BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Klausur zum Kurs Verteilte Systeme

1.Prüfungszeitraum Sommersemester 2016

[redacted]

☐ Dies ist mein dritter Prüfungsversuch

Wir begrüßen Sie herzlich zur Klausur Verteilte Systeme und bitten Sie diese Hinweise vollständig und aufmerksam durchzulesen, bevor Sie mit der Bearbeitung der Aufgaben beginnen.

- Die Klausurdauer beträgt 60 Minuten.
- Es sind keine Hilfsmittel zugelassen.
- Bitte geben Sie die Klausur komplett und geheftet wieder ab.
- Bitte schreiben Sie mit schwarzem oder blauem Stift. Nicht mit Bleistift.
- Bevor Sie mit der Bearbeitung der Aufgaben beginnen, tragen Sie bitte auf dem Deckblatt Name und Matrikelnummer ein.
- Schreiben Sie Ihre Lösungen bitte auf die Aufgabenblätter bzw. die dafür vorgesehenen Leerseiten und benutzen Sie ggf. auch die Rückseiten.
- Auf jedes Blatt, auf dem sich Teile Ihrer Lösung befinden, schreiben Sie bitte oben Ihren Namen und Ihre Matrikelnummer.
- Wenn Sie eine Prozedur oder ein Programm schreiben sollen, achten Sie auf eine klare Gliederung und eine ausführliche Kommentierung.
- Bitte halten Sie ein Personaldokument und Ihren Studienausweis bereit.

Wir wünschen Ihnen bei der Bearbeitung der Klausur viel Erfolg.

Aufgabe	1	2	3	4	5	6	Summe
Punkte	9	13	10	7	9	12	60
Erreicht	5	9,5	10	7	8	11	50,5

Note: 1,7

[redacted]

Aufgabe 1: Grundlagen (9 Punkte)

Bitte beantworten Sie kurz in jeweils wenigen Sätzen folgende Fragen.

1. Definieren Sie den Begriff „Verteiltes System“. (2P.)
2. Was bedeutet Transparenz? Welche Arten gibt es, nennen und beschreiben Sie kurz 3 mögliche Arten? (5P.)
3. Warum ist der Begriff eines „globalen Zustands“ bei verteilten Systemen problematisch? (2P.)

1. Ein verteiltes System ist ein Zusammenschluss mehrerer autonomer Systeme (Rechner) 0,5P

2. Transparenz verbirgt verschiedene Eigenschaften der Verteilung in einem VS

2.1 Ortstransparenz: Es ist irrelevant, ob sich bspw. ein Server in einem lokalen Netzwerk befindet, über das Internet angebunden ist oder sogar auf dem selben Rechner befindet. nicht genau.

3,5P.

2.2: Zugriffstransparenz: Entfernte und lokale Dienste lassen sich genau gleich aufrufen (z.B. RMI)

2.3: Replikationstransparenz: Wenn es mehrere Replikate einer Ressource gibt, ist es egal, auf welches man zugreift.

3. Es gibt in den Teilsystemen keine exakt gleichen Uhren, sodass es sehr schwierig ist, den Zustand der Teilsysteme zu einem bestimmten Zeitpunkt t in einem globalen Zustand zusammenzufassen. 1P

Aufgabe 2: Systemmodelle/Middleware (13 Punkte)

1. Beschreiben Sie den Unterschied zwischen synchroner und asynchroner Kommunikation. Nennen Sie jeweils einen Vor-und Nachteil. (4P.)
2. Beschreiben Sie die Funktionsweise und das Einsatzgebiet eines Reverseproxy. (2P.)
3. Welche Rolle spielt die Middleware-Schicht bei verteilten Systemen? (3P.)
4. Benennen Sie insgesamt drei Kernkomponenten auf Clientseite und auf Serverseite, die zur Implementierung eines entfernten RMI-Objektaufrufs erforderlich sind. Beschreiben Sie in einem Satz deren Funktionalität. (4P.)

1. synchrone Kommunikation: Beim Senden einer Nachricht wird der sendende Prozess geblockt/angehalten, bis die Antwort da ist.
- Vorteil: einfach zu implementieren, kaum Koordination
- Nachteil: Beim Senden mehrerer Nachrichten müssen alle nacheinander verarbeitet werden, dauert lange nicht unbedingt

3P.

asynchrone Kommunikation: Nachrichten ~~werden~~ können parallel verschickt werden, der sendende Prozess kann während des Wartens weiterarbeiten.
- Vorteil: kein Blocking / unnötiges Warten
- Nachteil: aufwändig zu implementieren, mehrere Prozesse müssen koordiniert werden, Antworten müssen ggf. ebenfalls koordiniert werden

Aufgabe 2

2. ~~Eine~~ Ein Reverseproxy agiert stellvertretend für einen Server. Bei der Kommunikation wendet sich ein Client zuerst an den Reverseproxy, dieser leitet die Anfragen auf den eigentlichen Server weiter bzw. verteilt sie unter mehreren Servern

1 P.

Einsatzgebiete:

- Load Balancing: Reverseproxy verteilt Anfragen auf mehrere (replizierte) Server.
- Single Sign On: Die Authentifizierung eines Nutzers findet für mehrere Dienste zentral im Reverseproxy statt.

3. Middleware versteckt in einem VS die Heterogenität der Einzelsysteme. Sie bietet standardisierte Schnittstellen z. B. für das ~~Senden~~ von Nachrichten ~~Aufr~~ im VS

2 P.

Es gibt die Formen Message oriented Middleware (MoM) und anwendungsorientierte Middleware.

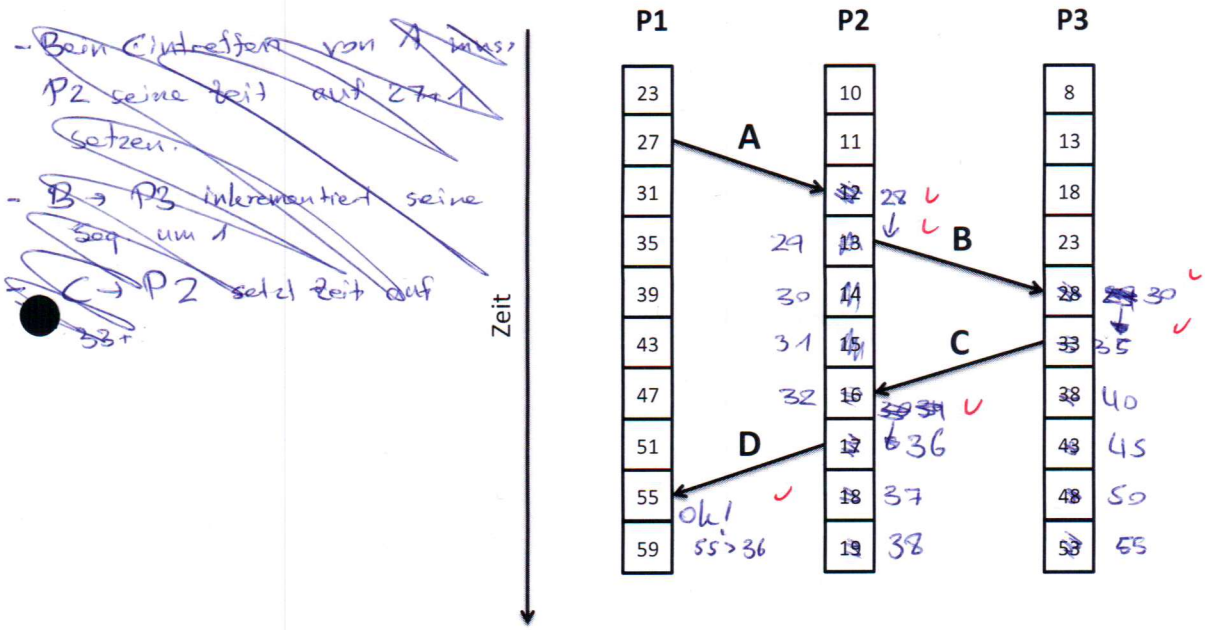
4. - Die RMI-Registry verwaltet die verschiedenen remote-Objekte und kann logische Namen auflösen, dient also als Nameserver in RMI
- ~~Pf~~ Der Client-Stub ist ein Stellvertreter für die Implementierung des Programms, ~~auf der Client Seite~~ der sich auf dem Client befindet und sich um die Kommunikation mit dem Server-Stub kümmert.

3 P.

- ~~RMI~~ Implementation beinhaltet Programmcode auf Server

Aufgabe 3: Synchronisation (10 Punkte)

Gegeben seien 3 Prozesse (P1-P3), die jeweils eine eigene logische Uhr gemäß des Algorithmus von Lamport besitzen. Die Taktung der Uhren ist unterschiedlich schnell. Prozess 1 läuft mit 4er Taktung, Prozess 2 in 1er Taktung und Prozess 3 in 5er Taktung. Die Prozesse tauschen Nachrichten (A-D) gemäß der u.a. Abbildung aus.



10P.

Erklären Sie den Algorithmus und zeigen Sie wie sich die Zeit der logischen Uhr pro Prozess verändert.

Der Lamport-Algorithmus basiert auf der happened-before-relation. Das bedeutet, dass die Prozesse zwar freie Hand bezüglich der Taktung haben, aber dafür verantwortlich sind, dass Ereignisse, die später passiert sind eine höhere Sequenznummer haben müssen, als früher geschehene Ereignisse.

Erhält ein Prozess eine Nachricht, vergleicht er die Sequenznummer aus der Nachricht mit seiner eigenen Zeit. Ist die eigene Zeit höher, inkrementiert der Prozess seine eigene Zeit und schickt diesen Wert in der nächsten Nachricht mit. Ist die eigene Zeit niedriger als die Sequenznummer, setzt er seine eigene Zeit auf den Wert der Sequenznummer + n (n>0) und schickt diese ebenfalls in der nächsten Nachricht mit.

So ist sichergestellt, dass in einem VS Nachrichten, die später kamen, immer eine höhere Sequenznummer haben

Aufgabe 4: Anwendungen/Sicherheit (7 Punkte)

1. HTTP/1.1 definiert acht verschiedene Anfrage-Operationen. Nennen Sie drei und erläutern Sie deren Zweck kurz. (3P.)
2. Erklären Sie die Begriffe Authentifizierung und Autorisierung. Wie kann dies in einer HTTP Web-Applikation umgesetzt werden. (4P.)

1. GET: [redacted] Mit GET wird eine Ressource gefragt und gelesen. (R in CRUD) ✓

3P. ✓

POST: Post fügt eine neue Ressource hinzu (C in CRUD) ✓

PUT: Put verändert/ersetzt eine Ressource (U in CRUD) ✓

2. Authentifizierung: Bei Authentifizierung wird ein Benutzer eindeutig identifiziert. ✓ Dies kann zB durch HTTP Basic Auth ✓ oder HTTP Digest Auth erreicht werden.

4p. Autorisierung: Bei der Autorisierung wird geprüft, ob der ~~A~~ authentifizierte Benutzer die Berechtigung hat, eine Operation auf einer Ressource auszuführen bzw. einen Dienst zu nutzen. ✓ Dafür kann ein Rollen/Gruppen-System genutzt werden ✓

Aufgabe 5: Netzwerke/Namensdienste (9 Punkte)

1. Wie unterscheiden sich Paketvermittlung und Leitungsvermittlung? Nennen Sie für jede Technik einen bedeutenden Vorteil. (4P.)
2. Wofür stehen die Abkürzungen URI, URL und URN. Geben Sie die ausgeschriebene Form und jeweils ein Bespiel an. (3P.)
3. Wozu werden allgemein im ISO/OSI-Modell bzw. dem TCP/IP-Modell Header benötigt? (1P.)
4. Welches Protokoll wird im Internet eingesetzt, um eine Auflösung von Layer-3 auf Layer-2-Adressen vorzunehmen? (1P.)

1. Bei der Paketvermittlung werden Daten in Form von Paketen durch ein Netzwerk geschickt.
Vorteil: auf einer Leitung können mehrere Prozesse Pakete verschicken und empfangen. ✓

Bei der Leitungsvermittlung besteht ~~eine~~ eine exklusive Leitung zwischen zwei Kommunikationspartnern. ✓
Vorteil: auf dieser Leitung können nur Daten dieser speziellen Kommunikation gesendet werden. Das bedeutet weniger Koordination und größeren Datendurchsatz. ✓

4P. ✓

Aufgabe 5

2 P. URI (Uniform Resource Identifier) ist eine abstrakte Art, Ressourcen zu identifizieren

z.B. "tel: +4930 95317"

URN (Uniform Resource ~~Identifier~~ Name) ~~ist ein~~ beschreibt einen logischen Namen innerhalb eines Namespaces

3 P. z.B. "urn:isbn:371-987-0-0-1" im Namespace von ISBN-Nummern

URL (Uniform Resource Locator) ist eine Möglichkeit, Ressourcen im Internet zu lokalisieren. Es ist eine Konkretisierung der URI

z.B. "http://www.beuth-hochschule.de"

3 P. Es werden Header benötigt, um beispielsweise Quell- und Zieladressen (oder Ports) zu ermitteln, aber auch, um das Protokoll auf der höchst höheren Schicht zu ermitteln, um das Datenpaket weiterzugeben. Im IP-Header kann so stehen, dass der Body ein TCP Segment ist, inkl. TCP-Header 1 P.

In den Headers können noch viele andere Meta-Informationen gespeichert werden.

4 P. DNS (Domain Name Service) löst Domains (auf Layer 3) zu IP-Adressen (auf Layer 2, Network) auf. IP gehört zu Layer 3

ARP

Aufgabe 6: Netzwerkprogrammierung (12 Punkte)

Einer der ältesten Internet-Dienste ist der ***echo***-Dienst. Er ist mit TCP und UDP über den Port 7 erreichbar. Der Dienst empfängt die vom Client gesendeten Daten und sendet sie unverändert zurück.

1. Schreiben Sie dafür in Java einen einfachen echo-Server (UDP **oder** TCP), der den Dienst auf einem variablen Port anbietet (Default-Port ist 7) und wie untenstehend aufgerufen wird. Die Anfrage des Clients soll auf der Konsole ausgegeben werden.

2. Beschreiben Sie die Funktionsweise des Programms.

Programmaufruf: 11P.

$> java EchoServer 1234

```
public class EchoServer {
   public static void main [String[] args) {
      if (args.length > 1) {
         System.exit(1);
      }
      int port = 7;
      if (args.length > 0) {
         port = Integer.parseInt(args[0]);
      }
      ServerSocket socket;
      try {
         socket = new ServerSocket(port);
         PrintWriter consoleOut = new PrintWriter (System.out, true);
         while (true) {
            try {
               Socket clientSocket = socket.accept();
               BufferedWriter clientOut = new BufferedWriter(new OutputStreamWriter(clientSocket.get
                     OutputStream()));
               BufferedReader clientIn = new BufferedReader(new InputStreamReader(clientSocket
                  .getInputStream()));
               String input = clientIn.readLine();
               consoleOut.println(input);
               clientout.println(input);
               clientout.flush();
            } catch (IOException e) {
               // per client connection
               e.printStackTrace();
            }
         } // endwhile
```

Exception möglich

Endlosschleife

```
} catch (IOException e) {
  // for ServerSocket
  e.printStackTrace();
} finally {
  if (socket != null) {
    socket.close();
  }
} //endmethod
} //endclass
```

Beschreibung:

- Es wird ein Socket auf dem angegebenen Port oder 7 geöffnet
- socket.accept(); wartet auf eine Verbindung eines Clients und gibt dann ein Objekt, dass den Client-Socket repräsentiert zurück.
- I/O wird vorbereitet
- ~~Wenn der Client eine Zeile abs~~
  Wenn eine Zeile vom Client angekommen ist, wird diese über OutputStream an ihn zurückgesendet und auf Console ausgegeben.
- Fehlerbehandlung findet pro Client-Connection und für gesamten ServerSocket statt. Bei Client-Socket Fehler, wartet auf nächsten Client.