JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

# Matura Ausarbeitung

# Dezentrale Systeme


**Systemtechnik**
**5BHITT 2015/16**

**Selina Brinnich**


**Version 2.0**

# Inhaltsverzeichnis

# Kompetenzen

- (K1) können die in dokumentenbasierten, nachrichtenorientierten und serviceorientierten Systemen eingesetzten offenen Dokumentenformate und Auszeichnungssprachen erläutern
- (K2) können Programmiertechniken in verteilten Systemen zur Realisierung von entfernten Prozeduren, Methoden und Objekten anwenden sowie webbasierte Dienste und Messaging-Dienste in solchen Systemen implementieren
- (K3) können Sicherheitskonzepte für verteilte, dezentrale Systeme entwickeln
- (K4) können den Datenbankentwurf in verteilten Systemen durchführen und zur dynamischen Generierung von Inhalten einsetzen

# 1. Cloud Computing und Internet of Things

## 1.1. Datenübertragung (K1)

Für die Übertragung von Daten zwischen mehreren Geräten sollte ein einheitliches Datenformat festgelegt werden, um die Kommunikation so einfach und effizient wie möglich zu gestalten. Im folgenden werden die beiden Datenformate XML und JSON genauer betrachtet.

### 1.1.1. XML

Extensible Markup Language, kurz XML, wird zur Beschreibung von Daten bzw. Datenstrukturen verwendet. Anders als bei HTML, wo lediglich festgelegt wird, wie die Daten letztendlich angezeigt werden sollen, wird bei XML hauptsächlich auf die Struktur der Daten wert gelegt. Dazu werden die Daten in Elementen organisiert. Der Elementname beschreibt die Daten des Elements und die Struktur der einzelnen Elemente definiert die Beziehung zwischen diesen.

Vorteile von XML:

- Es können eigene Tags definiert werden, dadurch kann besser auf die jeweiligen Anforderungen eingegangen werden
- Die Darstellung der Daten entfällt, dadurch kann der Fokus besser auf die Struktur gelegt werden, wodurch die Daten letztendlich besser beschrieben werden können und demnach verständlicher werden
- Ein XML Schema kann zur Validierung der Daten verwendet werden
- Mithilfe von XPath bzw. XQuery können Daten bei größeren Datenmengen einfacher gesucht und gefunden werden
- XML ist für den Menschen besser lesbar als bspw. HTML
- Komplexe Beziehungen zwischen Daten können einfach erstellt werden

Nachteile von XML:

- Bei größeren Datenmengen kann die Lesbarkeit der Daten deutlich eingeschränkt sein
- Die Byteanzahl ist, vor allem bei größer werdenden Datenmengen, relativ hoch im Vergleich zu anderen Datenformaten wie bspw. JSON

[1]

### 1.1.2. JSON

JavaScript Object Notation, kurz JSON, ist ein Datenübertragungsformat, bei dem der Fokus auf möglichst geringer Übertragungsmenge liegt. Dazu wird versucht, die Daten innerhalb des JSON-Dokuments möglichst kompakt und einfach darzustellen.

Vorteile von JSON:

- Geringe Byteanzahl, da die Syntax von JSON einfacher und kompakter gehalten wurde, als bspw. bei XML
- Gute Kompatibilität zu mehreren Programmiersprachen, daher wird meist keine Umwandlung der Daten benötigt und sie können direkt verarbeitet werden
- Gute Lesbarkeit, sowohl für Menschen als auch für Maschinen
- Einfachere und schnellere Verarbeitung der Daten, da die Syntax sehr simpel ist

Nachteile von JSON:

- Geringe Menge an verfügbaren Datentypen zur Beschreibung von Daten
- Keine Validierung gegen ein Schema möglich, kann zu Problemen führen, wenn die kommunizierenden Geräte sich nicht an dieselbe Struktur halten

[2,3]

### 1.1.3. Vergleich

Das Datenformat **XML** bietet viele Vorteile. Vor allem, wenn die Anforderung besteht, **Daten validieren** zu können, kann XML hier sehr hilfreich sein. Durch Schemadateien ist es relativ einfach, ein Format zu definieren um anschließend einzelne XML-Dateien damit abgleichen und validieren zu können. JSON im Gegensatz dazu bietet zwar auch eine Art Schema an, allerdings wird dies kaum verwendet.

Dennoch wird **JSON** mittlerweile bevorzugt verwendet, hauptsächlich aus dem Grund, dass es **weniger Overhead** mit sich bringt und damit die Daten schneller übertragen und, bedingt durch die Syntax von JSON-Dokumenten, auch schneller von Programmen verarbeitet werden können. [4]

# 2. Automatisierung, Regelung und Steuerung

=> n-1

# 3. Security, Safety, Availability

## 3.1. Verteilte/Replizierte Datenbanken (K4)

Die Datenbank ist in einem System für die Speicherung aller Daten zuständig, daher ist es besonders wichtig darauf zu achten, dass diese nicht ausfällt. Eine Möglichkeit dafür zu sorgen, dass zu jeder Zeit Daten vom System abgerufen werden können, ist Replikate der Daten zu erstellen. Dabei gibt es prinzipiell zwei Arten: Komplette Replikate von Datenbanken oder Teil-Replikate.

Bei **kompletten Replikaten** werden alle Daten einer Datenbank zusätzlich auch an (mehreren) anderen Standorten gespeichert. Dies hat den großen Vorteil, dass **Zugriffe lokaler** und damit **meist schneller** erfolgen können. Dazu sollten die Standorte möglichst so gewählt werden, dass sie sich dort befinden, wo besonders viele Zugriffe erfolgen. Nachteil bei kompletten Replikaten ist allerdings die **hohe Speicherkapazität**, die dafür benötigt wird. Wenn tatsächlich alle vorhandenen Daten an jedem der Standorte gespeichert wird, ist die benötigte Speicherkapazität dementsprechend hoch und mit jedem weiteren Replikat steigt diese zudem wieder enorm an. [5]

Allerdings werden zumeist gar nicht alle Daten an jedem der Standorte benötigt. Sollte dies der Fall sein, so sollten eher **Teil-Replikate** eingesetzt werden. Hierzu gibt es meist eine zentrale Datenbank, an der alle Daten gespeichert werden, und nur bestimmte benötigte Teile der Daten werden dann an einem gewissen Standort repliziert. Dabei können bei jedem Standort andere Daten gespeichert sein. Dadurch verringert sich die benötigte Speicherkapazität drastisch und bei einem Ausfall eines Standortes kann dennoch auf die entsprechenden Daten zugegriffen werden. [5]

Ein großes **Problem** bei beiden Arten ist allerdings die **Konsistenz**. Bei Änderungen an einem der Replikate müssen alle anderen Replikate, die diese Daten besitzen, ebenfalls die Änderungen mitbekommen, um die Daten konsistent zu halten. Dies wird mit unterschiedlichen Synchronisationsverfahren gelöst. [5]

## 3.2. Sichere Kommunikation (K3)

Eine sichere Kommunikation zwischen zwei Geräten ist in vielen Anwendungsfällen von Nöten. Um das Abhören von privaten Daten zu erschweren, kann eine Verschlüsselung der Nachrichten stattfinden. Es wird dabei zwischen drei grundlegenden Arten von Verschlüsselung unterschieden: symmetrische, asymmetrische und hybride Verschlüsselung.

### 3.2.1. Symmetrische Verschlüsselung

Die symmetrische Verschlüsselung ist prinzipiell die einfachste der drei Arten von Verschlüsselung. Sie beruht auf **einem einzigen Schlüssel**, den sowohl der Sender als auch alle gewünschten Empfänger besitzen müssen. Mit demselben Schlüssel kann die Nachricht beim Sender verschlüsselt sowie beim Empfänger entschlüsselt werden. [6]

**Vorteil**

Der große Vorteil bei dieser Art von Verschlüsselung ist die Einfachheit bzw. geringe Komplexität. Außerdem ist die symmetrische Verschlüsselung für gewöhnlich auch recht schnell und effizient, da die Nachricht mit demselben Schlüssel ver- und entschlüsselt werden kann. [6]

**Nachteil**

Das größte Problem, das hierbei auftreten kann, liegt beim Austausch des Schlüssels. Sowohl Sender, als auch Empfänger benötigen denselben Schlüssel, um ihre Nachrichten verschlüsseln zu können. Dazu muss einer der beiden einen Key erstellen und ihn an den jeweils anderen übermitteln. Sollte dabei allerdings ein Angreifer mitlauschen und den Key abhören, so kann er die Nachrichten, die im späteren Verlauf ausgetauscht werden, ganz einfach mit dem abgehörten Schlüssel entschlüsseln und damit lesen. Daher muss beim Schlüsselaustausch darauf geachtet werden, dass auch wirklich nur die gewünschte Person den Schlüssel erhält. Am einfachsten ist hier eine persönliche Übergabe des Schlüssels. [6]

**Beispiele: DES & AES**

DES (Digital Encryption Standard) und AES (Advanced Encryption Standard) sind die wohl bekanntesten symmetrischen Verschlüsselungsverfahren, wobei mittlerweile fast ausschließlich noch AES verwendet wird, hauptsächlich aufgrund der zu geringen Schlüssellänge von 64 Bit von DES. Dadurch konnte ein DES-Schlüssel innerhalb weniger Stunden mittels Brute-Force Attacke geknackt werden. AES hingegen arbeitet mit einer variablen Schlüssellänge

von bis zu 256 Bit, was einen Erfolg mittels Brute-Force Angriff deutlich verringert. Aber nicht nur die Schlüssellänge, sondern auch die gute Performanz von AES ist ein entscheidender Vorteil. [6]

### 3.2.2. Asymmetrische Verschlüsselung

Bei der asymmetrischen Verschlüsselung besitzt jede Person ein sogenanntes „**Keypair**", das aus einem **Public-Key** und einem **Private-Key** besteht, wobei der Public-Key jedem bekannt ist und der Private-Key geheim gehalten wird. Wenn der Sender dem Empfänger nun eine Nachricht zukommen lassen möchte, so verschlüsselt dieser mit dem Public-Key des Empfängers und sendet es an jenen weiter. Sobald die Nachricht mit einem Public-Key verschlüsselt wurde, kann sie nur noch mit dem dazu passenden Private-Key, also nur beim tatsächlichen Empfänger, entschlüsselt werden. [6]

**Vorteil**

Ein großer Vorteil beim asymmetrischen Verschlüsselungsverfahren ist die hohe Sicherheit. Im Gegensatz zum symmetrischen Verfahren ist die Schlüssellänge um ein Vielfaches größer und dadurch auch sicherer. Zudem gibt es durch die eigenen Keypairs pro Person auch nicht mehr das Problem der unsicheren Schlüsselübertragung. [6]

**Nachteil**

Ein (eher geringes) Problem ist hierbei, dass das Verfahren um einiges komplexer ist als das Symmetrische. Das Erstellen eines Keypairs nimmt mehr Zeit in Anspruch. Zudem gibt es dadurch nicht mehr die Möglichkeit, dieselbe verschlüsselte Nachricht an mehrere Empfänger zu übermitteln. Um mehrere Empfänger zu erreichen, muss die Nachricht für jeden Empfänger einzeln mit seinem jeweiligen Public-Key verschlüsselt und übertragen werden. [6]

**Beispiel: RSA & Signatur**

Beim RSA-Verfahren werden zwei voneinander unabhängige Schlüssel generiert, ein Public-Key, der veröffentlicht wird, und ein Private-Key, der geheim gehalten wird. Dies wird durch mehrere Berechnungen mit Primzahlen durchgeführt. Anschließend kann die Nachricht in Blöcke unterteilt und mit dem Public-Key ver- bzw. mit dem Private-Key entschlüsselt werden. Zusätzlich besteht die Möglichkeit Verschlüsselung und Entschlüsselung sozusagen umzukehren und damit eine elektronische Unterschrift zu realisieren. Dazu wird eine Nachricht vom Sender mit dem eigenen Private-Key verschlüsselt und anschließend versandt. Der Empfänger kann dann diese Verschlüsselung

mithilfe des Public-Key des Senders wieder rückgängig machen und damit feststellen, ob die Nachricht tatsächlich vom entsprechend erwarteten Sender kommt. Zusätzlich zur Verschlüsselung mit dem eigenen Private-Key kann natürlich auch noch mit dem Public-Key des Empfängers verschlüsselt werden, um nur dieser Person Zugriff auf die Nachricht zu ermöglichen. [7]

### 3.2.3. Hybride Verschlüsselung

Die hybride Verschlüsselung ist eine Kombination aus symmetrischer und asymmetrischer Verschlüsselung. Dabei wird ein symmetrischer Schlüssel generiert und eine Nachricht mit dem generierten Schlüssel verschlüsselt. Anschließend wird der symmetrische Schlüssel mit dem Public-Key des Empfängers verschlüsselt und sowohl die verschlüsselte Nachricht, als auch der verschlüsselte symmetrische Schlüssel an den Empfänger übermittelt. Der Empfänger kann dann zunächst den symmetrischen Schlüssel mit dem eigenen Private-Key entschlüsseln und mit diesem Schlüssel anschließend die eigentliche Nachricht entschlüsseln. [6]

**Vorteil**

Die hybride Verschlüsselung vereint die Sicherheit des asymmetrischen Verfahrens mit der Schnelligkeit des symmetrischen Verfahrens. Dadurch, dass die Nachricht nur mit dem symmetrischen Verfahren verschlüsselt wird, wird die eigentliche Verschlüsselung vor allem bei größeren Nachrichten deutlich schneller. Gleichzeitig ist aber auch das Problem des unsicheren Schlüsselübertragung gelöst, da der symmetrische Schlüssel mit dem sichereren asymmetrischen Verfahren verschlüsselt übertragen wird. [6]

**Nachteil**

Es entsteht ein wenig mehr Aufwand bei der hybriden Verschlüsselung, da sowohl ein Keypair auf beiden Seiten (Sender und Empfänger) vorhanden sein muss, als auch beim Sender ein symmetrischer Schlüssel zusätzlich generiert werden muss. Zusätzlich sollte darauf geachtet werden, dass der symmetrische Schlüssel eine hohe Qualität hat, sprich keine Systematik zu anderen Schlüsseln vorweist und eine entsprechend hohe Schlüssellänge aufweist, da sonst wieder das Problem eines erfolgreichen Brute-Force-Angriffs auftritt. [6]

# 4. Authentication, Authorization, Accounting

## 4.1. Begriffserklärung (K3)

**AAA** bezeichnet die drei Begriffe Authentication Authorization und Accounting. [8]

**Authentication** beschreibt den Prozess der Feststellung einer User-Identität (siehe folgendes Kapitel). [8]

**Authorization** bezeichnet die Überprüfung der Berechtigungen eines Users. In diesem Prozess wird dem User entweder die Erlaubnis erteilt, auf bestimmte Ressourcen zuzugreifen, oder der Zugriff wird ihm verwehrt. [8]

**Accounting** beschreibt das Messen bzw. Protokollieren von Ressourcenverbrauch, Zugriffen auf Ressourcen, usw. eines Users. [8]

## 4.2. Arten von Authentication (K3)

Anmeldedaten werden mittlerweile bei fast jeder Applikation benötigt, um die Identität des Users sicherzustellen. Dabei gibt es mehrere Arten, wie diese Anmeldedaten aussehen können. Grundsätzlich wird hier zwischen drei Arten unterschieden: Wissen, Besitz und Biometrie.

Die am häufigsten verwendete Methode der Authentifizierung sind **Passwörter**. Passwörter zählen zur Kategorie **Wissen**. Bei vielen Applikationen gibt es die Möglichkeit sich mit einem selbst gewählten Passwort anzumelden und damit seine Identität zu bestätigen. Allerdings gibt es bei Wissen immer ein großes Problem: Was ist, wenn jemand anders dasselbe Wissen über mein Passwort hat? Es gibt viele Leute, die es sich zum Ziel gemacht haben, Passwörter von anderen Usern zu knacken um damit eine andere Identität vorzutäuschen. Das kann vor allem bei beispielsweise Online Banking gravierende Folgen haben. Zudem gibt es sehr viele Leute, die bei mehreren Applikation dasselbe Passwort verwenden. Wird es dann einmal gehackt, kann der Hacker Zugriff auf jede Applikation dieses Users erlangen. Um das zu vermeiden, sollte prinzipiell ein Passwort gewählt werden, dass aus **Groß- und Kleinbuchstaben**, **Zahlen**, sowie **Sonderzeichen** besteht und zusätzlich kein erratbares Wort darstellt und möglichst lange ist. Werden all diese Faktoren bei der Passworterstellung berücksichtigt, ist die Wahrscheinlichkeit, dass ein Hacker das Passwort bspw. durch Brute-Force Attacke errät sehr gering. [9]

Abgesehen von Wissen gibt es noch die Möglichkeit der Authentifizierung mittels **Besitz**. Dies wird häufig mit sogenannten **Smartcards** realisiert. Das sind kleine Geräte, die Informationen über einen User, beispielsweise public und private Keys, speichern können. Um diese Informationen dann zum Anmelden nutzen zu können, muss die Karte durch ein entsprechendes Lesegerät gezogen und mit einem PIN-Code bestätigt werden. Dadurch wird es Hackern schwer gemacht, da die Karte **physisch beim Besitzer** ist und nur mit der Karte eine Anmeldung durchgeführt werden kann. Ein Problem dabei kann natürlich sein, dass keine Anmeldung mehr stattfinden kann, wenn die Karte verloren geht. [9]

Eine weitere Möglichkeit zur Authentifizierung ist die **Biometrie**. Biometrische Daten, wie beispielsweise **Fingerabdrücke**, sind einzigartig und eignen sich dadurch sehr gut zur Identifizierung. Doch nicht nur Fingerabdrücke kommen hier infrage: **Stimme, Retina- und Irismuster** eignen sich ebenso gut als eindeutige Identifikation. Der große Vorteil bei der biometrischen Identifizierung ist die Einfachheit für den Anwender. Dieser muss sich kein Passwort mehr merken und nicht mehr darauf achten, immer eine Smartcard mitzunehmen. Die biometrischen Daten trägt jeder immer mit sich herum. Das große Problem dabei ist allerdings der Aufwand für diese Art von Identifizierung. Die Scanner, die zur Aufnahme von biometrischen Daten benötigt werden, sind nicht sehr billig. [9]

# 5. Desaster Recovery

## 5.1. 7 Stufen von Desaster Recovery (K3)

**Tier 0: No off-site data – Possibly no recovery**

„Unternehmen mit einer Tier 0-Lösung haben keinen Disaster Recovery Plan. Sie haben keine gespeicherten Informationen über das System, keine Dokumentation und auch keine Backups. Die Zeit, die benötigt wird um ein solches System wiederherzustellen ist unvorhersehbar, es kann sogar sein, dass es unmöglich ist zum Normalzustand zurückzukehren.“ [10]

**Tier 1: Data backup with no hot site**

„Tier 1-Systeme erhalten regelmäßig ein Backup, und lagern diese an einem sicheren Ort, der sich außerhalb des eigenen Hauses (der eigenen Infrastruktur) befindet. Dies ist notwendig, da vor einem Desaster auch Backups mit lokalem, nicht redundanten Speicherort nicht sicher sind. Diese

Methode Backups zu transportieren heißt PTAM, ausgeschrieben 'Pick-up Truck Access Method'. Sprich, das Backup ist so schnell greifbar, wie der Transport vom Aufbewahrungsort dauert. Abhängig davon, wie oft Sicherungen gemacht und versendet werden müssen Unternehmen einige Tage bzw. Wochen Datenverlust inkaufnehmen, dafür sind die Sicherungsdateien geschützt außerhalb des Geländes aufbewahrt." [10]

**Tier 2: Data backup with a hot site**

„Bei Verwendung von Tier 2 werden ebenso regelmäßige Sicherungen vorgenommen, auf langlebigen Speichermedien wie etwa Tapes. Das wird kombiniert mit eigener Infrastruktur außerhalb des eigenen Geländes (genannt 'Hot Side'), von welchen die Backups rückgelesen werden im Falle eines Desasters. Diese Lösung benötigt immer noch einige Stunden oder Tage zur Wiederherstellung, jedoch ist die Gesamtdauer besser vorhersehbar." [10]

**Tier 3: Electronic vaulting**

„Der Ansatz Tier 3 baut auf Tier 2 auf. Hinzufügend dazu werden kritische Daten elektronisch abgekapselt von den weniger prioren. Die Abgekapselten sind üblicherweise aktueller als die Daten, die via PTAM abgelegt werden. Als Ergebnis ist hier weniger Dateiverlust, sollte eine Katastrophe eintreten. Einrichtungen, die 'Electronic Remote Vaulting' bereitstellen, verfügen über Hochgeschwindigkeitsanbindungen und entweder physische oder virtuelle Sicherheitstape-Lesegeräte, mit automatisierter Sicherungsbibliothek beim entfernten Standort. Als praktischen Beispiel zur Implementierung dienen IBMs Peer-to-Peer TotalStorage Virtual Tape Server oder Oracles VMS Clustering." [10]

**Tier 4: Point-in-time copies**

„Tier 4-Lösungen werden oft von Unternehmen verwendet, die hohen Wert auf Datenkorrektheit und schneller Wiederherstellung legen als die unteren Stufen bereitstellen. Eher als das Auslagern von Speichertapes wie bei 0-3 gegeben, integriert diese Stufe Sicherungen auf Basis von Disks (also Festplatten). Immer noch sind mehrere Stunden Datenverlust möglich, jedoch ist es einfacher Point-in-Time-Backups (PiT, jeweils zu einem festgelegten Zeitpunkt) zu erstellen mit einer höheren Frequenz als Tape-Sicherungen, sogar wenn elektronisch gekapselt." [10]

**Tier 5: Transaction integrity**

„Tier 5 wird verwendet, wenn es zwingend erforderlich ist, dass Daten konsistent sind zwischen Produktivsystem und dem Wiederherstellungs-Remoteserver. Bei einem solchen Aufbau kommt es zu kaum bis gar keinem

Datenverlust, aber die Verfügbarkeit dieser Funktionalität ist stark abhängig von der verwendeten Implementierung.“ [10]

**Tier 6: Zero or near-zero data loss**
„Tier 6 hält das höchste Maß an Datenrichtigkeit aufrecht. Dieser Ansatz wird eingesetzt von Systemen, in denen wenig oder gar kein Datenverlust vertretbar ist, und wo Daten für den weiteren Gebrauch schnell wiederhergesellt werden müssen. Solche Lösungen haben keine Abhängigkeit von Anwendungen oder Mitarbeitern, um Datenkonsistenz zu gewährleisten. Tier 6 erfordert eine Form von Disk Mirroring zu einem Remoteserver, hierfür gibt es verschiedenste synchrone oder asynchrone Implementierungen von vielen Herstellern. Jede Herangehenweise ist leicht anders, mit verschiedenen Möglichkeiten und anderen Recovery Point/Recovery Time-Anforderungen. In den manchen Fällen wird jedoch auch eine Art von Tape-Speicherung benötigt, abhängig von dem erforderlichen Speicherplatz, der von Backup in Anspruch genommen wird.“ [10]

**Tier 7: Highly automated, business integrated solution**
„Das höchste Level nimmt all die Hauptkomponenten von Tier 6 zusammen und fügt die Intgration von Automation hinzu. Das erlaubt Tier 7 damit auch die Datenkonsistenz, die bei Tier 6 gegeben ist. Desaster werden automatisch erkannt von Geräten außerhalb des eigenen Computersystems. Außerdem wird die Wiederherstellung automatisch ausgeführt, was sozusagen den kompletten Wiederherstellungsprozess bei System und Anwendungen beschleunigt, und diese schneller und zuverlässiger laufen lässt als es überhaupt möglich wäre durch händische Prozeduren. Die Ausfälle belaufen sich auf wenige Minuten oder Sekunden.“ [10]

## 5.2. Desaster Recovery Plan (K3)

„Ein Disaster Recovery Plan muss Desasteridentifikation, Kommunikationsrichtlinien, das Koordinieren der Prozesse, etwaige Ausweichmöglichkeiten, Prozesse, um so schnel wie möglich wieder zum Normalzustand zurückzukehren und einen Feldtest des Plans sowie Wartungsroutinen beinhalten. Es muss kurz ein funktioneller Plan sein, der alle Prozessketten richtig adressiert, um die Systeme wiederherzustellen, inkl. eines Zuständigen zur stetigen Wartung des Plans. Es empfiehlt sich außerdem ein eigenes Disaster Response-Team auszuweisen, abhängig von den gegebenen Anforderungen. Wenn der Plan entworfen wird, ist es wichtig eine Priorisierung aufzustellen, welche Infrastruktur bzw. Systeme für den Erhalt

der Einsatzfähigkeit zwingend nötig sind. Ein erstes Maß ist die Wiederherstellung der voraussetzenden Umgebungen, etwa ein funktionierendes internes Netzwerk. Ein zweiter Punkt ist die auferlegte nötige Uptime für jedes System. Diejenigen, die eine 24/7-Uptime erreichen sollen sind den weniger prioren vorzuziehen. Es ist auch zu bedenken, dass bestimmte Workarounds effektiv laufen sollen, ohne noch größere Probleme zu verursachen. Der Ersteller des Plans muss möglichst viele Daten sammeln über alle verwendeten Systeme, sowie Abhängigkeiten untereinander abbilden und miteinander in Konflikt stehende, gleichrangige Priorisierungen klären. Ein DRP wird nicht automatisch ausgeführt, sondern händisch, angepasst an den Bedarfsfall.“ [10]

# 6. Algorithmen und Protokolle

## 6.1. Kommunikationsformate (K1)

Siehe Kapitel 1.1

## 6.2. Verschlüsselung (K3)

Siehe Kapitel 3.2

# 7. Konsistenz und Datenhaltung

## 7.1. Datenbanktypen (K4)

Bei der Persistierung von Daten wird prinzipiell zwischen zwei Konzepten von Datenbanken unterschieden: relationale Datenbanken und NoSQL Datenbanken. [11]

### 7.1.1. Relationale Datenbanken

Relationale Datenbanken folgen alle dem SQL-Standard. Der SQL-Standard definiert einheitliche Befehle zur Verwaltung der Daten in der Datenbank. Bei allen Datenbanken, die diesem Standard folgen, können somit dieselben Anweisungen zur Abfrage und Verwaltung der gespeicherten Daten verwendet werden.

Um Daten in einer relationalen Datenbank abspeichern zu können, muss vorher eine Datenstruktur erstellt werden. Diese Datenstruktur besteht aus

Tabellen, in die die Datensätze gruppiert werden. Jede Tabelle besteht wiederum aus einer oder mehreren Spalten, die festlegen, welche Daten in dieser Tabelle gespeichert werden können. Ein einzelner Datensatz ist dann eine Reihe in einer entsprechenden Tabelle. Sollte ein Attribut (eine Spalte) auf einen Wert in einer anderen Tabelle referenzieren, so kann ebenso eine Relation (eine Beziehung bzw. eine Verbindung) zwischen den beiden Tabellen definiert werden.

Die **Vorteile** bei dem relationalen Konzept liegen vor allem in der einheitlichen Abfragesprache SQL. Damit ist es sehr einfach, sich in eine andere relationale Datenbank einzufinden und diese schnell verwenden zu können. Zudem ist bei relationalen Datenbanken zumeist eine hohe Datenverlässlichkeit geboten. Der größte Nachteil daran ist die feste Datenstruktur, die bereits zu Beginn klar definiert werden muss und zu späterem Zeitpunkt nicht einfach verändert werden kann. [11]

### 7.1.2. NoSQL Datenbanken

NoSQL ist ein Konzept, das sich aufgrund von aufgetretenen Problemen bei relationalen Datenbanken entwickelt hat. NoSQL Datenbanken halten sich, im Gegensatz zu relationalen Datenbanken, nicht an strikt festgelegte Strukturen.

Bei NoSQL Datenbanken wird prinzipiell zwischen vier Arten unterschieden:

- **Key-Value**
  Verwendet Werte-Paare, die aus jeweils einem Schlüssel und einem dazu passenden Wert bestehen. Der Wert kann mithilfe des Schlüssels ausgelesen werden. Key-Value Datenbanken besitzen keine Struktur und keine Relationen zwischen Datensätzen. Sie werden hauptsächlich für eine schnelle Speicherung von simplen Datensätzen verwendet.
- **Spaltenorientiert**
  Spaltenorientierte Datenbanken bauen auf dem Prinzip von Key-Value Datenbanken auf. Sie gruppieren die Key-Value Paare. Eine dieser Gruppierungen enthält dann mehrere Key-Value Paare, die die sogenannten Spalten darstellen, wobei die Spalten, anders als bei relationalen Datenbanken, nicht zwingend dieselbe Struktur aufweisen müssen. Diese Art von Datenbank wird hauptsächlich für eine große Anzahl an zu speichernden Datensätzen verwendet, wenn eine einfache Key-Value Datenbank nicht mehr ausreicht.
- **Dokumentenbasiert**

Dokumentenbasierte Datenbanken haben prinzipiell einen ähnlichen Ansatz wie Spaltenorientierte. Jedoch erlauben sie eine deutlich tiefer gehende Verschachtelung von Key-Value Paaren und damit auch komplexere Strukturen.

- **Graph**
  Graphendatenbanken verwenden eine Baumartige Struktur, bestehend aus sogenannten Knoten und Kanten, die durch Relationen miteinander verbunden sind. Diese Art von Datenbank ist besonders dann sinnvoll, wenn klare Verbindungen von Daten gefragt sind.

Ein großer **Vorteil** von NoSQL Datenbanken ist vor allem die flexible Struktur. Die Datenstruktur muss nicht, wie bei relationalen Datenbanken, bereits zu Beginn klar definiert werden, sondern kann für jeden Datensatz individuell ausfallen. Dadurch ist zudem nicht nur eine vertikale, sondern auch eine horizontale Skalierung einfach möglich. Ein großer Nachteil ist indes die Unterschiedlichkeit in der Abfrage von Daten, da jede NoSQL Datenbank eine eigene Form der Abfrage besitzt. [11,12]

## 7.2. 2-Phase-Commit Protokoll (K4)

Das 2-Phase Commit Protokoll dient zur Konsistenzerhaltung bei Transaktionen, die über mehrere Maschinen verteilt stattfinden müssen, also bei verteilten Transaktionen. Dazu benötigt es einen Koordinator, der mit jedem der Teilnehmer kommuniziert und den verteilten Commit koordiniert. Um den verteilten Commit durchführen zu können, benötigt es zwei Phasen: eine Abstimmungsphase und eine Entscheidungsphase. Diese beiden Phasen umfassen jeweils zwei Schritte [13]:

- Abstimmungsphase
  1. Der Koordinator sendet an alle Teilnehmer ein VOTE_REQUEST.
  2. Jeder Teilnehmer sendet nach Empfangen des VOTE_REQUEST entweder ein VOTE_COMMIT, wenn die Transaktion lokal durchgeführt werden kann, oder ein VOTE_ABORT, wenn die Transaktion lokal nicht durchgeführt werden kann, an den Koordinator zurück.
- Entscheidungsphase
  1. Sobald alle Teilnehmer eine Nachricht an den Koordinator zurückgesendet haben, entscheidet dieser über den Commit. Wenn

alle Teilnehmer ein VOTE_COMMIT gesendet haben, schickt der Koordinator an alle Teilnehmer ein GLOBAL_COMMIT. Wenn mindestens einer der Teilnehmer jedoch ein VOTE_ABORT gesendet haben, so kann die Transaktion nicht durchgeführt werden und der Koordinator schickt an alle Teilnehmer ein GLOBAL_ABORT.

2. Jeder Teilnehmer wartet auf die entsprechende Antwort des Koordinators. Bei einem GLOBAL_COMMIT, wird die Transaktion bei jedem Teilnehmer einzeln lokal committed, bei einem GLOBAL_ABORT wird ein Rollback durchgeführt.

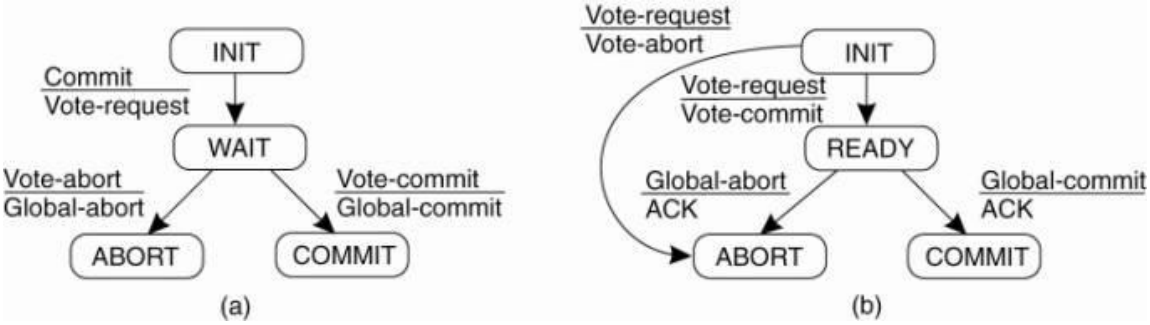


*Abbildung 1: Zustände im 2PC-Protokoll bei (a) dem Koordinator (b) einem Teilnehmer [13]*

Ein **Problem**, das hierbei allerdings auftreten kann, ist, wenn der Koordinator abstürzt. In diesem Fall ist es für die Teilnehmer eventuell nicht mehr möglich zu entscheiden, ob ein Commit oder Rollback durchgeführt werden soll (wenn im READY Zustand). Das bedeutet, dass die Teilnehmer solange warten müssen, bis der Koordinator wieder funktionsfähig ist und dabei keine anderen Aktionen ausgeführt werden können (Blocking). [13]

Um dieses Problem zu lösen, gibt es das **3-Phase Commit Protokoll**, das genau solche Zustände verhindern soll. Dazu ist es notwendig, dass die Zustände des Koordinators und der einzelnen Teilnehmer folgende zwei Bedingungen erfüllen [13]:

1. Es gibt keinen einzigen Zustand, von dem ein direkter Wechsel zu einem ABORT oder COMMIT Zustand möglich ist
2. Es gibt keinen einzigen Zustand, von dem aus es nicht möglich ist sich für entweder einen ABORT oder COMMIT Zustand zu entscheiden

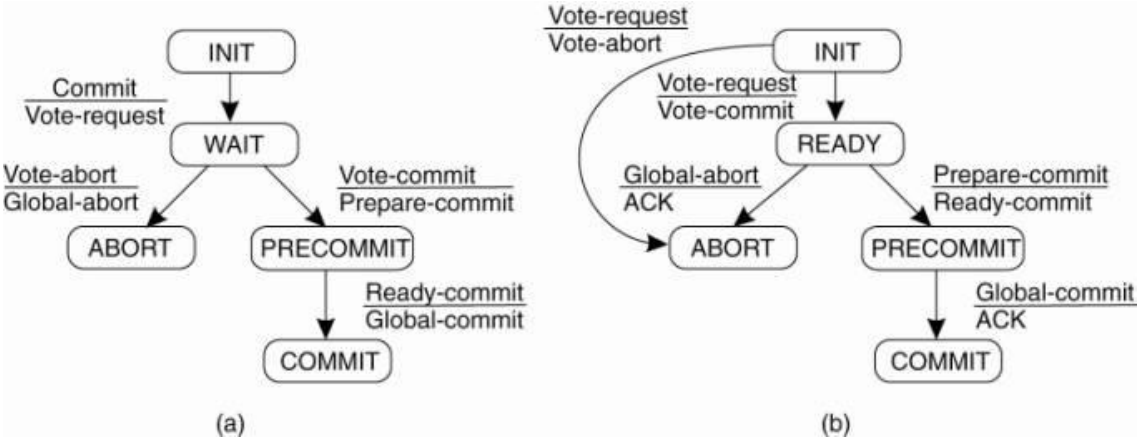


*Abbildung 2: Zustände im 3PC-Protokoll bei (a) dem Koordinator (b) einem Teilnehmer [13]*

Der **Unterschied** hier zum 2PC-Protokoll ist, dass wenn der Koordinator von jedem Teilnehmer ein VOTE_COMMIT erhalten hat und damit ein Commit durchgeführt werden kann, der Koordinator nicht direkt ein GLOBAL_COMMIT sondern stattdessen zunächst ein PREPARE_COMMIT an alle Teilnehmer sendet. Jeder Teilnehmer geht dann in den Zustand PRECOMMIT über und sendet anschließend ein READY_COMMIT an den Koordinator zurück. Sobald der Koordinator von jedem diese Nachricht erhalten hat, wird erst das GLOBAL_COMMIT ausgesendet und die Transaktion bei jedem Teilnehmer lokal committed. [13]

Sollte hier der Koordinator abstürzen, kann ein Teilnehmer durch kontaktieren eines anderen Teilnehmers dessen Zustand herausfinden und damit eine eindeutige Entscheidung treffen, ob ein COMMIT oder ABORT durchgeführt werden soll. [13]

# 8. Weiterführende Informationen

Zu Kapitel 3.1:

**Ausarbeitung Replikation**
Hagen Aad Fock, Stefan Polydor

Zu Kapitel 3.2 und Kapitel 6.2:

**Ausarbeitung Sicherheit**
Philipp Adler

Zu Kapitel 4:

**Ausarbeitung Autorisierung & Authentifizierung**
Andreas Ernhofer, Jakub Kopec

Zu Kapitel 3.1 und Kapitel 5:

**Ausarbeitung Synchronisierung & Konsistenz**
Erik Brändli, Michael Weinberger

Zu Kapitel 7.2:

**Ausarbeitung Transaktionen und Nebenläufigkeit**
Melanie Göbel, Tobias Perny

# 9. Quellen

[1] **XML Advantages & Disadvantages**
Selena Sol
http://www.theukwebdesigncompany.com/articles/xml-advantages-disadvantages.php
Zuletzt besucht am 29.05.2016

[2] **JSON: The Fat-Free Alternative to XML**
json.org
http://www.json.org/xml.html
Zuletzt besucht am 29.05.2016

[3] **What are the pros and cons of JSON?**
quora.com
https://www.quora.com/What-are-the-pros-and-cons-of-JSON
Zuletzt besucht am 29.05.2016

[4] **XML vs. JSON - A Primer**
Sean Lindo
http://www.programmableweb.com/news/xml-vs.-json-primer/how-to/2013/11/07
Zuletzt besucht am 27.05.2016

[5] **Kapitel 9 Replizierte Datenbanken**
Unbekannt
http://dbs.uni-leipzig.de/buecher/mrdbs/mrdbs-91.html
Zuletzt besucht am 27.05.2016

[6] **Verschlüsselungsverfahren**
Besim Karadeniz
http://www.netplanet.org/kryptografie/verfahren.shtml
Zuletzt besucht am 28.05.2016

[7] **Der RSA-Algorithmus**
Michael Busse, Matthias Schmitt, Jörg Steeg
http://www.zum.de/Faecher/Inf/RP/infschul/kr_rsa.html
Zuletzt besucht am 28.05.2016

[8] **Authentifizierung, Autorisierung und Abrechnung (AAA)**
Margaret Rouse
http://www.searchsecurity.de/definition/Authentifizierung-Autorisierung-und-Abrechnung-AAA
Zuletzt besucht am 31.05.2016

[9] **Understanding and selecting authentication methods**
Deb Shinder
http://www.techrepublic.com/article/understanding-and-selecting-authentication-methods/
Zuletzt besucht am 28.05.2016

[10] **Ausarbeitung Synchronisierung & Konsistenz**
Erik Brändli, Michael Weinberger

[11] **Understanding SQL And NoSQL Databases And Different Database Models**
O.S. Tezer
https://www.digitalocean.com/community/tutorials/understanding-sql-and-nosql-databases-and-different-database-models
Zuletzt besucht am 29.05.2016

[12] **A Comparison Of NoSQL Database Management Systems And Models**
O.S. Tezer
https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models
Zuletzt besucht am 29.05.2016

[13] **Distributed Systems: Principles and Paradigms**
Andrew S. Tanenbaum, Maarten Van Steen
Pearson
2. Aufl., erschienen 2006
ISBN 0-13-239227-5
https://vowi.fsinf.at/images/b/bc/TU_Wien-Verteilte_Systeme_VO_%28G%C3%B6schka%29_-_Tannenbaum-distributed_systems_principles_and_paradigms_2nd_edition.pdf
Zuletzt besucht am 30.05.2016