Fragebogen 1 – Einführung

1. Versuchen Sie möglichst präzise den Begriff des verteilten Systems zu definieren. Ist nach Ihrer Definition das Web ein verteiltes System?
    a. Zusammenschluss unabhängiger Rechner innerhalb Netzwerk - präsentieren sich dem Benutzer als ein Rechner
    b. Ja, das Web ist ein VS
2. Beschreiben Sie die grundsätzliche Architektur eines Client-ServerSystems. Wann ist eine Komponente ein Client und wann ein Server?
    a. Client und Server (Server oft verbunden mit DB, etc.)
    b. Client: wenn er Request sendet
    c. Server: erbringt Dienst (Response)
3. Was wird unter einer Komponente in der Software verstanden? Definieren Sie diesen Begriff in Hinblick auf Service und Benutzt-Relation.
    a. Klassenkomplex mit einer explizit definierten Schnittstelle, an der eine in sich geschlossene Leistung erbracht wird und die eine weitgehend freie Kombinierbarkeit mit anderen Komponenten ermöglicht
    b. Service: Erbringung einer in sich geschlossene Leistung, d.h. Realisierung einer konkreten Aufgabe
    c. Benutzt-Relation: Abhängigkeit von der Korrektheit, d.h. von der korrekten Erbringung einer Dienstleistung an einer definierten Schnittstelle
4. Was wird unter einer Benutzt-Relation verstanden? Erläutern Sie dies an einem kleinen Beispiel.
    a. Beziehungen zwischen Komponenten: Abhängigkeit von der Leistungserbringung sowie Korrektheit anderer Komponenten.
    b. Beispiel: A abhängig von B: Wenn B defekt, hat A die A-Karte
5. Warum ist die Betrachtung von Komponenten bei den verteilten Systemen wichtig?
    a. Verteiltes System besteht aus vernetzten Komponenten, die miteinander kommunizieren
6. Nennen Sie mindestens zwei wichtige Ziele, die mit Komponenten erreicht werden sollen.
    a. Unabhängigkeit
    b. Wiederverwendung
7. Auf welche prinzipiellen Arten können Komponenten interagieren? Nennen Sie zwei Möglichkeiten.
    a. Aufruf von Routinen (RPC, LPC)
    b. Austausch von Nachrichten (unicast)
    c. Signalisierung von Ereignissen (Events, Exceptions)
8. Erläutern Sie die Funktion der Middleware. Skizzieren Sie dabei, welche Aufgaben die Middleware in verteilten Systemen hat.
    a. Schnittstelle zwischen zwei oder mehreren Komponenten
    b. Schicht eines verteilten Systems
    c. Datentransfer, Umkodierung, (De)-Serialisierung
9. Nennen Sie drei Beispiele für Implementierungen von Middleware.
    a. RMI
    b. CORBA
    c. JBOSS
    d. .NET
10. Bei Corba gibt es eine IDL. Was ist das? Wofür steht das Kürzel IDL?
    a. Definition der Schnittstelle programmiersprachenunabhängig

    b. Interface Definition Language
11. Was wird (bei Corba) unter dem Provider- und was unter dem Required Interface verstanden? Wie sieht so etwas in Java aus?
    a. Provider-Interface: Schnittstelle zur Diensterbringung
    b. Required-Interface: Schnittstelle mit Anforderung an andere Komponenten
    c. Java: Komponenten als Packages. Zugriff und Benutzung nur über Interfaces.
12. Sollten Komponenten untereinander wie Klassen auch in Vererbungsbeziehungen stehen können? Begründen Sie die Antwort.
    a. Nein: Kapselung ist nicht mehr garantiert
    b. Erschwerte Wartbarkeit da gekoppelte Systeme
13. Bei der Beschreibung von Architekturen verteilter Systeme können Standpunkte eingenommen und aus deren Sicht Aspekte definiert werden. Erläutern Sie dieses Verfahren anhand eines kleinen Beispiels und drei Aspekten.
    a. Standpunkt = viewpoint = Ein Standpunkt ist eine konkrete Kombination von Aspekten, die durch die Interessen der Projektbeteiligten bestimmt ist.
    b. Komponenten und Konnektoren-Standpunkt: Aufbau des Komponentensystems
    c. Laufzeitstandpunkt: Komponenten und Laufzeitumgebungen bzw. Sprachen
    d. Prozess-Standpunkt: Threads und Prozesse sowie Synchronisation - Verteilungsstandpunkt: Wo welche Komponenten laufen
    e. Entwicklungsstandpunkt: Aspekte der Realisierung, z.B. Testen
    f. Datenstandpunkt: Persistenz von Daten bzw. Datenbanken
14. Was wird unter Persistenz einer Komponente verstanden?
    a. Fähigkeit eines Programms/Software/Komponente ihren internen Zustand länger als die Lebenszeit des korrespondierenden Objekts im RAM zu erhalten
15. Nennen Sie zwei wichtige Ziele von verteilten Systemen bzw. Anwendungen.
16. Worin besteht der wesentliche Unterschied zwischen einem ClientServer-System und einem Peer-to-Peer-System?

Fragebogen 2 – Komponenten

1. Komponenten werden meist innerhalb von Containern betrieben. Welche Aufgaben bzw. Funktionen haben diese Container?
    a. Laufzeitumgebung und Management für Komponenten
    b. Zusammenfassung aller Management-Funktionen, die allen Komponenten gemeinsam sind
2. Zwischen Container und Komponente liegt ein Management-Interface. Nennen Sie Beispiele von Operationen dieses Interfaces.
    a. Starten und Zusammensetzen einer Komponente
        i. Dynamisches Laden aller Klassen zur Laufzeit
        ii. Konfigurieren
    b. Stoppen/Neustart (Sichern/Wiederherstellen des inneren Zustandes)
        i. Serialisieren und Deserialisieren des internen Zustands
        ii. Wiederanlauf – Verwaltung der Threads, z.B. in einem Thread-Pool
    c. Ein Container übernimmt alles, was allen Komponenten gemeinsam ist
3. Was wird unter einer Factory-Klasse verstanden?
    a. Entwurfsmuster, Erzeugsmuster
    b. Es definiert eine Schnittstelle zur Erzeugung einer Familie von Objekten, wobei die konkreten Klassen der zu instanziierenden Objekte nicht näher festgelegt werden.
4. Wo sollten die Factory-Klassen zum Aufbau einer Komponente liegen? Im Container oder in der Komponente selbst?

    a. factory klassen innerhalb der komponente, kommt drauf an, kann beides möglich sein
5. Was sind (z.B. in Java) Annotations?
    a. Mittel zur Strukturierung von Programmquelltexten, bei der die Erzeugung von Programmtexten und mit der Programmierung verbundener Hilfsdateien teilweise automatisiert wird.
    b. Erlaubt Einbindung von Metadaten in den Quelltext
6. Die Konfiguration einer Komponente kann mit Annotations oder über eine externe Datei erfolgen. Beschreiben Sie die Vor- und die Nachteile beider Vorgehensweisen.
    a. konfigurations in datei -> können Datei unabhängig von source code ändern,
    b. Annotations: starr und schnell (innen)
    c. Extra Datei: flexibel und langsam(außen)
7. Ist Parallelität innerhalb einer Komponente sinnvoll? Nennen Sie dazu ein gutes Beispiel.
8. Ein Container für Komponenten kann in einen Upper und einen Lower Layer aufgeteilt werden. Nennen Sie jeweils Beispiele für Funktionen dieser Schichten.
    a. Upper: Laden, Starten, Stoppen, Entladen, Wiederanlauf
    b. Lower: Kommunikation, Betriebssystem
9. Welche zwei konkreten Arten von Managementbeziehungen können zwei Komponenten einander haben?
    a. eine komoponente ist zustäding eine andere hochzuladen,
    b. laden, entladen
10. Welche konkreten Arten von Kommunikation können zwei Komponenten haben?
11. Ein Container hat viele Aufgaben, erläutern Sie die folgenden prinzipiellen: Authentifizierung, Permission, Time, Mover
    a. Authentifizierung: Prüfung der Identität
    b. Permission: Vergabe von Rechten (Autorisierung)
    c. Time: Abstimmung der Uhren bzw. der Uhrzeit
    d. Mover: Wandern von Komponenten im Netz
12. Welche drei Funktionen hat der Booter innerhalb der Container?
    a. Hoch- und Herunterfahren der Anwendung,
    b. Migration
13. Worin liegt der wichtigste Unterschied zwischen dem Booter und dem Loader einer Komponente?
    a. loader: läd komponente in ram
    b. booter mach das gleiche aber auf mehrere Knoten
14. Benötigt der Booter eine Konfiguration in Form einer IDL/CDD-Definition oder in Form einer ADD? Was bedeuten die Kürzel?
    a. IDL: Interactive Data Language
    b. CDD: Component Deployment Definition
    c. ADD:
15. Wenn Sie eine verteilte Applikation implementierungsunabhängig beschreiben wollen, was benötigen Sie dazu? IDL, ADD oder CDD? Was bedeuten diese Kürzel?

Fragebogen 3 – Prozess und Threads

1. Was ist ein Prozess und was ein Thread? Und was wird in dieser Lehrveranstaltung (und auch im Kontext von Unix) unter einer Task verstanden?
    a. Prozess - Sequentiell ablaufendes Programm
    b. Thread - Eigenständiges sequentiell ablaufendes Programmstück als Teil eines Prozesses

    c. Task - Oberbegriff von Thread und Prozess
2. Aus welchen RAM-Bereichen besteht ein laufender Prozess? Nennen Sie drei und beschreiben Sie ihre Aufgaben.
    a. Stack - Bereich für lokale Variablen und Sprunginformationen
    b. Heap - Bereich für globale, statische Variablen und Objekte
    c. Code - Bereich der Instruktionen
3. Wenn zwei unterschiedliche Prozesse im RAM liegen: wieviele CodeSegmente liegen dann im RAM? Gilt das auch für zwei Threads eines Prozesses?
    a. Zwei
    b. nein, da alle Threads innerhalb eines Prozesses den selben Code benutzen
4. Betrachten Sie das Umschalten der CPU zwischen zwei Prozessen und dann zwischen zwei Threads. Worin liegt der wesentliche Unterschied?
    a. Threads - selber virtueller Speicher
    b. Prozesse - Wechsel des virtuellen Speichers
5. Beschreiben Sie abstrakt die einzelnen Schritte zum Umschalten der CPU zwischen zwei Threads innerhalb desselben Prozesses.
    a. 1) CPU-Status von 1. thread in RAM retten
    b. 2) CPU-Status von 2. thread in CPU laden
6. Was wird unter dem CPU-Status verstanden?
    a. Inhalt der CPU, also alle Register
7. Was ist bei Prozessen bzw. Threads ein Scheduler?
    a. Thread innerhalb OS Zuteilung CPUs zu UserThreads nach einem Verfahren.
8. Welche Aufgaben hat der (CPU-)Scheduler?
    a. Planer: - Verwaltung Tasktabelle
    b. Zuordnung Threads zu CPU
    c. Verwaltung Taskzustand
9. Worin besteht der wesentliche Unterschied zwischen Kernel- und User-Threads? Liegen diese Threads im Kernel- oder im UserAdressraum?
    a. Kernel-Threads - Verwaltet vom OS User-Threads - Verwaltet von Software (Bsp.: JVM) innerhalb ihres Prozesses.
    b. Im User-Adressraum
10. Welche Informationen stehen in der Thread-/Prozess-/Task-Tabelle? Nennen Sie beispielhaft drei Arten von Informationen und erläutern Sie, was diese bedeuten.
    a. Deskriptor:
        i. Identifikation (Name, Nummer)
        ii. Vebrauchte CPUZeit des Threads
        iii. Priorität des Threads
        iv. Permission des Threads
    b. Alles, was einen Thread ausmacht.
11. Prozesse haben aus der Sicht der Schedulers Zustände. Nennen Sie mindestens zwei und beschreiben sie, was sie bedeuten.
    a. Running: Task hat gerade CPU
    b. Ready: Task wartet auf CPU Waiting: Task wartet auf Beendigung von I/O
    c. Sleeping: Task benötigt später CPU (Beispiel: cron , wenn nicht benutzt , dann sleeping)
12. Welche Aufgaben hat der Timer bei dem Time-Sharing-Verfahren? Was ist bzw. was macht ein Timer?
    a. Zeitscheiben definieren
    b. Senden eines Interrupts in zyklischen Abständen

13. Was ist eine Zeitscheibe?
    a. Zeitraum wo ein Prozess die CPU bekommt. Dauer zwischen zwei TimerInterrupts
14. Beschreiben Sie das Time-Sharing-Verfahren.
    a. In regelmäßigen Abständen die CPU mit einzelnen Threads oder Tasks versorgt (mit Zeitscheiben) nach bestimmten Verfahren, HighPriority-First, etc

Fragebogen 4 – Synchronisation

1. Was ist ein kritischer Abschnitt? Was verbindet eine Gruppe verschiedener kritischer Abschnitte konzeptionell miteinander?
    a. Codebereich in dem Probleme mit anderen Threads auftreten können
    b. Alle kritischen Abschnitte, die sich auf dieselbe Menge von Betriebsmitteln beziehen, werden zu einer Klasse zusammen gefasst.
2. Wann bilden kritische Abschnitte eine Klasse? Müssen kritische Abschnitte derselben oder unterschiedlicher Klassen koordiniert werden?
    a. Klasse - Kritischen Abschnitte beziehen sich auf dieselbe Menge von Ressourcen
    b. kritische Abschnitte müssen Derselben Klasse koordiniert werden.
3. Was ist ein Deadlock? Nennen Sie dazu ein Beispiel. Was wird unter Starvation verstanden?
    a. Deadlock: Warten auf ein Ereignis, das nie eintreten kann
    b. Beispiel: Verkehrskreuzung mit vier Autos
    c. Starvation: Zu langes Warten auf ein mögliches Ereignis
4. Beschreiben Sie das Szenario der Spaghetti essenden oder denkenden Philosophen. Konstruieren Sie eine Deadlock-Situation und eine Situation, bei der Starvation entstehen kann.
    a. Spaghetti - Jeder von vier Philosophen isst entweder Spaghetti oder denkt. Es existieren nur vier Gabeln aber jeder P. braucht zwei Gabeln zum Essen. Da alle P. denselben Algorithmus benutzen, verhungern sie. Deadlock - Vier Autos an einer Kreuzung
    b. Starvation - Auf Aufzug warten (HTW Stock 8), teilweise wochenlang als Maschine wartend oder Treppe nutzen als Mensch
5. Beschreiben Sie eine durchaus reale Situation im Straßenverkehr, in der ein Deadlock entsteht.
    a. Eine Verkehrskreuzung mit vier Autos. Jeder hat Rechts vor Links Vorfahrtsrecht -> Deadlock
6. Was ist ein wechselseitiger Ausschluss und wozu dient dieser?
    a. Ergebnis der Abstimmung zur Problemverhinderung in kritischer Abschnitt
    b. Threads werden daran gehindert, eine Ressource, die in Benutzung ist, zu benutzen
7. Mit welchen Mitteln kann ein wechselseitiger Ausschluss realisiert werden? Nennen Sie dazu ein Verfahren.
    a. Benutzung von Semaphoren.
8. Welche Aufgaben haben die beiden abstrakten Operationen enter() und leave() (wie sie in der Veranstaltung eingeführt wurden)?
    a. enter - Betreten eines kAs
    b. leave - Verlassen eines kAs
9. Zur Realisierung des wechselseitigen Ausschlusses gibt es optimistische und pessimistische Verfahren. Beschreiben Sie beide Verfahren und nennen Sie jeweils ein Beispiel aus der Informatik, in dem diese angewandt werden.
    a. Pessimistisch - Vor kA sicherstellen, dass keine Probleme auftreten (Bsp.: Semaphoren)
    b. Optimistisch - Im kA wird auf Probleme geprüft (Bsp.: Transaktion mit Rollback)

    c. Optimistisch - Beispiel: Kreuzung mit Crash - Undo wird gemacht
10. Mit welchen Verfahren können die beiden Operationen lock() und unlock() realisiert werden? Nennen Sie dazu ein Beispiel. Diese werden manchmal als Spinlock bezeichnet. Was charakterisiert diese Verfahren?
    a. Mit Spezialinstruktionen
    b. swap-Operation
    c. Aus Nichtsperre eine Sperre machen
11. Welche Operationen werden im Kontext des wechselseitigen Ausschlusses mit speziellen Instruktionen realisiert? Die SemaphorOperationen p() und v()?
    a. 1. lock und unlock nur einer darf rein
    b. 2. NEIN
12. Was sind Semaphoren? Wie arbeiten binäre Semaphoren? Worin liegt dabei die Implementierungsidee? Nennen Sie die beiden Operationen, die typisch für Semaphoren sind.
    a. Semaphoren: Zähler, die passives Warten ermöglichen
    b. Es gibt nur zwei Zustände für die state Variable (0 und 1)
    c. Wartende Threads werden in schlafenden Zustand überführt
    d. p und v
13. Wenn Semaphoren mehr als zwei Werte annehmen können (IntegerSemaphoren): was bedeutet es für die kritischen Abschnitte, wenn der Initialwert einer Semaphore z.B. 4 ist?
    a. Es dürfen gleichzeitig 4 Tasks im kA vorhanden sein
14. Was machen die Operationen p() und v() speziell bei den IntegerSemaphoren?
    a. p() erniedrigt den internen Zähler
    b. v() erhöht den internen Zähler
15. Wie gelingt es den Semaphoren ohne Verschwendung der CPU-Zeit durch Busy Waiting das Warten von Prozessen bzw. Threads zu realisieren?
    a. Durch die Operationen p() und v() werden die Tasks in eine Warteschlange überführt und die CPU wird für die Wartezeit an andere Threads gegeben.
16. Was sind Monitore? Was waren die wichtigsten Gründe für deren Konzeption und Einführung in bestimmte Programmiersprachen?
    a. Abstrakter Datentyp mit notwendigen Operationen und der Bedingung, dass immer nur ein Task eine der Methoden zu einem Zeitpunkt ausführen kann
    b. Zur Vermeidung von Programmierfehlern (Deadlocks)
17. Jeder Monitor hat eine Entry-Queue. Welche Aufgabe hat diese Queue? Welche Prozesse/Threads kommen in diese Queue?
    a. Dient als Warteschleife für die Tasks
    b. Tasks, die Monitor betreten wollen, aber nicht dürfen
18. Was sollte ein Prozess machen, wenn er sich innerhalb eines Monitors befindet und feststellt, dass er auf den Aufruf einer Methode dieses Monitors durch einen anderen Prozess/Thread warten muss?
    a. In Condition-Queue gehen.
19. Welche Aufgaben haben die Condition-Queues und die UrgendQueue bei den Monitoren?
    a. Condition: Tasks warten auf bestimmte Ereignisse
    b. Urgent: Wird verwendet, damit nicht zwei Prozesse in einem Monitor sind, wenn zwei Prozesse voneinander abhängen. Prozess aus Condition wird in Urgent überführt. Urgent wird anschließend bevorzugt

Fragebogen 5 – Interprozess-kommunikation

1. Welche drei prinzipiellen Arten der Interprozess-Kommunikation werden in der Praxis benutzt? Schildern Sie kurz deren Funktionsweise.
    a. Speicherbasiert
    b. kleiner Bereich in den Prozessen ist VM (Virtueller per MMU realisierter Adressraum) bzw. Shared Memory
2. Zwei Threads in verschiedenen Prozessen auf einem System wollen miteinander kommunizieren. Welche der drei Arten von Interprozess-Kommunikation ist am schnellsten?
    a. Shared Memory durch VM (Virtueller per MMU realisierter Adressraum) je nach OS
3. Zwei Threads im selben Prozess wollen miteinander kommunizieren. Welche der drei Arten von Interprozess-Kommunikation wird immer angewendet?
    a. Shared Memory/gemeinsam benutzter Speicher
4. Mit Hilfe welcher Hardware-Komponente wird die Interprozess-Kommunikation mit der Benutzung gemeinsamen RAMs zwischen Prozessen realisiert (Virtueller Speicher wird vorausgesetzt)?
    a. Durch die Memory Management Unit (MMU).
5. Zwei Threads innerhalb eines Prozesses kommunizieren durch Austausch von Nachrichten. Müssen diese kopiert werden? Wenn die beiden Threads unterschiedlichen Prozessen angehören: müssen diese kopiert werden?
    a. Müssen nicht kopiert werden, werden aber häufig kopiert. Nur bei Shared Memory nicht, ansonsten immer
6. Für welche Aufgaben lassen sich zirkuläre Puffer gut verwenden?
    a. Zur Kommunikation zwischen Tasks.
7. Skizzieren Sie die Idee der zirkulären Puffer. Wie verläuft das Einbringen einer Nachricht (enqueue) und wie das Herausholen (dequeue)?
    a. enqueue - Inhalt des Index in wird beschrieben. Mithilfe Modulo wird geschaut, ob das Ende erreicht ist.
    b. dequeue - Inhalt des Index out wird geholt. Mithilfe Modulo wird geschaut, ob das Ende erreicht ist
8. Warum werden die zirkulären Puffer als zirkulär bezeichnet?
    a. Weil die Füllzeiger in und out immer im Array im Kreis herumlaufen. Hinterste Element ist verbunden mit dem ersten Element.
9. Was wird unter den Producer-Consumer-Problemen verstanden? Was ist dabei das Problem?
    a. Producer will schreiben wenn voll
    b. Consumer will lesen wenn leer
    c. Beide greifen auf einen gemeinsamen Buffer zu.
    d. Derjenige der produziert muss auch platz zum abspeichern haben . Consumer kann nicht mehr consumen als er Kapazität hat
10. Beschreiben Sie die Funktion und prinzipielle Implementierung von Pipelines (Pipes) in UNIX-Systemen. Welche Komponente implementiert diese Pipelines?
    a. Eine Queue vom Betriebssystem erstellt und verwaltet
    b. Nutzung von pipe-Syscall
    c. Das Betriebssystem
11. Worin besteht der wesentliche Unterschied zwischen Pipes und Named Pipes in UNIX? Für welche Art von Anwendungen lassen sich Named Pipes gut verwenden?
12. Wenn ein Prozess einen Port hat, was ist damit gemeint? Also, was ist ein Port und worin besteht der Unterschied zu einer Named Pipe?

    a. Port: Adresse mit symbolischen Namen. Feste Verbindung zwischen Server und Schnittstelle. Server müssen Semantik erbringen.
    b. Named Pipe: = Pipeline . Keine feste Verbindung zwischen Server und Schnittstelle. Beliebiger Server kann auslesen und irgendwie verarbeiten.
13. Wenn alle Prozesse (und auch Threads) Ports haben, gibt es dann noch einen wesentlichen Unterschied zwischen Clients und Servern? Wann ist eine Task dann ein Server und wann ein Client?
    a. Server und Client sind nur Rollen. Diese Rollen kann jeder einnehmen.
    b. Clients: nutzt sendOperationen
    c. Server: nutzt receiveOperation
14. Bei der Interprozess-Kommunikation wird zwischen synchron und asynchron unterschieden. Wann ist ein send() synchron und wann asynchron?
    a. asynchron: Sender und Empfänger zeitlich entkoppelt. Empfang nicht quittiert.
    b. synchron: Das send() wartet auf Bestätigung der Verarbeitung.
15. Was wird unter dem Rendezvous-Verfahren verstanden?
    a. Eine synchrone Kommunikation mit reply() Operation mit Bezugnahme auf die jeweilige Nachricht (eindeutige Nummer).
16. Worin besteht der wesentliche Unterschied zwischen einem synchronen send() und einem blockierenden send()?
    a. Blockieren - Fortschritt innerhalb Operation
    b. Synchron - Verhältnis zweier Partner
17. Beschreiben Sie die Realisierung eines asynchronen send(), wenn nur synchrone send()-Operationen zur Verfügung stehen.
    a. Zwischenspeicher (Queue)
    b. zweiter Sender, der stellvertretend kommuniziert.
    c. wie Just-in-time
18. Wann sind ganz allgemein Queues bei der Kommunikation erforderlich?
    a. häufige asynchrone Kommunikationunterschiedliche Geschwindigkeiten bei Beteiligten
19. Unter welchen Bedingungen kann eine Sende-Operation einer Nachricht blockieren? Nennen Sie zwei.
20. Kann eine wartende Empfang-Operation scheitern? Konstruieren Sie dazu eine pathologische Situation.
21. Beschreiben Sie die Idee des Publish-Subscribe-Musters bei Nachrichten-orientierten Systemen.
22. Was bedeutet es, wenn eine Task bei einem Broker ein Topic anmeldet?
23. Nennen Sie drei Aufgaben des Brokers bei Publish-Subscribe-Verfahren?

Fragebogen 6 – Remote Procedure Call Nr. 1 (RPC)

1. Definieren Sie den Remote Procedure Call als Gegensatz zum Local Procedure Call.
    a. LPC - Lokaler Prozeduraufruf innerhalb eines virtuellen Speichers (VM)
    b. RPC - Entfernter Prozeduraufruf (Routine liegt in einem anderen virtuellen Speicher / Host als der Aufrufer)
2. Können zwei Threads mit einem Remote Procedure Call miteinander kommunizieren? Unter welcher Bedingung?
    a. Ja, wenn nicht im selben Prozess.
3. Mit welchen besonderen Problemen muss sich die Implementierung des Remote Procedure Calls im Gegensatz zum lokalen Prozeduraufruf beschäftigen? Nennen Sie zwei.
    a. hohe Wartezeiten aufgrund von unterschiedlichen Geschwindigkeiten

    b. Deadlock durch Paketverlust
    c. Dubletten (mehrfacher Datensatz) beim Empfänger aufgrund von zu langem Paketversand
4. Warum kann beim Remote Procedure Call in jedem Falle vom Client und vom Server gesprochen werden? Was ist in dieser Hinsicht anders als beim einfachen Nachrichtenaustausch zwischen Gleichberechtigten?
    a. Aufgrund der Semantik per Definition.
    Client - Routine aufrufen
    Server - Routine realisieren
    b. Wenn Server B fehlerhaftes produziert, dann ist auch Client A fehlerhaft
5. Was macht die reply()-Operation anders als die send()-Operation auf der Seite des Servers?
    a. Eine synchrone Kommunikation mit reply() statt send() Operation mit Bezugnahme auf die jeweilige Nachricht (eindeutige Nummer).
6. Welche Möglichkeiten der Reaktionen bestehen, wenn der Verlust eines send()-Requests auftritt? Welche vier Arten von RPC-Semantik können prinzipiell daher realisiert werden?
    a. Senden kann wiederholt werden. - Maybe - At-least-once - At-most-once - Exactly-once
7. Die Möglichkeit von Dubletten von Datenpaketen kann dazu führen, dass beim RPC modifizierende Operationen doppelt ausgeführt werden. Wie lässt sich dies verhindern?
    a. Durch eine Prüfsumme
8. Aufgrund von Ausfällen von Systemen kann es beim RPC zu Waisen kommen. Was ist ein Waise? Und wie entsteht ein solcher?
    a. Eine Server-Aktivität, dessen Auftraggeber nicht existiert.
    b. Ein Waise kann beim Ausfall einer Komponente entstehen
9. Beschreiben Sie ein Verfahren zur Waisenbehandlung mittels Epochen. Was ist hierbei eine Epoche?
    a. Mit jedem Start des Client beginnt eine neue Epoche mit einer bestimmten ID, die den Paketen/Aufträgen mitgegeben wird, so dass veraltete Pakete identifiziert werden können.
    b. Epoche - Zeitabschnitt mit sequentiell nummerierten Teilabschnitten
10. Wenn ein Zusammenbruch einer verteilten Anwendung festgestellt wird, müssen eventuell schon erfolgreich durchgeführte Modifikationen zurückgenommen werden. Begründen Sie dies anhand eines Beispiels.
    a. Inkonsistenz kann entstehen. ABC-Beispiel: A & B haben Zustand aktualisiert, C aber nicht.
    b. Falls ein Aufruf in einem Kontext erfolgt, sollten die Modifikationen zurückgenommen werden. Bsp.: Transaktionen.
11. Mit den DB-Transaktionen können alle Effekte mittels rollback() zurückgenommen werden. Was ist erforderlich, wenn ein commit() schon durchgeführt wurde und die Effekte anschließend zurückgenommen werden sollen?
    a. Kompensationstransaktion, indem alle Operationen einzeln zurückgenommen werden.
12. Beschreiben Sie die beiden Phasen des 2PC-Verfahrens zur Realisierung von Transaktionen in verteilten Anwendungen.
13. Beim 2PC-Verfahren zur Realisierung von Transaktionen fällt der Koordinator aus und es gehen sämtliche Informationen über die Transaktionen dadurch verloren. Kann dann immer noch die ACID-bedingung eingehalten werden? Und falls ja, wie?
14. Was passiert beim Write-Ahead-Logging-Verfahren (WAL)?

Fragebogen 7 – RPC Nr. 2

1. Welche Aufgaben hat der Stub beim Remote Procedure Call? Nennen Sie mindestens zwei.
    a. Serialisieren und Deserialisieren.
2. Stubs werden zum Remote Procedure Call benötigt. Benötigt der Austausch von Nachrichten auch Stubs?
    a. Natürlich, selbstverständlich.
3. Aus welchen Gründen müssen die Parameter des Remote Procedure Calls auf der Senderseite serialisiert und auf der Empfängerseite deserialisiert werden?
    a. Senden von komplexen Datenstrukturen - Semantisch gleiche Repräsentation
4. Gilt der Zwang zur Serialisierung/Deserialisierung auch für die Resultate von Funktionen?
    a. Ja
5. Wenn mit einem Remote Procedure Call eine Funktion aufgerufen wird: was muss alles beim Rücktransport der Informationen zum Aufrufer übertragen werden? (Es sind zwei Arten von Informationen)
    a. Liste der Resultate (Informationen) - ExceptionInformationen (MetaInformationen)
6. Wenn verkettete Strukturen, z.B. Objekte vom Typ List<> in Java, serialisiert werden, muss ein bestimmtes nicht einfach zu lösendes Problem gelöst werden, welches?
    a. Datenstruktur exakt an der Stelle liegen, an der sie beim Client liegt
    b. Referenzvariablen zeigen vermutlich falsch.
7. Wie können Referenzen mit Pointern serialisiert werden? Beschreiben Sie drei Vorgehensweisen und bestimmen Sie deren Vor- und Nachteile.
    a. Verbieten (Nachteil: wird häufig gebraucht)
    b. Referenzobjekte mit serialisieren (Nachteil: Ineffizient)
    c. Referenzobjekte lokal verfügbar machen. Objekte zwischen Knoten verschiebbar. (Nachteil: Management von Speicherorten)
8. Wie sieht die grundsätzliche Software-Struktur beim RPC auf der Seite des Servers aus?
    a. Outline bestehend aus einer Endlosschleife. - Pakete empfangen - Namen auslesen - Prüfen, ob Name auf dem Server existiert - Parameterliste entpacken - Routine aufrufen - Ergebnis serialisieren - Nachricht zurückschicken
9. Eine Routine soll vom Aufruf zum nächsten mit unterschiedlich vielen Parametern aufgerufen werden können. Welches besondere Problem muss dazu gelöst werden?
    a.
10. Eine Routine soll vom Aufruf zum nächsten mit unterschiedlich vielen Parametern an derselben Stelle im Code aufgerufen werden. Welches Problem muss die aufrufende Routine dazu lösen? Warum besteht dieses Problem nicht, wenn die Aufrufe mit unterschiedlich vielen Parametern an verschiedenen Stellen im Code gemacht werden?
    a. Die Routine muss unterschiedlich viele Parameter handlen können. Weil überladen werden kann
11. Was wird unter Reflexion bei Programmiersprachen verstanden? Über was wird reflektiert und was ist dessen Ergebnis?
    a. Eigene Programmstruktur selbst analysieren. Über die Symboltabelle des Compilers.
12. Wie wird bei Java und anderen per Compiler übersetzten Sprachen Reflexion prinzipiell zur Laufzeit realisiert?
    a. Zur Laufzeit liegen Symboltabellen vor
13. Wie wird bei Skript-Sprachen wie z.B. JavaScript oder PHP Reflexion prinzipiell intern im Interpreter realisiert?
    a. Bei SkriptSprachen ist sämtlicher Quelltext vorhanden.
14. Was bedeutet technisch, wenn von einem nicht-wartenden send() oder receive() gesprochen wird? Welche Konsequenzen hat das „Nicht-Warten“?

    a. Es können zwei Befehle bei einer Komponente gleichzeitig ankommen.
15. Welche Reaktionen sind sinnvoll, wenn beim nicht-wartenden send() festgestellt wird, dass die erforderliche Sende-Queue im eigenen System voll ist und daher das send() nicht ausgeführt werden kann?
    a. send () muss Trotzdem Warten Es wird auf freie Plätze gewartet.
16. Auf dem Server sollen möglichst parallel viele RPCs bzw. Nachrichten verarbeitet werden. Wie sieht eine dafür geeignete Software-Struktur aus?
    a. Scheduler - Queue mit Prioritäten - Danach wird weiterverteilt
17. Treten bei wartenden (klassischen) RPCs Probleme des wechselseitigen Ausschlusses innerhalb einer verteilten Anwendung auf?
    a. Nein, da die Aufrufe nicht parallel ablaufen.
18. Was muss bei der Benutzung mehrfacher asynchroner send()-Operationen innerhalb einer verteilten Anwendung beachtet werden?
    a. Es entsteht ein kritischer Abschnitt
19. Beim Starten eines verteilten Systems müssen die beteiligten Komponenten erzeugt werden. Wie kann dies realisiert werden? Skizzieren Sie eine prinzipielle Lösung.
    a. Es wird eine anwendungslokale Instanz der betroffenen Komponente gebildet.
    b. Eine Anwendung besteht aus mehreren Instanzen von Komponenten auf mehreren Knoten
20. Wenn unter einer Komponente das Äquivalent einer Klasse verstanden wird, was wäre dann bei den Komponenten eine Instanz?
    a. Objekt
21. Warum sollten alle laufenden Anwendungen jeweils eine eindeutige ID besitzen? Was kann anhand dieser ID bestimmt werden?
    a. Damit nicht alle Instanzen gemeinsam genutzt werden. Die ID bestimmt welche Instanz einer Komponente zu welcher Anwendungsinstanz gehört.
22. Auf einem Server liegen z.B. zwei Instanzen derselben Komponente. Nun wird eine Nachricht empfangen und muss den Instanzen zugeordnet werden. Beschreiben Sie dazu den prinzipiellen Algorithmus.
    a. ??? Sofern IDs vorhanden sind, werden die Instanzen anhand dieser zugeordnet
23. Wie kommen in einem verteilten System Deadlocks zustande? Konstruieren Sie dazu ein Beispiel.
    a. Rekursion bzw. gegenseitiges Warten
24. Nennen Sie mindestens zwei Gründe, warum jeder RPC-Aufruf eine eindeutige Nummer haben sollte.
    a. Mehrere nichtwartende send() ausführbar
    b. Ergebnisse lassen sich leicht zuordnen

Fragebogen 8 - Namensdienste & Namens-Pepository

1. Bei den Namensdiensten wird zwischen Adressen und Namen unterschieden. Worin liegen die Unterschiede? Definieren Sie beide.
    a. Name - Ortsunabhängige, symbolische Identifikation (Websiteadressen, Email-Adressen)
    b. Adresse - Identifikation eines Ortes oder eimes Objektes an diesem Ort (z.B. IP oder RAM)
2. Was unterscheidet einen Namen für einen Dienst von einem Namen von einem Server?
    a. Dienstname ist die Abstraktion von Namen von Servern.
3. Welche Aufgaben hat ein Namensdienst bzw. Verzeichnisdienst (Name Service, Directory Service)? Nennen Sie dazu drei typische Operationen dieses Dienstes.

    a. Allgemeine Informationen über Objekte, Personen oder Organisationen bereitstellen
4. In der Datenbank eines Namensdienstes werden verschiedene Informationen gespeichert. Nennen Sie dafür drei Beispiele.
    a. - Adresse - Name - Art des Dienstes – Kosten
5. Wenn zwei Klienten sich innerhalb eines Netzwerks finden wollen: was müssen sie dazu tun, wenn sie ohne Broadcast-Operationen auskommen wollen?
    a. Sich beim directory Server anmelden
6. Wofür steht das Kürzel DNS? Was ist die wichtigste Aufgabe des DNS-Dienstes?
    a. Domain Name Services
    b. Umsetzung symbolische Adresse nach IP
7. Nennen Sie ein Beispiel für einen Namensdienst.
    a. DNS-Dienst
8. Was ist beim DNS eine Top-Level-Domain? Was ist eine Domain?
    a. TLD - Länderkennzeichen und weitere (.com)
    b. Domain – Bereiche
9. Wie viele Subdomains kann eine Domain grundsätzlich besitzen? Und wie ist es in der Praxis?
    a. Beliebig viele, je nach Vertrag.
    b. Oftmals gibt es je Niederlassung einer Firma eine Subdomain.
10. Da ein einziger DNS-Server nicht die gesamte Information weltweit speichern kann, sondern nur einen Teil, ist er nicht in der Lage, alle Anforderungen zu bedienen. Was muss gemacht werden in dem Fall einer fehlenden Information auf einem Server? Wer führt diese Maßnahmen durch?
    a. Die DNS Server handeln im rekursiven Modus und geben Anfragen untereinander weiter. Der Resolver befragt mehrere DNSServer sofern der erste DNS-Server keine Antwort hat.
11. Was ist eine DNS-Cache-Only-Server?
    a. Ein DNS Cache-OnlyServer hat keine eigene Zone. Er vermerkt bzw. lernt Anfragen und Ergebnisse.
12. Was sind LDAP-Server? Wofür steht LDAP? Und was hat LDAP mit der X.500-Empfehlung zu tun?
    a. Ein Protokoll zum Verzeichniszugriff.
    b. Lightweight Directory Access Protocol
    c. Basiert auf X.500 als vereinfachte Version.
13. Inwiefern besitzen die Einträge in der LDAP-Datenbank eine Hierarchie?
    a. Eintrag - Attribut – Wert
14. Alle Einträge in der LDAP-Datenbank werden eindeutig durch den Distingushed Name (DN) identifiziert. Wie ist dieser prinzipiell aufgebaut? Skizzieren Sie dies an einem Beispiel.
    a. cn=Burkhard Messer,ou=FB4,o=HTWBerlin,c=DE
    b. common name organisational unit organisation country

Fragebogen 9 – Rest

1. Wofür steht das Kürzel REST bzw. welches Prinzip wird dadurch zum Ausdruck gebracht?
    a. Representation State Transfer
    b. Übertragung der Repräsentation eines Zustandes.
2. Welche Operationen sind bei REST erlaubt? Nennen Sie vier
    a. GET POST PUT HEADER DELETE
3. Nennen Sie vier Prinzipien, die eine REST-API realisieren muss, damit sie RESTful ist.
    a. Client-Server-Prinzip (Pull)
    b. Zustandslos (ohne Sessions)

    c. Einheitliche Schnittstelle (nur HTTP-Methoden)
    d. Namen von Ressourcen (eindeutige URI)
    e. Relation zwischen Ressourcen und URI (Beziehungen werden mit Hilfe von URIs ausgedrückt)
4. Bei REST wird Zustandslosigkeit gefordert. Was ist darunter zu verstehen?
    a. Der Client liefert sämtliche Informationen in jedem Request.
    b. Im Server gibt es keine Sessions zu dieser Kommunikation.
5. Was für Vorteile hat die Zustandslosigkeit der Kommunikation bei REST? Nennen Sie zwei Vorteile und mindestens einen Nachteil.
    a. + Besseres (einfacheres) Skalieren
    b. + Gute Benutzung von Cache
    c. - Keine Prüfung der Identität möglich
    d. - Keine Prüfung von Autorisierungen möglich
    e. - Keine Realisierung von Transaktionen möglich
6. Was wird bei REST unter einer "safe method" verstanden? Nennen Sie die HTTP-Methoden, die diese Eigenschaft haben. Sind diese Methoden per se safe oder müssen sie safe realisiert werden?
    a. safe method - Ressource wird nicht verändert (Read-Only)
    b. GET HEAD OPTIONS TRACE
    c. Sie müssen safe realisiert werden
7. Was wird unter Idempotenz einer Operation verstanden? Welche HTTP-Methoden sollen idempotent sein? Sind diese Methoden idempotent oder müssen sie so realisiert werden?
    a. Keine Änderung der Semantik bei mehrmaliger Ausführung derselben Operation.
    b. GET, HEAD, PUT, DELETE
    c. Sie müssen idempotent realisiert werden
8. Unter welchen Bedingungen darf eine Ressource in einem Cache gehalten werden? Nennen Sie eine HTTP-Methode, deren Paketinhalt in jedem Fall in einem Cache gehalten werden kann.
    a. Read-Only Operationen können im Cache gehalten werden. Schreibende Operationen sollen nicht im Cache gehalten werden.
    b. GET, HEAD
9. Was wird bei REST unter einer Ressource verstanden? Nennen Sie dazu drei Beispiele.
    a. Alles, was im Web adressiert werden kann.
    b. Datei mit HTML
    c. Datei mit PHP-Code
    d. Grafiken
    e. Ladbare Klassen oder Module
10. Wenn mit PUT eine neue Ressource angelegt werden soll, entsteht ein Problem. Welches? Nennen Sie dazu zwei Lösungen.
    a. Es ist nicht vordefiniert, was bei PUT passiert.
    b. URL muss eindeutig sein
    c. wenn 2 mit gleichem Namen anfragen kommt es zur Kollision
    d. Lösung: 1. Anfragende müssen lange Zufallszahl generieren.
    e. 2. Es kann verboten werden
11. Nennen Sie mindestens einen wichtigen Unterschied zwischen folgenden Stilen: Objekt-Orientiert und REST.
    a. OO - Operationen primär im Server, es gibt RPCs
    b. REST - Operationen primär im Client, es gibt keine RPCs

12. Bei den URIs wird auch von Templates gesprochen. Was ist das? Geben Sie dazu ein Beispiel an.
    a. Eine Schablone zum Aufbau von URIs mit variablen Bereichen.
    b. http://....de/{Name}/{ID} Die Variablen werden durch aktuelle Werte ersetzt
13. Neben einfachen Ressourcen gibt es bei REST auch komplexe. Nennen Sie dafür zwei praxisnahe Beispiele.
    a. Zusammensetzung aus Strukturen oder Arrays, wie z.B. eine Kundendatenbank oder ein Server, der das Fernsehprogramm liefert
14. Die POST-Methode wird bei vielen APIs zum Anlegen von neuen Ressourcen benutzt. Was sind die Gründe dafür und welche Probleme hat diese Methode?
    a. PUT verlangt eine ID beim Anlegen einer Ressource
    b. POST generiert diese ID beim Anlegen einer Ressource (einfacher)
15. Warum sollte die POST-Methode bei REST möglichst vermieden werden?
    a. nicht definiert (universelle Methode)- ist nicht
    b. idempotent (jedes Mal wird eine neue (leere) Ressource angelegt)
16. Erläutern Sie die Idee der Benutzung von UUIDs. Was sind UUIDs und in welchen Fällen werden sie häufig benutzt?
    a. Eindeutige ID für Entities
    b. Genutzt bei PUT
17. Beschreiben Sie das POST-exactly-one-Verfahren.
    a. Anfrage eindeutige URI bei Server mit GET
    b. Ressource mit POST und der generierten URI anlegen Ressourcen werden nur einmal angelegt.
18. Was waren die Gründe für das Einführen der neuen HTTP-Methode PATCH?
    a. Umgang mit großen Datensätzen wie bspw. großen Bestellungen.
    b. In einer Anweisungsliste werden verschiedene Operationen abgearbeitet
19. Welche Probleme hat die Benutzung der Methode PATCH bei REST?
    a. PATCH ist nicht idempotent
20. Mit welchem Verfahren könnten die Probleme der HTTP-Methode PATCH gelöst werden?
    a. Durch eine Checksumme im Kopf (If-Match beim Request und Etag bei Response)
    b. PATCH : Methode wo komplexe Datenstruktur heruntergebrochen wird

Fragebogen 10 – OAuth

1. Worin bestehen die Ziele von OAuth? Skizzieren Sie dies anhand eines praktischen Beispiels.
    a. Das OAuth-Verfahren ermöglicht mit der Erlaubnis eines Benutzers (User) den Zugriff von Applikation2 auf Daten/APIs von Applikation1 für eine bestimmte Zeit ohne Austausch vertraulicher Login-Daten.
    b. Beispiel:
        i. Application 1 und 2 sind verschiedene Sites mit unterschiedlichen Accounts von Alice (und hoffentlich unterschiedlichen Passwörtern).
        ii. Alice loggt sich bei Application 1 ein, die auf ihre Daten zugreift.
        iii. Nun möchte Alice, dass Application 2 auf die Daten von Application 1 zugreifen kann, ohne dass Application 2 die Login Daten von Application 1 erhält.
2. Geht es bei OAuth um Authentisierung oder um Autorisierung?
    a. OAuth ist ein offenes Protokoll für die Autorisierung und Authentifizierung im Internet.

   b. Die Abkürzung OAuth steht für Open Authorization und ist ein offenes Protokoll, das eine sichere Autorisierung von Webservices oder mobilen Anwendungen ermöglicht, ohne Drittanbietern Passwörter offenlegen zu müssen.
   c. Laut Internet für beides. Im Namen steht halt schon Authentisierung
3. Die Version 2.0 von OAuth verlässt sich auf die Sicherheit einer anderen Realisierung, da keine kryptographischen Verfahren benutzt werden. Welcher? Hatte diese in den letzten Jahren Probleme mit der Sicherheit?
   a. Verlässt sich auf tls (Transport Layer Security)
   b. Ja, Probleme: Zertifizierung / Zertifikatsausstellung, Schlüsselerzeugung, Optionale Verschlüsselung
4. Welche vier Rollen sind bei OAuth wichtig?
   a. Resource Owner
   b. Resource Server
   c. Client
   d. Authorization Server
5. Was wird bei OAuth unter Credentials verstanden?
   a. Vertrauliche Daten zur Authentisierung oder Autorisierung
6. Bei OAuth wird von Profilen gesprochen. Was ist das? Und welches Profil wird am meisten in der Praxis verwendet?
7. Welche Arten vom Token werden bei OAuth benutzt? Beschreiben Sie die Funktion von zweien davon.
   a. Token = Fälschungssichere, geheime Information für die Autorisierung (anderer Name für Ticket)
   b. Access Token (Bearer Token)
      i. dienen zur Übertragung von Credentials
      ii. haben eine bestimmte Gültigkeitsdauer (Minuten bis Monate)
      iii. beinhalten mit dem scope-Parameter weitere Informationen
   c. Refresh Token
      i. dienen der Verlängerung der abgelaufenen Gültigkeitsdauer
      ii. sonst wie Access Token
8. Der Client muss im Rahmen einer Registrierung Informationen festlegen lassen. Nennen Sie zwei wichtige Informationen.
   a. Client-ID (öffentlich)
   b. Client-Geheimnis (geheim)
   c. Client-Typ (öffentlich)
   d. Umleitungs-URI/Redirection URI (geheim)
9. Skizzieren Sie den prinzipiellen Protokollablauf bei OAuth 2.0.
   a. Authorization Request an User: Entweder direkt über WebFormular oder vermittelt über Authorization Server
   b. Authorization Response vom User: Entweder es wird abgelehnt oder ein Authorization Grant wird geliefert.
   c. Access Token Request: Mit dem Authorization Grant fordert der Client einen Access Token an.
   d. Access Token Response: Authorization Server authentisiert den Client mittels Credentials aus der Registrierung, prüft den Authorization Grant und liefert im positiven Fall einen Access Token zurück.
   e. Resource Request: Der Client benutzt eine API oder greift auf Resourcen beim Resource Server zu. 6)Resource Response: Resource Server antwortet entsprechend der Operation.
10. Bei OAuth 2.0 wird von Weiterleitungen gesprochen. Beschreiben Sie, was damit auf der Ebene von HTTP gemeint ist.

    a. Ist der Status-Code eines Responses eine Weiterleitung so wird ein erneuter Request an die neue URL gesendet. Dies kann wiederum zu einer Weiterleitung führen, so dass eine Kette von Weiterleitungen entsteht.

11. Bei den Grant Types (OAuth 2.0) wird ein dieser Typen davon fast immer realisiert, welcher? Beschreiben Sie kurz für welche Anwendungstypen dieser Typ geeignet ist.
    a. Authorization Code Grant Type Typisch für Web-Server-Anwendung
    b. Einer der am häufigsten verwendeten Grant-Typen ist „authorization-code“.
    c. Um beispielsweise den Datenspeicherdienst Dropbox nutzen zu können, muss ein neuer Benutzer sich dort zunächst registrieren. Alternativ bietet Dropbox jedoch auch ein Login über ein bestehendes Google-Konto an. Nutzt man diese Möglichkeit, so werden die bestehenden Benutzerdaten aus dem eigenen Google-Konto als Autorisierung an Dropbox übertragen und das Konto bei Dropbox aktiviert.