nicht erfüllt".)*
  - *Einheitliche Schnittstelle: Es werden URIs sowie a-priori bekannte HTTP-Methoden und Internet Media Types verwendet. (Dies hilft allerdings noch nicht bei der semantischen Bedeutung der Schnittstellen.)*

- *Nicht erfüllt:*
  - *Caching: Es fehlen Angaben zur "Cache-Control". Caches müssen wissen, wie lange eine Repräsentation verwendet werden darf ("max-age"). (Die Web-Schnittstelle für den Bildschirm könnte einen Zeitraum von wenigen Minuten nehmen, innerhalb dessen eine Änderung unbedeutend ist, z.B. weil es für eine Berechnung des Energieverbrauchs vernachlässigbar ist.)*
  - *Hypermedia as the Engine of Application State: Es wurden nur fixe URI-Pfade definiert. Dies entspricht noch nicht der REST-Architektur und hat wenig Vorteile gegenüber dem SOA-Ansatz, da man die Clients mit diesen URIs vorprogrammieren muss. (Die Web-Schnittstelle könnte eine Indexseiten anbieten, über die die Konfiguration, Bildschirminhalt und Sensoren mittels Links erreichbar sind. Diese Links können Informationen über die Funktionalität enthalten (`rel`-Attribut) oder die Ressourcen bieten spezielle Media Types an, die die Funktionalität charakterisieren (spezielle `application/[custom]`-Typen oder komplexere, selbstbeschreibende Repräsentationen wie XML oder JSON).)*

8 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.

9 Internet Protokolle

Jeder Aufruf eines Web-Services muss mehrere Protokollschichten durchlaufen, damit er dem Server zugestellt werden kann. Bei einem SOAP-Request mit HTTP-Binding sind das:

- Anwendungsschicht
- WS-*-Transportschicht (HTTP)
- Transportschicht (TCP)
- Vermittlungsschicht (IP)
- (Tiefere Schichten, die hier nicht behandelt werden müssen)

Beschreiben Sie für jede der vier genannten Schichten jeweils

1. in welcher Form der Aufruf verpackt wird,
2. welche Informationen dort jeweils zur Adressierung benötigt werden und
3. woher der Client die jeweilige Information hat.

Lösungsvorschlag:
Anwendungsschicht:

- *Request in SOAP-Envelope (<message> als Body, ggf. mit Header)*
- *Benötigt gültige <message> des Dienstes, <operation>*
- *Informationen aus WSDL (Teile <types>, <messages>, <portType>)*

WS--Transportschicht (HTTP):*

- *SOAP-Envelope (XML) als HTTP-Body*
- *Benötigt Host(name) und Pfad der URI*
- *Informationen aus WSDL (Teil <service>)*

Transportschicht (TCP):

- *HTTP-Stream als TCP-Segmente*
- *Benötigt (Quell- und Ziel-)Ports*
- *Quellport zufällig (Ephemeral-Port), Zielport HTTP-Standardport oder URI-Port aus WSDL*

Vermittlungsschicht (IP):

- *TCP-Segmente als IP-Pakete*
- *Benötigt IP-Adresse (und Route)*
- *DNS-Anfrage mit URI-Host aus WSDL liefert IP.*

10 Broadcast

In Abbildung 1 sind zwei Broadcast-Fälle dargestellt.

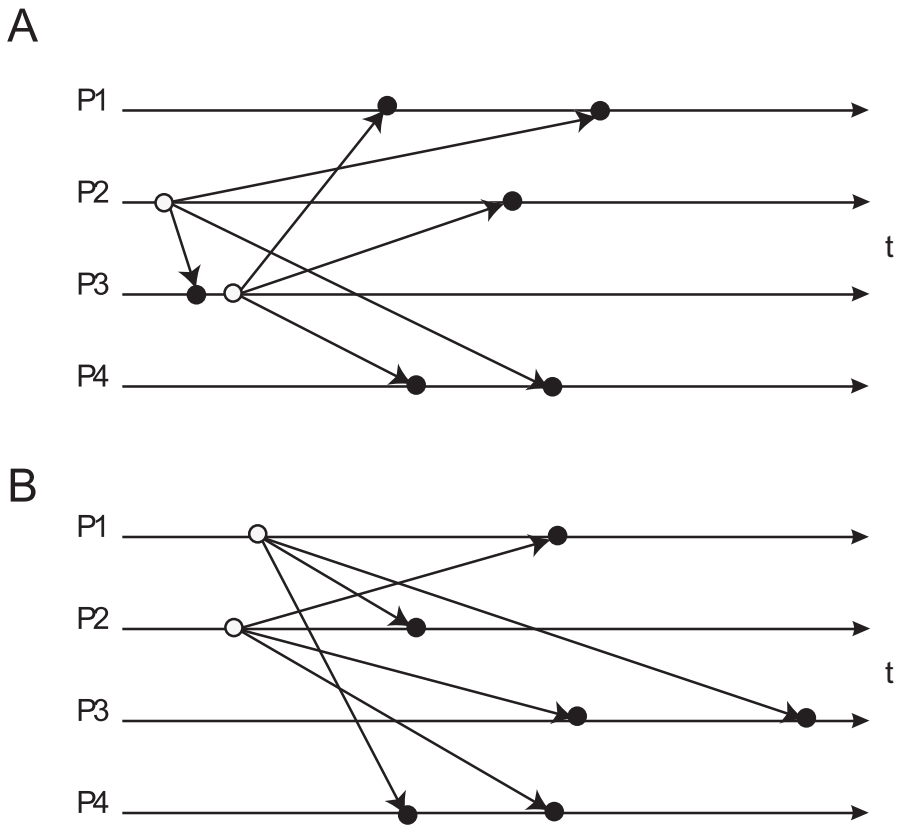


Abbildung 1: Broadcast

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. 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

3. 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).

11 Lamport-Zeit

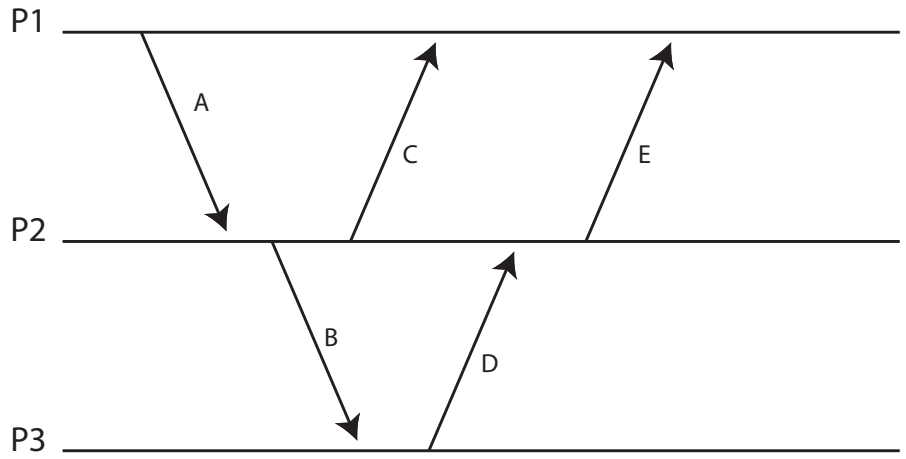


Abbildung 2: Zeitdiagramm

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.receive, 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 P2 gesendet und von P3 empfangen wird.

   Lösungsvorschlag: *Analog zum ersten Beispiel aus Lösung 11.2*

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

12 Lamport-Zeit – Wechselseitiger Ausschluss

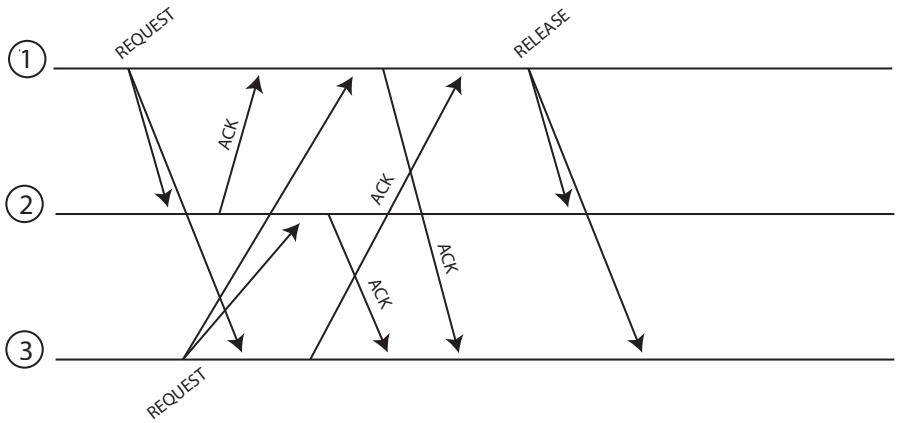
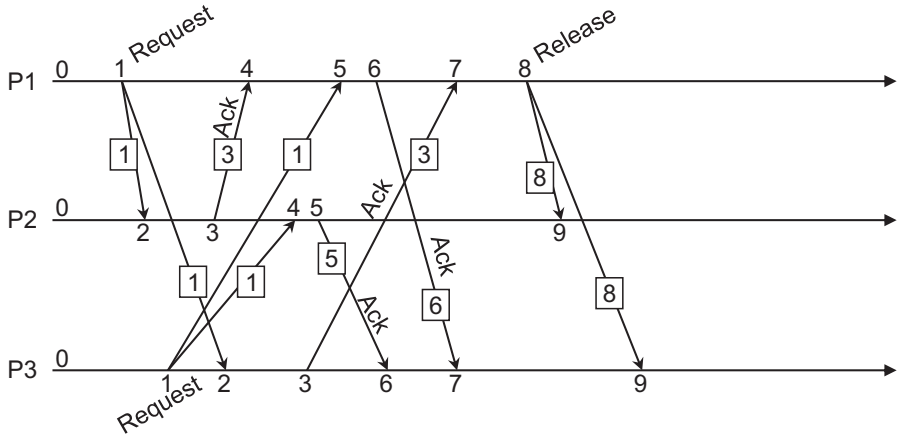


Abbildung 3: Wechselseitiger Ausschluss mit Lamport-Zeit

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:

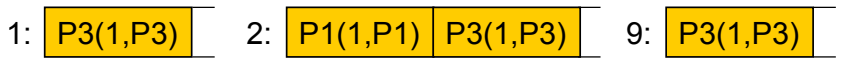


2. Geben Sie für die Prozesse 1 und 3 an, wie die Warteschlange des jeweiligen Prozesses nach jedem Sende- bzw. Empfangsereignis aussieht.

   Lösungsvorschlag: *Beachten Sie die lexikographische Ordnung bei gleichen Zeitstempeln!*

Prozess P1:

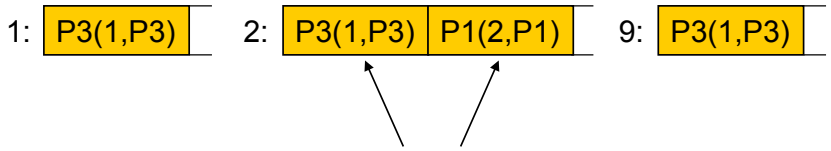
Prozess P3:



Hier Prioritätswarteschlange sortiert nach Zeitstempeln: P1(1,P1) rutscht beim Einfügen an die Spitze.

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:*



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 ($t = 7$).*
   P3: Nach RELEASE von P1 ($t = 9$).

13 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 mögliche 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).*

14 Einwegfunktionen

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:

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

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

   (a) Ein Angreifer darf nichts über x_n erfahren, die Übertragung muss also geheimnisbewahrend erfolgen.

   (b) Es muss sichergestellt sein, dass x_n bei der Übertragung nicht verändert wird.

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

2. Wir nehmen an, es sei $n = 100$. Dem Server S wird x_{100} bekannt gemacht. 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.*

15 One-Time-Pads

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 Voraus 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.*

16 Diffie-Hellman

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

1. 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.*

2. 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 sendet 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 $G_A = G_B$.

3. 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 erkannt werden.*

17 Zertifikate

Wie funktioniert zertifikatsbasierte Authentifizierung? Von welchem weitverbreiteten Protokoll 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/Ressource 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 (und darauf basierend HTTPS) verwendet zertifikatsbasierte Authentifizierung.