Modul Verteilte Software-Systeme
Prüfung

Name: _____ Note: ____

Nr.	Aufgabe	Max. Punkte	Erreichte Punkte
1	VSS-Grundlagen und -Konzepte	15	
2	Sockets	13	
3	Enterprise Integration Patterns und Messaging	13	
4	RMI-Programmierung	12	
5	Quality Attributes, VSS-Design und -Test	12	
6	Operational Modeling	15	
7	Naming	5	
8	Verteilte Hash Tabelle	13	
9	Zeit Synchronisation	15	
10	Bitcoin & BitTorrent	7	
		Total 120	

Hinweise allgemein:
- Zeit (für beide Prüfungsteile): 120 min. D.h. ein erreichter Punkt entspricht ca. 1 Minute Lösungsaufwand.
- Zuerst diese Hinweise, dann die Prüfungsteile ganz durchlesen.
- Mit Ausnahme von Prüfungen aus früheren Semestern sind alle gedruckten Unterlagen erlaubt (Open Book).
- Es sind keine elektronischen Hilfsmittel zugelassen; auch kein Handy oder Smart Watch.
- Alle Antworten und Lösungen in diese Aufgabenblätter schreiben.
- Keinen Bleistift und keine rote Farbe verwenden.
- Die Antworten und Lösungen müssen nachvollziehbar sein. Bitte alles in leserlicher Schrift!
- Keine Fragen mitten in der Prüfung. Bei Unklarheiten sind sinnvolle Annahmen zu treffen und diese klar zu dokumentieren.
- Die Bewertungspunkte der Aufgaben können nach der Korrektur noch etwas variieren.

Hinweise zum Prüfungsbeginn:
- Versehen Sie das Titelblatt mit Ihrem Namen
- HSR-Badge oder pers. Ausweis (ID/Fahrausweis mit Foto) sichtbar auf den Tisch legen.
- Allfällig bereitliegende Prüfung nicht umdrehen oder umblättern, bis die Aufsichtsperson explizit das Ok dazu gegeben hat.

Hinweise zur Prüfungsabgabe:
- Kontrollieren Sie nochmals, ob die Prüfung mit Ihrem Namen versehen ist. Geben Sie alle Blätter – auch von nicht gelösten Aufgaben – geordnet und geheftet ab.
- Bis 20 min. vor Prüfungsende: Geben Sie ein Zeichen und lassen Sie sich die Prüfung von der Aufsichtsperson abholen. Verlassen Sie dann leise den Raum.
- Ab 20 min. vor Prüfungsende: Drehen Sie Ihre Prüfung um und warten Sie, bis die Prüfung eingesammelt wird. Bleiben Sie dann sitzen bis die/eine Aufsichtsperson ein Zeichen zum Verlassen des Raums gegeben hat.

Aufgabe 1: VSS-Grundlagen und -Konzepte (15 Punkte)

Stimmen die folgenden Aussagen? Tragen Sie wahr oder falsch ein und begründen Sie Ihre Antwort mit 2-3 Stichworten. Hinweise zum Ausfüllen:

- Falls die Antwort einer Vorlesungsfolie direkt entnommen werden kann, reicht der Verweis auf die Folie (in der Syntax „Lektion x, Folie y"). Man kann aber auch anders begründen, der Folienverweis ist nicht explizit gefordert.
- Falls eine verallgemeinernde Aussage getroffen wird, reicht ein Gegenbeispiel, um diese zu falsifizieren. Bei wahren Aussagen kann ein Beispiel ebenfalls als Begründung dienen.
- Falls nur ein Teil der Aussage stimmt, ist die Gesamtaussage als falsch zu kennzeichnen; als Begründung kann der falsche Teil der Aussage gekennzeichnet (und begründet) werden.

Aussage	(W/F)	Begründung (Ausführung, Erklärung)
In jedem verteilten Software-System findet Remote-Kommunikation statt; diese kann entweder synchron oder asynchron erfolgen.		
Das World-Wide Web realisiert den Architekturstil Client-Server.		
Architekturstile wie Client/Server, Peer-to-Peer und Hub-and-Spoke unterscheiden sich hinsichtlich ihrer Topologien, also der Remote-Kommunikationsbeziehungen.		
Drei Abstraktionsebenen bei der Remote-Kommunikation (also Kommunikation über Prozessgrenzen hinweg) sind 1. Socket-RPC, 2. Document Messaging, 3. Web Services (SOAP oder REST).		
Messaging APIs müssen die Aufrufprimitive Put (Nachricht in Queue stellen) und Get (Nachricht aus Queue auslesen) unterstützen und bieten meist auch eine nichtkonsumierende Leseoperation an.		
RabbitMQ ist ein Messaging Provider, der statt des JMS APIs ein eigenes proprietäres Java API anbietet (aber auch andere Programmiersprachen unterstützt).		

Mit SOAP-basierten Web Services und WSDL lässt sich das Designprinzip „Design by Contract" umsetzen; semantische Schnittstellenänderungen machen dann oft syntaktische Änderungen nötig, wenn der Service Contract fachdomänenspezifisch modelliert ist.		
Wenn Message Producer, Message Consumer und Message Broker auf verschiedenen Hostrechnern laufen, sind für einen Message Channel zwei oder mehr TCP/IP-Socketverbindungen erforderlich.		
Zu den Message Reception Styles gehören neben ereignisgesteuerten Request Handlern (Callbacks) und Blocking Receive Calls auch Request-Reply Queues.		
Bei der RPC-Kommunikation kümmert sich die RPC-Middleware (also z.B. RMI oder CORBA) um die Serialisierung und Deserialisierung der In- und Out-Parameter der Remote Procedure Calls.		
Bei der Parameterübergabe wird zwischen Call-by-Value (Copy Semantics) und Call-by-Stub unterschieden (Objektreferenzen werden dabei dann als Stubs verwendet).		
In RMI müssen alle Klassen, die direkt oder indirekt in der Parametersignatur einer Remote-Methode (inklusive Rückgabewert) verwendet werden, das Serializable-Interface implementieren.		
Systemmanagement gliedert sich nach ITIL in die fünf FCAPS-Disziplinen, was für Fault, Configuration, Accounting, Performance, Security steht. Logging ist keine eigene FCAPs-Disziplin, sondern wird unterstützend eingesetzt.		
Log-Einträge sollten möglichst knapp gestaltet werden, um Speicherplatz und Übertragungskapazitäten zu sparen. Aussagekraft und Lesbarkeit spielen dagegen nur eine untergeordnete Rolle.		
Wichtige Konzepte in der JMX-Architektur sind Managed Resources, JMX Agent, Protocol Adapter und Extended Journal Beans (EJBs).		

Aufgabe 2: Sockets (13 Punkte)

a) Ordnen Sie die folgenden Beschreibungen einer oder mehreren Socket API-Primitiven (und/oder einer Socket-API-Methode in java.net) zu:

Beschreibung	**API-Primitive und/oder Java-Methode**
Erstellen und Übermitteln einer Nachricht an den Kommunikationspartner (in Form eines Bytestroms)	
Blockierendes Warten und anschliessendes Annehmen eines eingehenden Connection Requests (Antrag auf Verbindungsaufbau) des Clients auf der Serverseite	
Blockierendes Warten auf eine Nachricht, die vom Kommunikationspartner als Bytestrom gesendet wurde, und anschliessendes Lesen dieses Bytestroms	
Beantragen eines Verbindungsaufbaus mit dem serverseitigen Kommunikationspartner (erster API-Call auf der Client-Seite)	
Kontaktaufnahme mit dem lokalen Netzwerkinterface (also der verwendeten TCP/IP-Library) als erster Call auf der Serverseite und Bekanntgabe der Socket-Adresse (IP-Adresse und Portnummer)	

b) Beantworten Sie die folgenden Fragen zum Java Socket API:

1. Welchen Architekturstil realisieren TCP/IP-Sockets inhärent? Sind weitere Stile möglich?

Antwort:

2. Welche(s) Message Exchange Pattern(s) gibt das Java Socket API vor?

Antwort:

3. Wer legt fest, wie die zu übertragenden Daten kodiert werden, a) Programmierer(in), b) TCP/IP Implementierung im Betriebssystem, c) die Java Spezifikation (JSR) des Socket APIs?

Antwort:

4. Welche wichtigen Timeouts gibt es in der Socketprogrammierung?

Antwort:

5. Nennen Sie mindestens einen blockierenden und einen nicht-blockierenden Socket API Call.

Antwort:

6. Nennen Sie zwei häufige, socketspezifische Konfigurationsfehler bei der Verwendung von Kommunikation über TCP/IP-Sockets.

Antwort:

7. Was ist der Vorteil eines Multi-Threaded Socket Servers (im Vgl. zu Single-Threaded Server)?

Antwort:

8. Wie heissen die Übertragungseinheiten im verbindungslosen Protokoll (oberhalb von IP im Protokollstack), das eine Alternative zu TCP darstellt?

Antwort:

Aufgabe 3: Enterprise Integration Patterns und Messaging (13 Punkte)

a) Ordnen Sie die folgenden fachlichen Beschreibungen den zugehörigen Enterprise Integration Pattern (EIP) zu.

Beschreibung	Enterprise Integration Pattern
Dieser Nachrichtentyp modelliert Arbeitsaufträge für das Zielsystem wie z.B. „Lege Kunde an" in Form von Prozeduraufrufen.	
Message Consumer können sich zur Laufzeit dynamisch für Kanäle von diesem Typ registrieren und erhalten dann jeweils eine Kopie der Nachricht des Message Producers.	
Repräsentiert ein Anwendungsprogramm, das dem Messaging System Nachrichten sendet oder Nachrichten von diesem empfängt (also aus Message Channels ausliest).	
Sorgt dafür, dass Messages nicht aufgrund von temporären Ausfällen von Hardware oder Messaging Middleware verlorengehen (brokerinterne persistente Speicherung).	
Dieser Consumertyp ist in der Lage zu spezifizieren, an welchen Nachrichteninhalten er interessiert ist; er erhält nur Nachrichten, die diesen Filter passieren (erfüllen).	

c) Beantworten Sie die folgenden Fragen zu Messaging und EIPs.

1. Welche JMS-Konzepte entsprechen dem EIP Patternpaar Request-Reply und Return Address?

Antwort:

2. Welches EIP wird vom RabbitMQ Konzept der Fanouts realisiert?

Antwort:

3. Welches Quality Attribute wird vom AMQP-Standard adressiert?

Antwort:

4. Welche der folgenden Message Exchange Pattern lassen sich mit Messaging realisieren: One Way, Duplex, Request-Reply?

Antwort:

5. Welches EIP und welches JMS-Konzept ermöglichen es, einen Timeout auf Nachrichtenebene zu spezifizieren?

Antwort:

c) Nennen Sie drei technische Herausforderungen und/oder zu treffende Designentscheidungen bei der Verwendung von queue-basiertem Messaging:

Aufgabe 4: RMI-Programmierung (12 Punkte)

Ausgangslage: Gegeben seien die folgenden vier Java-Programmcodes für die RMI-Kommunikation (Server und Client mit Interface und Data Transfer Object).

a) Data Transfer Object:

```
package vat;

import java.io.Serializable;
import java.math.BigDecimal;

public class Product implements Serializable {
    private static final long serialVersionUID = 1L;
    private final String productname;
    private final BigDecimal productprice;

    public Product(String name, BigDecimal price) {
        this.productname = name;
        this.productprice = price;
    }

    public String getName() {

        return this.productname;
    }

    public BigDecimal getPrice() {
        return this.productprice;
    }

    public String toString() {
        return "[Product] " + this.productname + ": " + this.productprice;
    }
}
```

b) Interface:

```
package vat;

import java.math.BigDecimal;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Collection;

public interface ValueAddedTax extends Remote {
    public static final int ID = 1295;
    public static final String HOST = "localhost";
    public static final String LOOKUP_NAME = "VAT";

    BigDecimal calcVAT(Collection<Product> articles) throws RemoteException;
}
```

c) Server:

```
package vat;

import java.math.BigDecimal;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
```

```
import java.util.Collection;

public class ValueAddedTaxServer implements ValueAddedTax {

    @Override
    public BigDecimal calcVAT(Collection<Product> products) throws RemoteException {
        BigDecimal result = new BigDecimal(0);
        BigDecimal vatRate = BigDecimal.valueOf(0.19);
        for (Product a: products) {
            result = result.add(a.getPrice().multiply(vatRate));
        }

        // delay return so that multiple clients can connect
        try {
        Thread.sleep(2000);
     } catch (InterruptedException e) {
        e.printStackTrace();
     }

        return result;
    }

    public static void main(String[] args) throws Exception {
        System.out.println("[VATServer] starting");

        ValueAddedTaxServer server = new ValueAddedTaxServer();
        System.out.println("[VATServer] calling export()");
        ValueAddedTax stub = (ValueAddedTax) UnicastRemoteObject.exportObject(server,
          ValueAddedTax.ID);

        System.out.println("[VATServer] creating local registry");
        Registry registry = LocateRegistry.createRegistry(ValueAddedTax.ID);
        System.out.println("[VATServer] get registry");

        registry.rebind(ValueAddedTax.LOOKUP_NAME, stub);
        System.out.println("[VATServer] " + ValueAddedTax.LOOKUP_NAME + " bound");
    }
}
```

d) Client:

```
public class ValueAddedTaxClient {
    public static void main(String[] args) throws Exception {

        // generate some test data
        System.out.println("[VATClient] creating sample data");
        ArrayList<Product> list = new ArrayList<Product>();
        list.add(new Product("P1", BigDecimal.valueOf(9.95)));
        list.add(new Product("P2", BigDecimal.valueOf(90.00)));
        list.add(new Product("P3", BigDecimal.valueOf(142.00)));
        list.add(new Product("P4", BigDecimal.valueOf(1.11)));

        // get link to stub from registry
        Registry registry = LocateRegistry.getRegistry(ValueAddedTax.HOST,
          ValueAddedTax.ID);
        System.out.println("[VATClient] looking up registry");
        ValueAddedTax stub = (ValueAddedTax)
          registry.lookup(ValueAddedTax.LOOKUP_NAME);
```

```
        // remote method invocation
        System.out.println("[VATClient] Remote Methode Invocation");
        System.out.println("[VATClient] Total VAT for "
          + list.size() + " products is: " + stub.calcVAT(list));
    }
}
```

Aufgabe: Beantworten Sie die folgenden Fragen zum obigen Code:

1. a) Müssen beide Programme gleichzeitig laufen, damit die RMI-Kommunikation stattfinden kann? Falls ja, welches Programm muss zuerst gestartet werden?

   b) Was passiert, wenn man das falsche Programm zuerst startet (also die richtige Reihenfolge aus a) nicht einhält)?

   c) Was passiert, wenn man das Client-Programm zweimal auf demselben Rechner startet?

   d) Was passiert, wenn man den Server zweimal auf demselben Rechner startet?

2. a) Was bedeutet die 1295 (im Interface definiert) in Client- und Server-Programm?

   b) Welche Remoting Pattern bzw. RMI-Konzepte erkennen Sie im Code (mind. zwei)?

   c) Schlagen Sie eine Verbesserung des Codes im Hinblick auf die Qualitätsattribute Flexibilität (des Deployments, der Kommunikationsbeziehungen) und Manageability vor.

3. Welche TCP/IP Socketcalls laufen wann und wo ab (in der RMI-Implementierung)?

4. Was ist administrativ erforderlich (bzw. was muss Im Code geändert werden), wenn man Client und Server auf zwei Nodes verteilt und die Adressierung entsprechend anpasst?

5. Was passiert, wenn man das Java-Interface auf dem Client in ein anderes Package verschiebt und rekompiliert, auf dem Server aber die alte Version verwendet?

Aufgabe 5: Quality Attributes, VSS-Design und -Test (12 Punkte)

Markieren Sie die richtige(n) Antwort(en) zu jeder Frage. Mindestens eine Antwort ist korrekt; bei einigen Fragen ist allerdings explizit angegeben, dass genau eine Antwort erwartet wird.

(Bewertung: ein Punkt pro richtiger und vollständiger Antwort)

1. Einen Performancetest sollte man ausführen…
   - deutlich unter der Kapazitätsgrenze des Systems.
   - an der Kapazitätsgrenze des Systems.
   - deutlich über der Kapazitätsgrenze des Systems.
   - es gibt keinen Zusammenhang zwischen der Kapazitätsgrenze und der Testdurchführung.

2. Gatling wird bei Performancetests von Web-Anwendungen eingesetzt für:
   - Black Box Testing (Aussensicht auf das getestete System)
   - White Box Testing (Innensicht der Anwendung)
   - Grey Box Testing (Innen- und Aussensicht)
   - ohne weitere Bibliotheken wie Metrics können mit Gatling keine Performancetests erfolgen.

3. Wenn man mehr Hardware vom selben Typ (Bsp. UNIX-Server) nutzt, ist die Scalability Tactic:
   - Scale Up
   - Scale In
   - Scale On Demand
   - Scale Out

4. Eine nur einmal deployte, systemkritische Infrastruktur- oder Anwendungskomponente ist ein:
   - Singleton
   - Non-distributed Deployment
   - Single Point of Failure
   - Bottleneck

5. Die Middleware Redis kann eingesetzt werden als:
   - Distributed Cache
   - Key-Value Store
   - Datenbank für Business Entities, auf die über den Schlüssel zugegriffen wird
   - Keines der Benutzungsszenarien a bis d.

6. Das Deployment Pattern Load-Balanced Cluster soll folgende Quality Attributes verbessern:
   - Performance
   - Robustness
   - Scalability
   - Fault Tolerance

7. Skalierbarkeit…
   - bezieht sich immer auf mindestens ein anderes Quality Attribute, z.B. Performance.
   - erreicht man durch Tactics wie Use Asynchronous Processing und Partition and Parallelize.
   - ist unabhängig vom Programmcode (reines Infrastrukturthema).
   - braucht man in jeder Anwendung, insbesondere in Web-Anwendungen.

8. Wenn man einen Database Node dreimal deployt, diese Nodes laufen und auch aktiv sind…
   ist der Standby Mode Cold Standby.
   ist der Standby Mode Warm Standby.
   ist der Standby Mode Hot Standby.
   ist der Standby Mode Hot Pool.

9. Den Übergang zwischen Anwendungs- und Infrastrukturdesign bilden in der UMF-Methode:
   Deployment Units
   WAR-Files
   Logical Components (aus dem UML Deployment Diagram)
   Maven pom.xml Files

10. Ein Circuit Breaker…
    prüft Anwendungsprogramme auf zyklische Abhängigkeiten zwischen Komponenten.
    konfiguriert Load Balancer nach der Strategie Round Robin.
    markiert kaskadierte Requests bei ausbleibenden Antworten als fehlerhaft (statt Retry).
    wird in Peer-to-Peer Netzwerken verwendet, um kreisförmige Kommunikation zu eliminieren.

11. Damit NFRs/QAs im Performancetest einsetzbar sind, müssen Sie…
    quantifiziert werden.
    maschinenlesbar sein (z.B. XML, JSON)
    realistisch gewählt werden.
    mit dem Auftraggeber des Projektes schriftlich fixiert und abgenommen werden.

12. Welche der folgenden Aussagen stimmen?
    Timeouts sind nur bei asynchroner Kommunikation sinnvoll, bei RPCs dagegen unwichtig.
    Timeouts werden von der Middleware und den Administratoren vergeben, nicht in APIs gesetzt.
    Timeouts muss man nur bei anspruchsvollen NFRs und Cluster-Patterns explizit setzen.
    Keine der Alternativen a bis c.

Aufgabe 6: Operational Modeling (15 Punkte)

Ausgangslage: Eine Web-Anwendung zum Verkauf von Elektronikkomponenten (Computer, Audio, Video, Photo) soll produktiv gehostet werden. Der Shop unterstützt die folgenden Use Cases:

1. Shop mit Elektronikkomponenten befüllen (Aktor: Sales Manager beim Shop-Betreiber)
2. Elektronikkomponenten browsen und zum Kauf vormerken (Aktor: Kunde)
3. Warenkorb anzeigen und zur Kasse gehen (Aktor: Kunde)
4. Auslieferdatum avisieren (Aktoren: Shop-Betreiber, Hersteller der Produkte; es gibt ca. fünf verschiedene Hersteller in vier Produktkategorien)
5. Bestellungen anzeigen (Aktoren: Kunde, Call Center Mitarbeiter beim Online-Shop).

Da Hersteller und Käufer über mehrere benachbarte Länder verteilt sind, treten Lastspitzen auf. Zahlungen werden über einen externen Payment Server abgewickelt. Kaufinteressenten verbringen zum Teil bis zu einer Stunde eingeloggt und suchend auf der Webseite (also in der Anwendung) und sammeln potentiell interessante Angebote in einem Warenkorb.

Wichtige Konzepte in der Anwendung sind der Elektronikkomponenten-Katalog (Produktanzeige) und Kunden mit ihren Warenkörben. Die Anwendungsdaten werden in einer lokalen MySQL-Datenbank persistiert. Diese Datenbank ist Teil des Anwendungsdeployments. Benutzer müssen sich bei der Anwendung anmelden; die Korrektheit der Login Credentials wird mit Hilfe eines Directory Servers überprüft, der über eine LDAP-artige Active Directory Schnittstelle angebunden ist. Für den Fall der Nichterreichbarkeit des Active Directory Servers steht unter einer anderen Netzwerkadresse ein zweiter hochgefahrener, aber defaultmässig nicht aktiver Active Directory Server bereit.

Als nichtfunktionale Anforderung wurde spezifiziert, dass 90% aller User Requests aus den Use Cases 1 bis 5 in weniger als 1s beantwortet werden sollen; die Verfügbarkeit der Anwendung soll über das Jahr gesehen 99.95% betragen. Es ist zunächst von 100.000 gleichzeitig eingeloggten Kunden auszugehen; dazu kommen ca. 20 Shop-Mitarbeiter.

Das Anwendungs- und Infrastrukturmanagement wird von einem externen Outsourcing Provider, bei dem die Anwendung auch gehostet wird, durchgeführt. Die Operator (Administratoren) des Outsourcing Providers befinden sich nicht in der gleichen Lokation wie das Data Center, in dem die IT-Infrastruktur und die Server-Hardware für den Online-Shop laufen.

Aufgaben:

a) Erstellen Sie ein operationales Modell für den Serverteil des Online Shops. Dieses operationale Modell soll die nichtfunktionalen Anforderungen aus der Ausgangslage erfüllen können. Die Anwendungskomponenten, die deployed werden, sollen ebenfalls dargestellt werden (mit Angabe des Deployment-Ortes).

Wählen Sie eine geeignete, standardisierte Notation und achten Sie auf syntaktisch korrekte Verwendung dieser Notation.

b) Begründen Sie eine Ihrer Architektur- und Modellierungsentscheidungen, z.B. warum Sie welche Deployment Patterns und Tiers gewählt haben (3-4 Sätze).

c) Wie gehen Sie vor, um Performance-Bottlenecks im Anwendungsdeployment zu finden (2-3 Sätze)?

Aufgabe 7: Naming (5 Punkte)

Vervollständigen Sie folgende Tabelle:

	Entity	Access Point	Address	Name
Postwesen (Beispiel)	Person	Briefkasten	Postanschrift	Name der Person
DNS				
ARP				
mDNS (multi-cast DNS)				
Bitcoin				
Bittorrent				

(.25 x 20 Felder)

Aufgabe 8: Verteilte Hashtabelle (13 Punkte)

Gegeben: Ein Chord System mit maximal 16 Knoten (IDs 0 bis 15).

a) (5 Punkte) In das leere Chord System wird der erste **Knoten** mit ID=4 eingefügt. Nun werden der Reihe nach **Objekte** mit den **Schlüsseln** 2, 5, 10, 14 eingefügt. Wie sieht die Finger-Tabelle für den einzigen Knoten mit ID=4 aus?

Knoten-ID (p) = 4

i	2^{i-1}	$p + 2^{i-1}$	$succ(p+2^{i-1})$
1	1		
2	2		
3	4		
4	8		

Schlüssel

Vorgänger

(.5 Punkte pro Feld x 10)

b) (8 Punkte) Wie ändern sich die Finger Tabellen, falls ein weiterer Knoten mit ID=12 eingefügt wird?

Knoten-ID (p) = 4

i	2^{i-1}	$p + 2^{i-1}$	$succ(p+2^{i-1})$
1	1		
2	2		
3	4		
4	8		

Schlüssel

Vorgänger

Knoten-ID (p) = 12

i	2^{i-1}	$p + 2^{i-1}$	$succ(p+2^{i-1})$
1	1		
2	2		
3	4		
4	8		

Schlüssel

Vorgänger

(.5 Punkte pro Feld x 16, Feld „$p + 2^{i-1}$“ bei p=4 wird nicht nochmal benotet)

Aufgabe 9: Zeit Synchronisation (15 Punkte)

a) (10 Punkte) Sind die folgenden Aussagen (immer) Wahr oder Falsch? Geben Sie eine Begründung für Ihre Antwort.

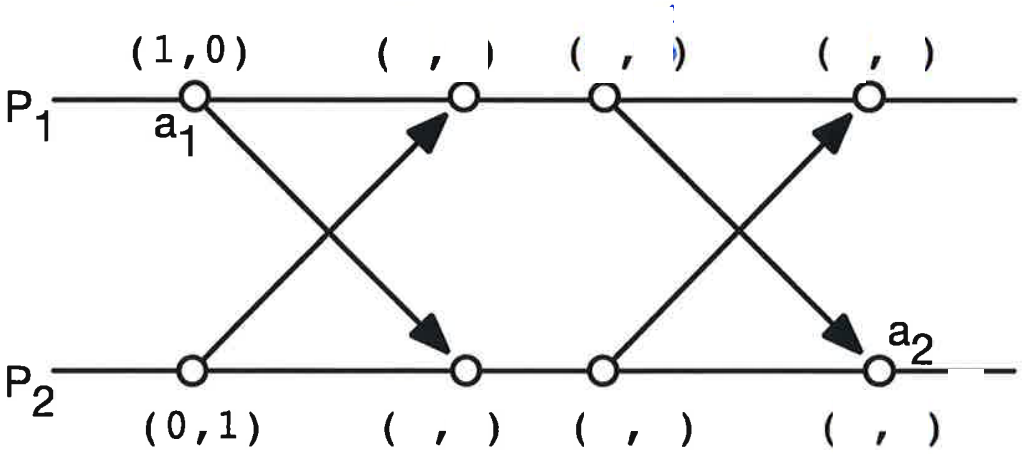
Notation:

- a und b sind beliebige Ereignisse
- "→" ist der "happened before"-Relation auf Ereignisse
- T(x) ist der Wert der (globalen) Echtzeit-Uhr für Ereignis x.
- C(x) ist der Wert der Lamport-Uhr für Ereignis x.
- V(x) ist der Wert der Vektor-Uhr für Ereignis x.

Aussage	Wahr (W) / Falsch (F)	Begründung
Zwei Ereignisse die nebenläufig (concurrent) sind passieren gleichzeitig.		
Gegeben: Ereignis a hat C(a) = 10 und Ereignis b hat C(b) = 20 Folgt: Ereignis a passiert vor Ereignis b		
Gegeben: V(a) = V(b) Folgt: T(a) = T(b)		
Gegeben: T(a) > C(b) Folgt: C(a) > T(b)		
Gegeben: V(a) = (0,0,1) und V(b) = (2,5,0) Folgt: a → b		

(1 Punkt x 10 Felder)

b) (3 Punkte) Bestimmen Sie die Vektorzeit für folgendes Prozessdiagramm.
Annahme: Als Ereignisse gelten nur das senden und empfangen von Nachrichten.



(.5 Punkte x 6 Vektorzeiten)

c) (2 Punkte) In der Abbildung oben sind zwei Ereignisse a_1 und a_2 markiert. Leiten Sie die zeitliche Abhängigkeit zwischen a_1 und a_2 anhand der Verktorzeiten der beiden Ereignisse ab.

Hinweis: Ihre Ableitung soll mit den Werten der Vektorzeiten für a_1 und a_2 anfangen, und mit einer „happens before"-Aussage enden.

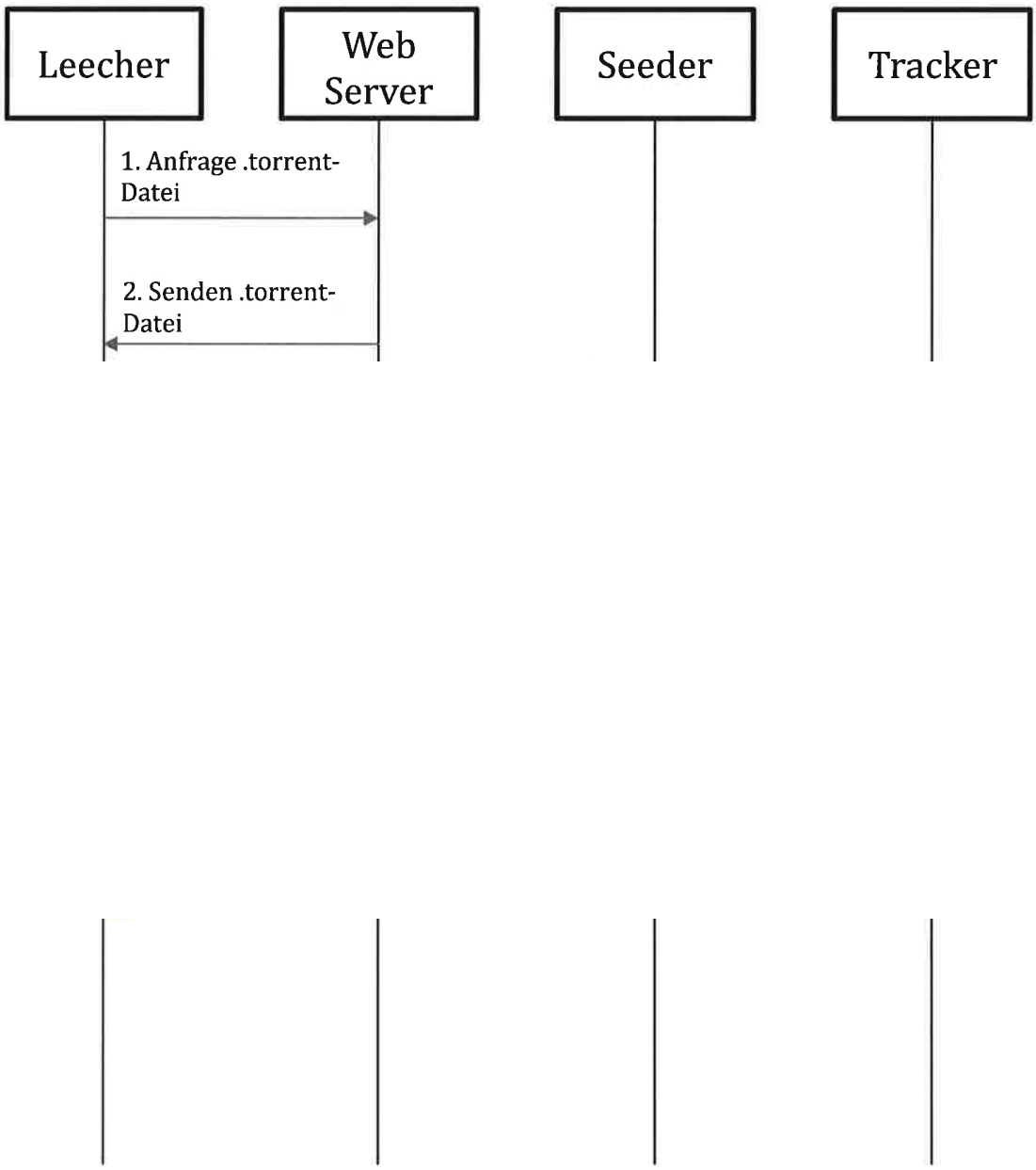
(1 Punkt Ableitung, 1 Punkt Resultat)

Aufgabe 10: Bitcoin & BitTorrent (7 Punkte)

a) (5 Punkte) Bitcoin: Wie wird Zeit synchronisiert im Bitcoin-System? Was sind die Ereignisse? Gibt es eine totale oder eine kausale Ordnung der Ereignisse? Begründen Sie Ihre Antwort. Wie wird die gewählte Ordnung erreicht?

Ereignisse: (1 Punkt)	
Ordnung (Total / Kausal): (1 Punkt)	
Begründung für Ihre Antwort bei „Ordnung“: (2 Punkte)	
Die gewählte Ordnung wird erreicht durch: (Stichwörtern genügen) (1 Punkt)	

b) (2 Punkte) BitTorrent: Vervollständigen Sie untenstehendes Sequenzdiagramm, welches die Komponenten eines BitTorrent-Schwarms abbildet. Es soll folgendes Szenario abgebildet werden: Leecher möchte die Datei 1337.bin über BitTorrent herunterladen. Seeder bietet diese Datei an. Die zum Schwarm gehörende .torrent-Datei liegt bereits auf dem Web Server. Der erste Schritt (Leecher lädt .torrent vom Webserver herunter ist bereits eingezeichnet). Vervollständigen Sie die nachfolgenden Schritte.



(.5 x 4 beschriebene Pfeile)