Nennen Sie drei wichtige Eigenschaften von Verteilten Systemen (2019-01)

- **Transparenz**:
  - Verschiedene Aspekte der Verteiltheit vom Client verstecken (Zugriff, Ort, Migration, Relocation, Replikation, Concurrency, Ausfall)
  - Kann durch die Bereitstellung von Lower Level Services erreicht werden
  - Client nutzt Services.
- **Offenheit:**
  - Bietet Services entsprechend von Standard-Regeln (Syntax und Semantik: Format, Inhalt, Bedeutung).
  - Formalisiert in Protokollen.
  - Interfaces (IDL): Semantik oft informell
    - Complete => Interoperabilität: Kommunikation zwischen Prozessen
    - Neutral => Portabilität: Different Implementierung von Interfaces
  - Flexibilität: Komposition, Konfiguration, Replacement, Extensibility (CBSE)
- **Skalierbarkeit:**
  - Die Fähigkeit eine Verteilten Systems zu wachsen um der Nachfrage standzuhalten
    - Größe
    - Geographisch
    - Administrativ
  - System muss effektiv bleiben
  - System und Applikationssoftware sollte sich nicht ändern
  - Trade-Off scalability/security

Zwei Eigenschaften von Transparenz erklären

- **Access**: Unterschiede in der Datenrepräsentation und wie auf eine Ressource zugegriffen wird verstecken.
- **Location**: Verstecken, wo sich die Ressource befindet
- **Migration**: Verstecken, dass eine Ressource woanders hin verschoben werden könnte
- **Relocation**: Verstecken, dass eine Ressource, die gerade verwendet wird, verschoben wird
- **Replication**: Ressource verstecken, die repliziert wurde
- **Concurrency**: Verstecken, dass eine Ressource von mehreren Clients gleichzeitig benutzt wird
- **Failure**: Ausfall und Wiederherstellung einer Ressource verstecken

Namensraum und Namensdomäne

- **Namensraum**
    - beinhaltet alle gültigen Namen die erkannt und verwaltet werden
    - Name muss nicht zu Entität gehören
    - Alias: Name verweist auf anderen Namen

- **Namensdomäne**
    - Namensraum mit einer einzelnen administrativen Autorität die Namen für den Namensraum verwaltet

Native/Hosted VM Monitor Erklären (2018-03-14)

Ein VM Monitor ist eine abstrahierende Schicht zwischen tatsächlicher vorhandener Hardware (ggf. auf dem System bereits installiertem Betriebssystem)und weiteren zu installierenden Betriebssystemen.

- Native VM Monitor
    - Ein separater Software Layer simuliert die Instruktionen der Hardware
    - Setzt direkt auf der Hardware auf
    - Hardware muss den VM Monitor unterstützen
- Hosted VM Monitor
    - Verwendet das existierende Betriebssystem
    - z.B.: Virtual Box

Stateless/Stateful Server Erklären + Beispiele (2018-03-14)

Stateless

- Keine Informationen über den Zustand des Clients speichern
- Clients und Server sind unabhängig voneinander
- Zustandsinkonsistenten durch einen Client- oder Server-Crash werden reduziert
- Möglicher Verlust von Performance
- z.B.: Blogs

Stateful:

- Persistente Informationen über den Client speichern
- Performance zuwachs möglich, da der Server zusätzliche Informationen zum Client hat
- z.B.: Webshop mit Warenkorb

Was versteht man unter Internet of Things (2019-01)

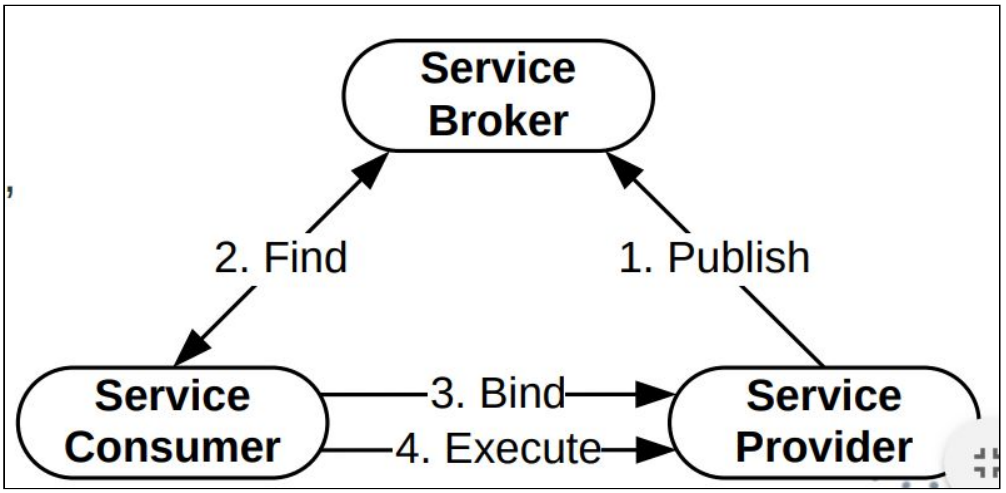
Physische Objekte

- sind nahtlos ins Internet integriert
- sind aktive Teilnehmer in Business Prozessen

- werden zu “Smart Objects”

Das Dreieck von SOA (Service-Oriented Architecture) (2019-01)

- Service-oriented Architecture
    - IT-Architektur aus einzelnen Services zusammengefügt. Eigenständige Softwarekomponenten mit eindeutiger Funktionalität
    - Komplexe Systeme entstehen durch die Kopplung von einzelnen Services
        - Service-based workflows
        - Mashups
    - Es ist auch möglich einzelne Service auszuführen
- Rollen in einer Service-orientierten Architektur
    - Service Anbieter
    - Service Konsument
    - Vermittler



Was versteht man unter Abstraction, Separation of concerns und Information hiding (2019-01)

- **Abstraction**
    - Client, server, service
    - Interface versus Implementation
- **Information hiding**
    - Interface Design
- **Separation of concerns**
    - Layering (filesystem example: bytes, disc, blocks, files)
    - Client and server
    - Components (granularity issues)

Lease - die Drei Arten erklären (2018-03-14)

Lease:

- Ein Lease ist ein Versprechen vom Server, dass er einem Client solange Updates pusht, bis das Lease von diesem Client abgelaufen ist.
- Wenn der Lease abgelaufen, dann muss der Client Informationen pullen.
- Leases werden verwendet, um den Server zu entlasten.

Age-based

- Wenn sich das Objekt lange Zeit nicht geändert hat und es sich zukünftig lange nicht ändern wird, dann verwendet man eine große Lease Zeit.

Renewal-frequency-based

- Je öfter ein Client ein spezifisches Objekt abfragt, desto länger wird der Lease des Clients für dieses Objekt.

State-based

- Je mehr ein Server belastet wird, desto kürzer ist die Lease Zeit.

Zwei Vorteile von Virtualization aufzählen + Beschreiben (2018-06-06)

Hardware ändert sich schneller als Software

- Für jede neue Hardware will man keine neue Software schreiben. Betriebssystem nimmt die Arbeit ab

Einfache Portabilität und Code-Migration

- Wechsel zwischen Systemen. Bsp.: Java VM

Fehlertoleranz: Isolation von Ausfällen, die durch Fehler oder Sicherheitsproblemen verursacht wird

- Ein Systemfehler bringt nicht das ganze System zum Absturz, sondern nur das VM-System

Warum sind Kommunikationsprotokolle in Layern designed? (2018-06-06)

Schichtenmodelle stellen eine sehr hohe Flexibilität zur Verfügung. Dem Webbrowser ist es egal, ob WLAN oder LAN verwendet wird, solange ein Zugriff zum Web besteht oder ob HTTP(S) über IPv4 oder IPv6 geroutet wird.

3 Arten der Redundanz erklären (2018-12-12)

- **Physical Redundancy**
  - Zusätzliche Komponenten hinzufügen (Backup Server)
- **Time Redundancy**

    - Requests wiederholen (Retransmissions in TCP/IP)
- **Information Redundancy**
    - Extra Information hinzufügen (parity bit)

Unterschiede: Transiente- und persistente Kommunikation (2018-12-12)

Persistent

- Nachrichten werden am Server solange gespeichert, bis sie zugestellt werden können.

Transient

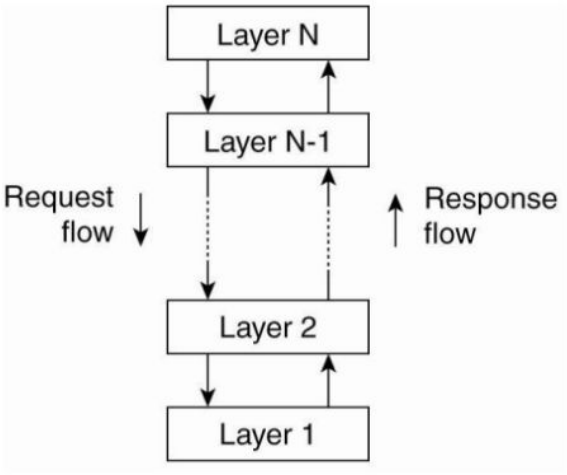
- Nachrichten werden vom Server gelöscht, wenn sie nicht zugestellt / weitergeleitet werden können.

Failure, Error & Fault erklären und wie sie zusammenhängen

- **Failure:** Anzubietende Dienst wird entweder überhaupt nicht oder fehlerhaft angeboten.
- **Error:** Abweichung des tatsächlichen Systemzustands von dem wahrgenommenen Zustand.
- **Fault:** Ursache eines Fehlers (Software bug)
- Kette: Fault → Error → Failure

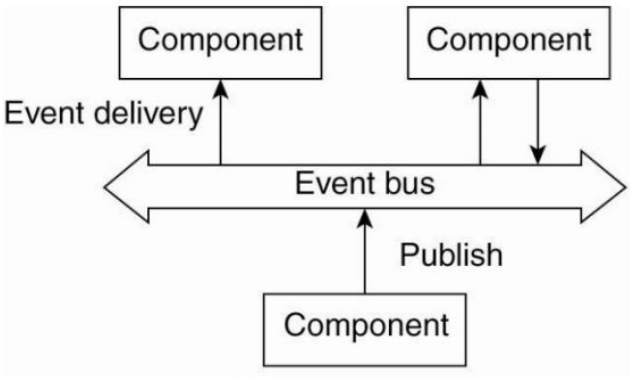
Unterschiede zwischen Layered- und Event-based Architektur erklären

Schichtenarchitektur



- Schichten unabhängig voneinander
- Anfragen werden von Schicht zu Schicht weitergegeben und abgearbeitet

Event-Based Architektur



- ein zentraler Event-Bus, der Events an verschiedene Komponenten weiterleitet
- Jeder Komponent kann Events an den Event-Bus schicken, um diese anderen Komponenten zukommen zu lassen

Prozessor, Thread & Prozess erklären

Prozessor

- Stellt ein Set von Instruktionen mit der Fähigkeit der automatischen Ausführung einer Serie von Instruktionen zur Verfügung.

Prozess

- Ein virtueller Software Prozessor in dessen Kontext ein oder mehr Threads ausgeführt werden.

Thread

- Ein **minimaler** Software Prozessor in dessen Kontext eine Serie von Instruktionen ausgeführt wird.

Wie können Prozess gruppiert werden (Flat Group + Hierarchical group) erklären

Flat group

- Gute Fehlertoleranz da ein sofortiger Informationsaustausch mit allen Mitgliedern erfolgt
- Kann zu einem Overhead führen
    - die Kontrolle ist komplett verteilt
    - Votings sind nötig
- Schwieriger in der Implementierung als Hierarchical group

Hierarchical group

- Kommunikation erfolgt durch einen Koordinator und mehreren workers
- Nicht wirklich fehlertolerant oder skalierbar
- Einfacher in der Implementierung

Drei Eigenschaften von Secure Channels (2019-01)

- Beide Parteien wissen,
    - … wer auf der anderen Seite ist (**Authentifiziert)**
    - …, dass die Nachricht nicht verändert wurde (**Integrität**)
    - …, dass die Nachricht nicht abgehört werden kann (**Vertraulichkeit**)

Forward Pointers, Vor- und Nachteile (2019-01)

- Wenn sich eine Entität bewegt, dann hinterlässt es einen Pointer zu seiner nächsten Location
- **Vorteile**:
    - Einfaches deferenzieren, da der Kette an Pointern gefolgt wird
- **Nachteile**:
    - Je länger die Kette, desto höher die Chance für eine Unterbrechung → kein Zugriff mehr möglich
    - Je länger die Kette, desto höher die Latenz

Name vs. Identifier (2019-01)

- **Name**:
    - Abfolge von Zeichen, um eine Entität in einem Kontext zu benennen.
    - Der Name hat keine Bedeutung, es sind zufällige Zeichen
- **Identifier**:
    - Name der einzigartig ist und genau eine Entität bezeichnet.
    - Der Identifier ist eindeutig und bezieht sich nur auf eine Entität
    - Jede Entität wird nur durch maximal einen Identifier identifiziert

Sie betreiben einen Server. Erklären Sie warum man einen Multi-Thread Server einem Single-Thread-Server bzw. einem Multi-Prozess-Server vorziehen sollte. (2019-01)

- Threads teilen sich einen Speicherbereich - Kontextswitch kann unabhängig vom OS geschehen.
- Prozess wechseln ist teurer, da das OS involviert werden muss.
- Erstellen und Zerstören von Threads viel günstiger als bei Prozessen
- Ein Server ist Ziel von mehreren/vielen Usern → Last kann viel besser auf mehrere Worker Threads aufgeteilt werden (Parallelisierung)
- Bei blockierenden Aktionen müssten bei einem Single Thread Server die User warten, bis der Block wieder freigegeben wird.

Was ist ein verteiltes System?

- Ein verteiltes System ist eine Ansammlung von Computern, welche durch ein Netzwerk verbunden und durch Software unterstützt sind, was ihnen die Möglichkeit gibt, als eine Einheit zu arbeiten.
- Komponenten: Hardware, Software, Netzwerk

Welche Typen von verteilten Systemen kennen Sie?

- Object/component based (CORBA, EJB, COM)
- File based (NFS)
- Document based (www, Lotus Notes)
- Coordination (or event) based (Jini, JavaSpaces, publish/subscribe, P2P)
- Resource oriented (GRID computing, Cloud Computing, MANET)
- Service oriented (Web services, Cloud, P2P)

Allgemeine Arten

Welche Strategie ist für folgende Anwendungsszenarien besser geeignet (P2P oder Cloud Service): a. Zentralisierte File Hosting System, b. Netzwerk um Medieninhalte untern Benutzern auszutauschen

- P2P ist gut geeignet, um Dateien unter Usern auszutauschen, da User direkt mit anderen Usern kommunizieren kann
- Cloud ist gut geeignet, um ein Zentralisiertes File Hosting zu betreiben

Die 8 Irrtümer von verteilten Systemen

1. Das Netzwerk ist verlässlich
2. Latenz ist null
3. Bandbreite ist unendlich
4. Das Netzwerk ist sicher
5. Topologie ändert sich nicht
6. Es gibt nur einen Administrator
7. Transportkosten sind null
8. Das Netzwerk ist homogen

Was ist QoS (Quality of Service)?

QoS ist ein Konzept, in dem Clients angeben können, welches Level von Services sie benötigen. So ist für zB.: Voice over IP schnell Übertragung wichtig, beim Onlinebanking steht die Sicherheit im Vordergrund. Alles zusammen ist jedoch nicht möglich.

BEN: Best Effort Network: Alles wird gleich behandelt. Grundlage für Netzneutralität.

Was ist ein Thread? Was ist ein Prozess?

Threads und Prozesse sind logische Einheiten, die eine Sammlung von Instruktionen abarbeiten können. Threads laufen auf Software-Prozessoren (virtueller Prozessoren). Ein Prozess hat im Gegensatz zu einem Thread seinen eigenen Adressraum.

Prozessor: Bietet ein Set von Anweisungen mit der Fähigkeit der automatischen Ausführung einer Serie von Anweisungen. Beinhaltet alle Dinge, wie einen Stackpointer und Adress Register.

Thread: Ein minimaler Software Prozessor, in dessen Kontext eine Serie von Anweisungen ausgeführt werden kann. Beinhaltet Teile des Prozessorkontexts.

Prozess: Ein Software Prozessor, in dessen Kontext eine oder mehrere Threads ausgeführt werden können. Prozess haben im Gegensatz zu Threads einen eigenen Adressraum und Speicher.

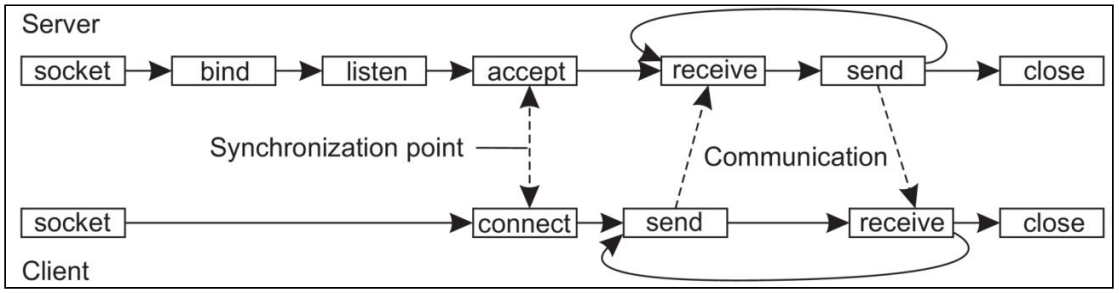
Multicast erklären

- Multicast bedeutet, dass ich Daten zu mehreren Empfängern schicken
- Publish/Subscribe Konzept
    - Subscriber hören auf Veröffentlichungen eines bestimmten Typs X
    - Wenn nun eine Nachricht des Typs X wird sie dem Message-Broker übergeben, der wiederum alle Subscriber informiert
- Application-Level Multicast:
    - basiert auf einem überliegendem Netzwerk, aufgebaut aus direkten Verbindungen zwischen Peers, ein “Overlay-Netzwerk” auf einem bestehenden Netzwerk vom physikalischen Netzwerk unabhängig, da vom TCP/IP Layer abstrahiert, separates addressing Schema
    - z.B.: Peer2Peer Systeme
    - Overlay-Netzwerk wird für Multicasting verwendet Flooding-based Multicasting

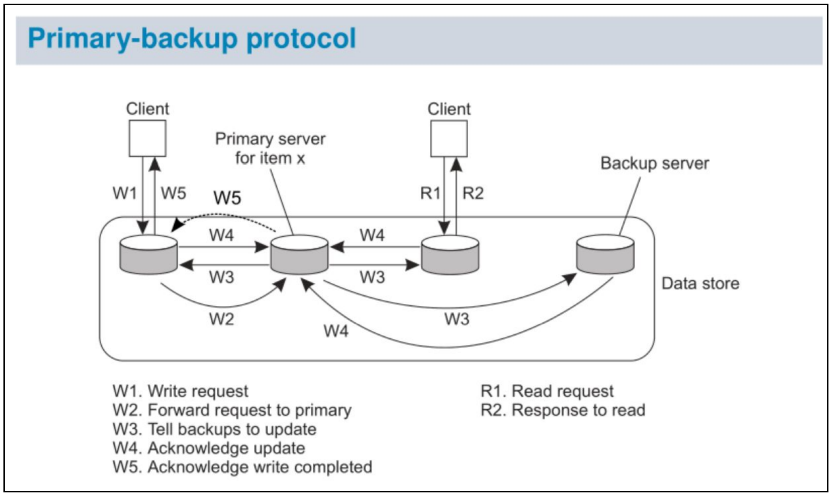
Socket Primitive Methoden zeichnen und erklären (2019-01)

- **Socket**:
    - Erzeugt einen neuen Kommunikationsendpunkt
- **Bind**:
    - Dem Socket eine lokale Adresse zuweisen
- **Listen**:
    - Bereitschaft eine neue Verbindungen zu akzeptieren, signalisieren
- **Connect**:

    - Versuchen eine Verbindung zum Socket herzustellen
- **Accept**:
    - Eingehende Anfrage akzeptieren
- **Send**:
    - Daten zum Socket schicken
- **Receive**:
    - Daten vom Socket empfangen
- **Close**:
    - Verbindung freigeben



Primary backup protocol erklären und Schritte einzeichnen (2019-01)



- Implementiert das **sequential consistency model**
    - Alle Schreiboperationen werden durch den primären Server geordnet an die verbleibenden Server geliefert.
    - Das Lesen der lokalen Kopie ergibt den aktuellsten Wert.
    - Lesen ist schnell. Schreiben ist langsam.
    - Wenn ein Knoten nicht verfügbar ist, dann kann keine Schreiboperation ausgeführt werden. Nicht widerstandsfähig gegenüber Netzwerk oder Knotenfehler.

3 Anwendungsfälle für Cloud Computing (2019-06)

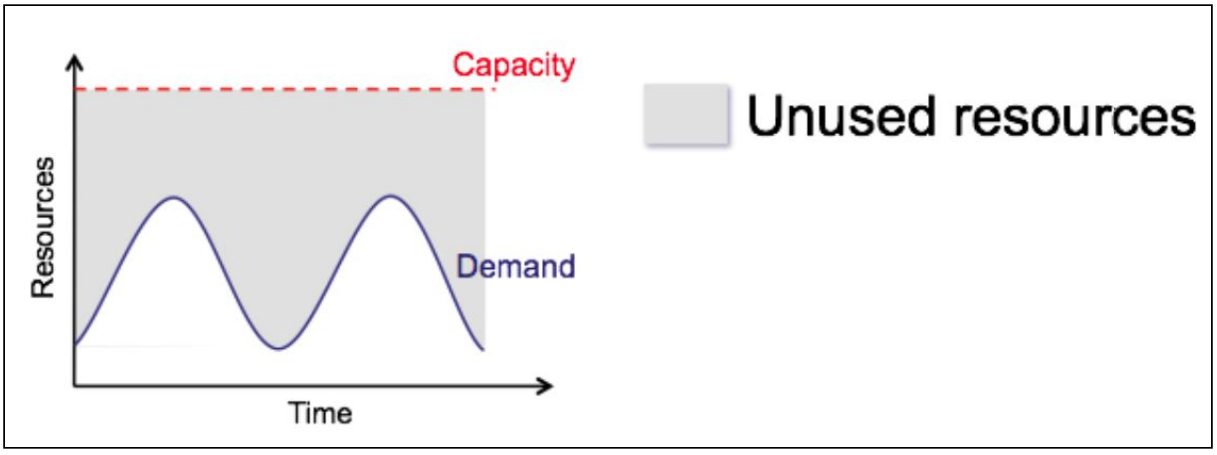
1. Bedarf für einen Service schwankt mit der Zeit
    a. Spitzenlast
2. Bedarf ist im Voraus nicht bekannt
    a. Für ein Startup
3. Batch-Analyse
    a. 1000 EC2 Instanzen kosten für eine Stunde so viel wie eine Instanz für 1000 Stunden.

Erklären Sie Namespace, Namingdomain und Name-Resolution (2019-06)

- **Namespace:**
    - Enthält alle Namen, die von einem Service verwaltet und erkannt werden.
        - Ein valider Name ist nicht an eine Entität gebunden
        - **Alias**: Ein Name bezieht sich auf einen anderen Namen
    - Namen sind in einem Namespace so organisiert, dass sie als Graph dargestellt werden können.
    - Jeder Blatt Knoten repräsentiert eine Entity.
- **Name resolution:**
    - Ein Prozess um **Informationen** und **Attribute** für einen Namen zu suchen.
- **Namingdomain:**
    - Namespace mit einer einzigen administrativen Authority, welche Namen für den Namespace verwaltet. (Bsp.: nic.at ist für alle *.at Domains zuständig)

Overprovisioning erklären (+Skizze) (2019-06)

- Mehr Computerressourcen zur Verfügung stellen, als eigentlich notwendig ist.



IaaS, SaaS, PaaS erklären + Beispiele (2019-12)

Cloud Infrastructure as a Service (IaaS)

- Computer Infrastruktur als einen Service liefern (Virtuelle Maschinen, Speicher)
- Bsp.: Amazon EC2, Amazon S3

Cloud Platform as a Service (PaaS)

- Computer Plattform und Lösungsstack als einen Service liefern.
- Bsp.: Google App Engine

Cloud Software as a Service (SaaS)

- ERP Software als einen Service, Salesforce.com

Unterschied zwischen **Accuracy** und **Precision** erklären (2019-12)

Accuracy

- Abweichungen mit einem Referenzwert in einem bestimmten Rahmen halten. (externe Synchronisation)

Precision

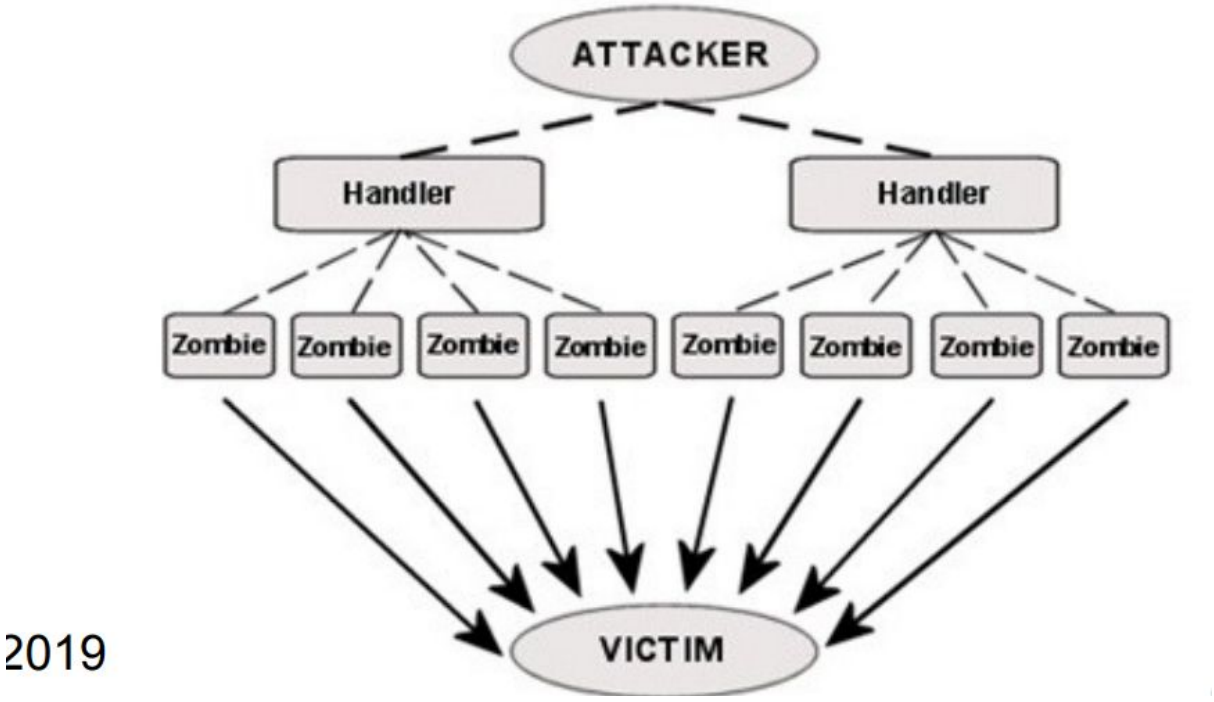
- Abweichungen der Zeit in einem verteilten System zwischen den Maschinen in einem bestimmten Rahmen halten. (interne Synchronisation)

Wann soll push statt pull verwendet werden

1. Server soll zustandslos sein
2. Client soll Updates so schnell wie möglich erhalten
3. Daten werden häufig geupdated, aber nur selten gelesen
4. Clients kommen und gehen

DDoS (erklären + skizzieren) (2016-06)

- Angreifer hackt viele Computer und nutzt diese als Zombies für einen verteilten Angriff auf einen Server und schafft es durch einen Request Overload, den Server in die Knie zu zwingen (bis zum Absturz).
- Schwer sich zu verteidigen (ISP-Level)
- Schwer den Angreifer zu identifizieren



2019

Unterschied zwischen Gruppe und Rolle (2016-06)

- Gruppe
  - Benutzer gehören zu einer spezifischen Gruppe. Jede Gruppe hat zugehörige Rechte.
- Rolle
  - Nicht zwischen Nutzern unterscheiden sondern zwischen Rollen, die sie spielen können. Die Rolle wird beim Login bestimmt. Rollenänderungen sind erlaubt.

Zählen Sie die Probleme mit Schlüsseln auf

- **Keys wear out**
  - Je mehr Daten mit dem gleichen Schlüssel verschlüsselt werden, desto einfacher ist es, diesen Schlüssel herauszufinden
- **Danger of Replay**
  - Wenn der gleiche Schlüssel für mehrere Kommunikation-Sessions verwendet wird, können alte Nachrichten in die aktuelle Session eingefügt werden.
- **Compromised Keys**

    - Wenn ein Schlüssel einmal kompromittiert wurde, dann kann dieser Schlüssel nicht mehr verwendet werden.
- **Temporary Keys**
    - Nicht vertrauenswürdige Komponenten sollten kein Wissen über die guten Schlüssel haben.

Zählen Sie die Security Mechanismen auf (+Erklärung)

- Encryption
- Authentication
- Authorization
- Auditing

Security Threats

- **Interception**
    - Inhalt von übertragenden Nachrichten lesen
- **Interruption**
    - Nachrichten Transfer verhindern
- **Modification**
    - Nachrichteninhalt ändern
- **Fabrication**
    - Nachrichten einfügen

Beschreiben einer Technik um Scalability zu erreichen

Eine der verbreitetsten Techniken ist die Replikation von Entitäten. Bsp.: Durch das Aufstellen zusätzlicher physikalischer Speicher mit identischen Services, sowie dem Einsatz eines Load Balancers kann die Performance eines Systems/Services erhöht werden, da die Auslastung der einzelnen Server reduziert wird.

Nicht funktionale Anforderungen von VS aufzählen und erklären

- **Integrität**
    - Die ungewollte Modifikation von Ressourcen muss verhindert werden
- **Security**
    - Die Sicherheit in und für ein verteiltes System muss gegeben sein; schwerwiegende Konsequenzen bei Fehlern müssen ausgeschlossen sein.
- **Verfügbarkeit**
    - Das System muss die bestmögliche Verfügbarkeit aufweisen

Drei wesentliche Eigenschaften von Peer-to-Peer Systemen

- Die Entitäten eines P2P-Systems kommunizieren untereinander durch direkte Verbindungen, nicht über eine zentrale Steuereinheit
- Alle Entitäten eines P2P-Systems sind gleichgestellt, es gibt keine höherwertigen Entitäten
- Jede Entitäten konsumiert und stellt die gleichen Services bereit (Bsp.: Torrent)

LDAP erklären und wo der Einsatz Sinn macht

d

Welche Probleme kann die Heterogenität in VS bereiten? (2014-03)

In einem heterogenen System muss zuerst sichergestellt werden, dass alle Prozess miteinander kommunizieren können. Normalerweise wenn unterschiedliche Systeme kommunizieren, müssen eine Art an Umwandlung stattfinden. Dies erhöht die Komplexität eines Systems und damit auch die Fehler.

Wie hilft die Middleware, die Probleme mit Heterogenität zu minimieren? (2014-03)

Eine Middleware versucht diese Unterschiede zu verstecken und allen verschiedenen Prozesse ein einheitliches Interface zu präsentieren. Die Prozesse müssen sich darüber bewusst sein, dass sie mit unterschiedlichen Systemen kommunizieren und sollten daher in der Lage sein, Nachrichten, Daten, usw. einfach zu senden.

Die 3 Anforderungen einer Hash-Funktion angeben (2014-03)

1. Einfach zu berechnen
2. Unmöglich eine Nachricht zu rekonstruieren, wenn die Hash-Funktion gegeben ist
3. Kollisionsresistent
    a. Zwei unterschiedliche Nachrichten haben unterschiedliche Hashes.

Zwei Arten wie Prozesse organisiert werden können + Vorteile/Nachteile

- **Flat**:

    - Alle Prozesse sind gleichwertig und ein Crash eines Prozesses beeinflusst die anderen nicht. Kommunikation und Koordination sind komplizierter, allerdings ist eine Wahl unter den Prozessen notwendig

- **Hierachical**:
    - Ein Koordinator existiert in der Gruppe und balanciert den Work Load unter den Mitgliedern der Gruppe. Aber der Koordinator ist ein Single Point of Failure. Einfach zu implementieren

Cloud vs. Hybrid beschreiben

- **Public Cloud:**
    - Für die Öffentlichkeit, wird von einer Organisation besessen, die Cloud Dienste verkauft
- **Private Cloud:**
    - Wird nur von einer Organisation verwendet
- **Community Cloud:**
    - Wird von mehreren Organisationen verwendet
- **Hybrid Cloud:**
    - Zusammenschluss von 2 oder mehreren Cloud Modellen (Privat, Community, Public)

Flat Naming erklären

- Unstrukturierte/Flache Namen: Identifier haben keine strukturierte Beschreibung
- Einfacher Weg um Identifier zu repräsentieren
- Enthält keine zusätzlichen Informationen, um die Entität zu verstehen
- **Beispiele:**
    - MAC-Adresse
    - m-bit numbers in Distributed Hash Tables

Erklären Sie DNS

- Hierarchisch organisierten Namespace, wo jeder Knoten exakt einen eingehende Kante hat.
- Domain: Ein Unterbaum
- Domain-Name: Ein Pfad zum Root Knoten der Domain
- String-Repräsentation für einen Pfad-Namen
    - Beschriftungen aufzählen
    - Beschriftung bei einem Punkt separieren
    - Root wird durch einen Punkt repräsentiert

Erklären Sie Attribut basierte Benennung

- Ein **Tupel** (Attribute, Wert) kann verwendet werden, um eine **Eigenschaft** zu beschreiben.
    - (“country”, “Austria”), (“language”, “German”)
- Eine Sammlung von Tupeln kann verwendet werden, um eine **Entität** zu beschreiben.
- **Tupel** verwenden, um eine **Entität** beschreiben.
- **Namensauflösung**
    - Query Mechanismus
    - Query verwendet den ganzen Raum

Grund für Replikation

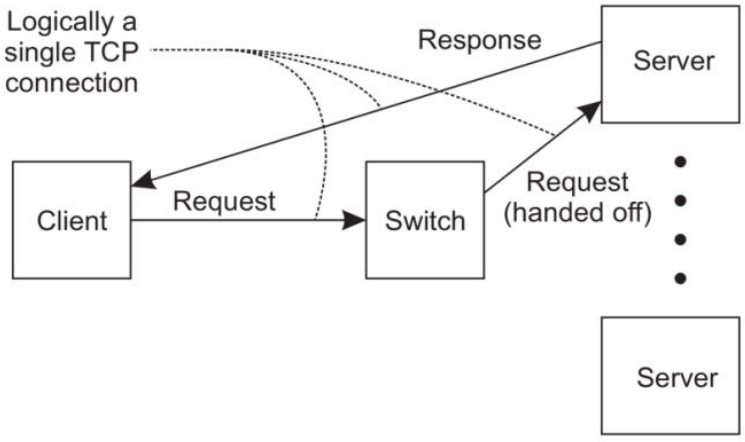
1. Performance und Skalierbarkeit
    a. Skalierung in Zahlen
        i. Mehr Replikate können mehr Clients handeln

    b. Skalierung in Geographischer/Topologischer Komplexität
        i. Skalierung in

Name a resource-oriented and a file-based Distributed System (2019-06-05)

- resource-oriented: GRID, Cloud, P2P, MANET
- file-based: NFS

TCP-Handoff (sketch) (2019-06-05)



- Client sendet Request an Switch
- Switch sucht besten Server aus und leitet Request an diesen Server weiter
- Switch muss alle Server kennen
- Server antwortet Client

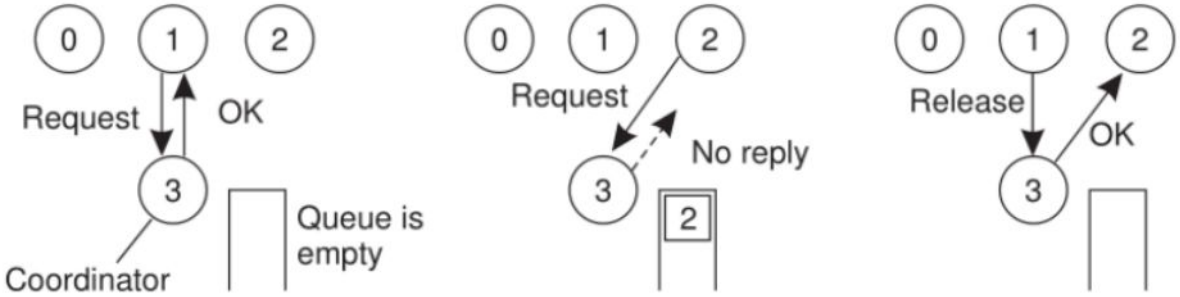
Lamport logical sequence + formula (2019-06-05)

- Verteiltes System soll sich global mit Passiert-vor Beziehung verhalten
- Wenn a, b Events im gleichen Prozess und a → b, dann muss gelten C(a) < C(b)
- Wenn a das Senden einer Nachricht ist, und b das Empfangen einer Nachricht, gilt C(a) < C(b)

Protection Domains

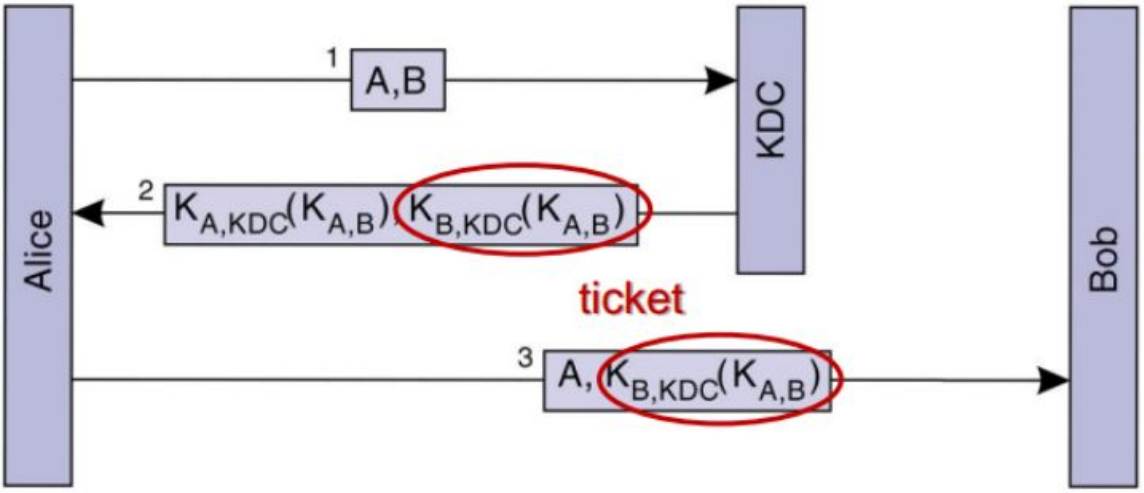
- feine Berechtigungen einzustellen (ACM, ACL, Capabilities) nicht immer sinnvoll
- Protection Domains Implementierungen:
    - Gruppen: User gehören zu Gruppen mit gewissen Berechtigungen
    - Rollen: keine Unterscheidung zwischen Usern sondern nur zwischen Rollen von Usern
- User stellen Zertifikate, zu welchen Gruppen/Rollen sie gehören

centralized mutual-exclusion (2019-06-05)



- Berechtigungs-basiert
- Koordinator erteilt Berechtigung
- Anfrage an Koordinator, ob Berechtigung schon vergeben wurde
- Anfragen in Queue gespeichert und abgearbeitet
- Vorteile:
    - kein aushungern
    - einfach
    - wenig Nachrichtenaustausch
- Nachteile:
    - Koordinator ist Fehlerquelle, Bottleneck
    - nicht skalierbar

Key distribution center mit ticket system (2019-06-05)



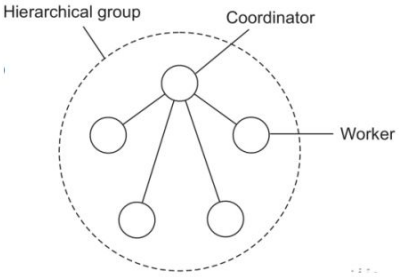
- stabiler, effizienter da nicht soviele Nachrichten geschickt werden
- Ticket wird von Alice verwendet, um Nachricht an Bob zu verschlüsseln

Failure Maskierung durch Redundanz

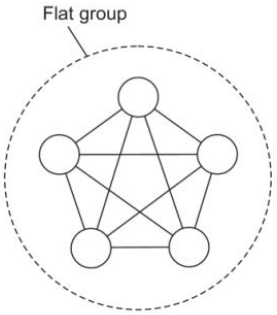
- Information Redundanz:
  - extra Informationen werden hinzugefügt
  - Bsp.: Paritätsbit, Error Correcting Codes
- Zeitliche Redundanz:
  - Anfrage wird erneut ausgeführt
- Physikalische Redundanz:
  - zusätzliche Komponenten werden hinzugefügt
  - Bsp.: Backup Server, RAID 1

Prozess Widerstandsfähigkeitsgruppen, Vorteile und Nachteile

- Hierarchische Gruppen (Master-Slave Ansatz)



    - **Vorteile**
        - einfach zu implementieren
    - **Nachteile**
        - Koordinator ist single-point-of-failure und Bottleneck
        - nicht fehlertolerant oder skalierbar
- Flache Gruppen (Peer-to-Peer Ansatz)



    - **Vorteile**
        - sehr fehlertolerant
        - nach außen ist die Gruppe ein Prozess
    - **Nachteile**
        - aufwändig zu implementieren
        - Konsens zwischen allen Komponenten muss gefunden werden (Voting)