JUST DO IT
HÖHERE ABTEILUNG FÜR
INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Ausarbeitung

DezSys

Systemtechnik-Matura
5BHIT 2015/16

Michael Weinberger

Version 1.0
Begonnen am 21. April 2016
Beendet am 16. Mai 2016

Betreuer: Graf/Borko

Inhaltsverzeichnis

Kompetenzen für Dezentrale Systeme

- **Lastenverteilung auf Applikationsebene**
  'können Lastverteilung auf Applikationsebene realisieren'
- **Sicherheitskonzepte**
  'können Sicherheitskonzepte für verteilte, dezentrale Systeme entwickeln'
- **Durchführung von Transaktionen in verteilten Systemen**
  'können in dezentralen Systemen Transaktionen durchführen'
- **Programmiertechniken zur Realisierung von entfernten Prozeduren, Methoden und Objekten**
  'können Programmiertechniken in verteilten Systemen zur Realisierung von entfernten Prozeduren, Methoden und Objekten anwenden sowie webbasierte Dienste und Messaging-Dienste in solchen Systemen implementieren'

Projektumfeld: Smart Home-Hersteller Loxone

Loxone als Smart-Home-Systemhersteller möchte seine Produktpalette mit Hilfe eines Partners erweitern. Die Firma Festo will expandieren und geht auf das Angebot ein.
Die Idee ist mehr mobile Robotik ins tägliche Leben einzubringen (Staubsauger, Rasenmäher, Lasttransporter, etc.). Zur Überwachung werden energieautarke Sensoreinheiten benötigt (Funklösung, energiesparend). Auch weitere Sensoren sollen miteingebunden werden (Ambient-Assisted-Living, Pflege/Gesundheit, etc.). Die Steuerung und Datenerfassung soll mittels einer Cloudimplementierung erfolgen (IoT, App, etc.). Dabei ist die Sicherheit bez. die Datenweitergabe ein wichtiger Aspekt.

In dieser Ausarbeitung findet sich Theorie, die die Kompetenzen abdeckt sowie Praxis, die auf das Projektumfeld eingeht.

1 Cloud Computing und Internet of Things

Wie bereits der Aufgabenstellung zu entnehmen ist, soll hier eine Cloudimplementierung geschehen. Jeder Client, egal ob Rasenmäher oder Staubsauger, ist ständig mit dem Internet verbunden. Hierfür benötigt es eine möglichst hochverfügbare Internetanbindung seitens des Clients. Ohne Konnektivität können die angebotenen Services nicht gewährleistet werden. Über eine zentrale, firmeneigene Stelle außerhalb des Netzwerks werden alle Clients gesteuert, und deren Daten aufgenommen. Der große Vorteil einer Implementierung mithilfe der 'Cloud' ist die hohe Skalierbarkeit. Sollte Loxone weiter expandieren wollen, so kann innerhalb kürzester Zeit mehr Kapazität zur Verfügung gestellt werden. Auch die bessere Wartbarkeit ist ein positiver Aspekt. Da die Cloudinfrastruktur meist von externen, marktführenden Providern bereitgestellt wird, wird das System als hochverfügbar gelten. Ein Ausfall soll besonders hier nicht passieren, da sonst 'der ganze Betrieb steht' und hunderte, wenn nicht sogar tausende User in Mitleidenschaft zieht, die ihre Heimrobotik nicht mehr verwenden können. [1]

1.0.1 Was versteht man unter Cloud Computing?

Heutzutage setzen viele Hersteller auf Cloud Computing und bieten dementsprechende Plattformen an, der Trend geht immer mehr in diese Richtung.

Konkret geht es bei Cloud Computing um die Auslagerung von Anwendungen, Daten und Rechenvorgängen ins Web.

Diese Auslagerung bietet einige Vorteile. Die Synchronisation zwischen mehreren Rechnern wird nicht mehr relevant und gemeinsame Arbeit an Dokumenten durch die zentrale Ablage vereinfacht. Für das Absichern der Datensätze ist die Cloud Computing-Plattform und die Anwendung selbst verantwortlich. Natürlich ist auch hier der Datenspeicher begrenzt, jedoch in jeder Hinsicht größer als der eines einzelnen Rechners oder Festplattenverbundes im normalen Stil. Verglichen mit traditionellen Systemumgebungen sind Cloud Computing-Plattformen wesentlich einfacher zu verwalten. Der Grund dafür ist der hohe Abstraktionsgrad der Plattformen, denn um typische Administrationsaufgaben wie Load Balancing oder Serverwartung kümmert sich der Anbieter. Bei Rechenvorgängen ist der Vorteil einer besseren Skalierbarkeit gegeben, dass man auf einen großen Pool von Instanzen zurückgreifen kann. Dank Cloud Computing und dessen hoher Flexibilität kann man diese Server beispielsweise auch für wenige Stunden oder Tage, in denen sie benötigt werden, *on demand* mieten und somit Betriebskosten einsparen. Die Bereitstellung erfolgt innerhalb von Minuten, und kommt ohne komplexe Verträge aus.[1]

1.0.2 Wer bietet Cloud-Infrastruktur an?

Zwei der größten Anbieten im direkten Vergleich.

Amazon Web Services (AWS)

Amazon ist mit Abstand der Innovationsmotor im Cloud-Computing-Markt. Im Bereich Infrastructure-as-a-Service (IaaS) ist AWS der unangefochtene Marktführer. Es gibt eine Vielzahl von bekannten, weltweit operierenden Unternehmen, die hier auf AWS schwören. Zur Verbesserung der Hochverfügbarkeit haben sie Rechenzentrum rund um den Globus verteilt, um einen möglichst guten Ping zu erzielen. Die Weboberfläche ist intuitiv, und bedarf kaum Einarbeitungszeit. Das AWS-Portfolio umfasst über drei Dutzend verschiedene Web-Services. Für die Implementierung relevant sind die Komponenten zum skalierbaren Bereitstellen von Rechenkapazität (Elastic Compute Endgine), ein vollständig verwalteter NoSQL-DB-Service (Dynamo DB) genauso wie die allgemeine Bereitstellung von Speicherkapazität (Simple Storage Service).

Microsoft Azure

Microsoft Azure ist ebenso eine Public Cloud-Platform, vergleichlich mit den AWS. Die Unterscheidungen finden eher im Detail statt. Azure wird weitläufig aufgefasst als Anbieter von Platform as a Service (PaaS) und Infrastructure as a Service (IaaS). Microsoft unterscheidet zwischen 11 verschiedenen Produkttypen, die Loxone-Anwendung würde in den Bereich *Internet of Things (IoT)* fallen, da der Fokus der Services darauf liegt, die Daten von Sensoren und anderen Geräten zu erfassen, überwachen und zu analysieren. [2]

1.0.3 Was versteht man unter IoT?

Das Internet der Dinge (Internet of Things / IoT) definiert eine Vielzahl von Komponenten, die mit einer IP-Adresse ausgestattet sind und selbstständig kommunizieren können. Ein Ding im Internet der Dinge kann im Falle des Projektumfeldes ein Rasenmäher, Temperatursensor oder Staubsaugerroboter sein. Die Anfrage wird von Node zu Node weitergereicht, bis das Ziel erreicht ist, mit zunehmender Anzahl lernt das Netzwerk die effizientesten Routen dazu. Informationen werden untereinander oder mit einer zentralen Servereinheit (hier: Cloud) ausgetauscht. Bisher wurde das Internet der Dinge am häufigsten mit Maschine-zu-Maschine-Kommunikation etwa bei einer Fertigungsstraße in Verbindung gebracht. Die Kommunikation hierbei kann kabelgebunden oder wireless geschehen. Sind Produkte mit M2M-Kommunikation ausgestattet, werden sie häufig als *intelligent* oder *smart* bezeichnet. Durch die wachsende Anzahl an verbundenen Nodes erwartet man, dass es neue Bedenken im Bereich des Datenschutzes und der Datenweitergabe gibt. [3]

1.1 Realisierung von entfernten Prozeduren, Methoden, Objekten zur Interkommunikation

Interkommunikation ist bei einem Smart Home ein großes Thema. Im Bereich Internet of Things ist es ein entscheidender Faktor, dass einzelne Komponenten mit Anderen kommunizieren können, ebenso wie mit einer zentralen Serverstelle. Wie dies geschieht, wird hier beschrieben.

1.1.1 Interprozesskommunikation

Interprozesskommunikation (IPC) ist eine Sammlung von Interfaces, die nebenläufige, koordinierte Aktivitäten zwischen verschiedenen Prozessen erzielen. So kann ein Programm viele Benutzeranfragen gleichzeitig verarbeiten. Jedoch kann auch jede einzelne Useranfrage mehrere Prozesse erzeugen, die allesamt miteinander kommunzieren können müssen. IPC nimmt sich dieser Problematik an. Jede Umsetzungsmethode hat Vor- und Nachteile, so ist es üblich, dass mehrere Methoden gleichzeitig zum Einsatz kommen. [4]

1.1.2 Wie werden Informationen ausgetauscht?

In der Interprozesskommunikation empfiehlt es sich, ein 'Protokoll' zu definieren, sprich, wie Nachrichten untereinander weitergegeben werden. SOAP ist in standardisiertes Verpackungsprotokoll für Nachrichten. Die Spezifikation definiert einen XML-basierten Umschlag (genannt Envelope) für die zu übertragenden Informationen. Des Weiteren werden Regeln für die Umsetzung von anwendungs- und plattformspezifischen Datentypen in XML-Darstellung definiert. Der Nachrichtenaustausch über XML stellt eine flexible, plattformunabhängige Methode zur Interkommunikation dar. Eine Nachricht kann im Falle des Projektumfeldes etwa Temperaturdaten des Sensors sein, inklusive Informationen über den Datentyp. Da XML an keine bestimmte Programmiersprache oder ein bestimmtes Betriebssystem gebunden ist, können XML-Nachrichten in allen Umgebungen verwendet werden. Aufgrund der Einfachheit empfiehlt sich in Java jedoch der Einsatz sogenannter POJOs. Das sind schlicht und einfach 'plain old java objects' mit Methoden und Attributen. Dieses Objekt kann in verteilten Systemen serialisiert übertragen werden, ist aber natürlich nicht so offen & flexibel wie SOAP. [5]

1.1.3 RPC

Der Remote Procedure Call (RPC) ist ein Protokoll, das die Implementierung verteilter Anwendungen vereinfachen soll. Die Idee ist, dass die lokale Instanz eine entfernte Funktion eines auf einem anderen Rechner laufenden Programms nutzen kann, ohne sich um Sachen wien Netzwerkdetails kümmern zu müssen. Der RPC ist nach dem Client-Server-Modell aufgebaut, daher arbeitet ein Aufruf in den meisten Fällen synchron. Das bedeutet die lokale Instanz, kurz der Client, sendet eine Anforderung an die entfernte Instanz, kurzum den Server, und unterbricht seine Arbeit, bis er eine Antwort erhält. In Verbindung mit Threads ist auch eine asynchrone Realisierung eines entfernten Funktionsaufrufs möglich. Die Implementierung eines Programms, das RPC-Aufrufe verwendet, gestaltet sich recht einfach. In einem der Sprache C sehr ähnlichen Programmcode wird eine so genannte Stub-Routine verwendet, die als Platzhalter für den kompletten Code zur Realisierung des Netzwerkzugriffs dient. [6]

1.1.4 Java RMI

Bei RMI (Remote Method Invocation) liegt der Fokus voll und ganz auf Java, und setzt das Aufrufen entfernter Methoden objektorientiert um. Objekte, die sich auf unterschiedlichen Rechnern befinden, können mithilfe von RMI über Methodenaufrufe miteinander kommunizieren. Die Funktionsweise von RMI lässt sich als Abwandlung von RPC erklären. Die lokale Instanz, ein Client-Objekt, sendet dabei eine Nachricht an den Server, die entfernte Instanz, welche die aufzurufende Methode sowie die dafür benötigten Parameter enthält. Das Server-Objekt führt die entsprechende Methode bei sich aus und schickt das Ergebnis wieder zurück an den Client. Zur Realisierung dieses Ansatzes kapselt Java die Daten, die über das Netzwerk übermittelt werden sollen, auf der lokalen Instanz in sogenannten 'Stubs', wie eingangs auch kurz angesprochen. Etwaige Parameter müssen dafür zuerst in einem passenden Format zusammengefasst werden. Komplexer jedoch ist diese Aktion bei Objekten, beispielsweise bei Strings oder eigenen Klassen. Objektreferenzen sind ja im Grunde nichts anderes sind als Pointer, sprich Zeiger auf bestimmte lokale Speicherstellen, damit kann der Server natürlich nicht viel anfangen. In diesem Fall muss also das gesamte Objekt übermittelt werden, wozu Objektserialisierung eingesetzt wird. Das ist der gleiche Mechanismus, der mit dem auch Objekte beziehungsweise dessen Referenzen auf einer lokalen Festplatte speichert. Um das zu ermöglichen, müssen Objekte, die als Parameter für RMI-Aufrufe dienen, das Interface *Serializable* aus dem Package *java.io* implementieren. [7]

1.2 Grundlagen Messaging-Dienste

Message Oriented Middleware (oft auch kurz genannt MOM) ist die Grundlage für eine asynchrone Kommunikation zwischen Client & Server in einer verteilten Anwendung. Diese Form der Interkommunikation steht im Gegensatz dazu, wenn Client und Server synchron, also direkt und zeitgleich, miteinander in Verbindung stehen und sich damit in einem IoT-Umfeld, wo oft viele kleine Daten geschickt werden, blockieren können. MOM wird daher auch als eine lose Kopplung zwischen den Komponenten bezeichnet. Der MOM-Server übernimmt die Verwaltung der asynchronen Nachrichten in den meisten Fällen in Form einer Warteschlange. Durch diesen 'Zwischenstopp' auf dem zentralen Server kann der Empfänger die Nachrichten zu einem beliebigen Zeitpunkt aus dieser Warteschlange abholen. Die Hauptaufgabe eines MOM-Servers ist die unabhängige Vermittlung von Nachrichten. Damit steht ein MOM-Server klar im Vorteil gegenüber klassischen, sich blockierenden verteilten Systemen, die auf RPC basieren. Message Oriented Middleware bietet damit ein Konzept zur Interkommunikation unabhängig von Programmiersprachen und Plattformen. Es existieren APIs für verschiedene konkrete Programmiersprachen, im Falle Java das Java Message Service (JMS). Das Konzept der asynchronen Kommunikation, nochmal veranschaulicht durch den Einsatz einer Warteschlange (Queue):

- Der Sender stellt eine Nachricht in die Queue des Empfängers.
- Dabei wirken Sender und Empfänger immer unabhängig voneinander.
- Der Sender agiert weiter, ohne dass er Kenntnis vom Status seiner Nachricht hat.
- Der Empfänger holt die Nachricht zu einem beliebigen Zeitpunkt aus seiner Queue. [8]

2 Automatisierung, Regelung und Steuerung

Themengebiet wird ausgelassen (1 von 1)

3 Security, Safety, Availability

Im Allgemeinen kann man die vier folgenden wichtigen Schutzziele definieren:

- Vertraulichkeit – Schutz vor unautorisiertem Zugang zu Informationen.
- Integrität - Schutz vor unautorisierter unbemerkter Änderung von Informationen.
- Verfügbarkeit - Schutz vor unautorisiertem Beschlagnehmen von Informationen oder Ressourcen.
- Zurechenbarkeit – für Aktionen und Ereignisse Verantwortliche müssen ermittelbar sein. [9]

3.1 Grundlegende Sicherheitskonzepte

Im gegebenen Projektumfeld ist eine Cloud-Implementierung gefragt - Cloud-Systeme erfüllen grundlegende, wenn nicht sogar erweiterte Sicherheitskonzepte bereits von Anfang an. Welche Möglichkeiten es gibt, die Firma und deren sensiblen IoT-Daten noch besser vor fremden Zugriff zu schützen auch auf Cloud-Ebene, wird hier erklärt. Wichtig ist ebenso, dass verwendete IoT-Komponenten über ihre ID-Zertifikate identifiziert werden, welche sie zum Herstellungszeitpunkt bekommen haben. Wie die Kommunikation abgesichert werden sollte, ist einem späteren Punkt zu entnehmen. [10]

3.1.1 Intrusion Detection-Systeme

Ein IPS (Intrusion Prevention System) hat sich in der Vergangenheit als nicht wirksam gezeigt. Es bekämpft lediglich die grundlegenden Probleme, und kümmert sich nicht darum, was passiert, wenn der Angreifer Zugriff auf die Ressourcen erlangt hat. Heutzutage gibt es eine Vielzahl von verschiedenen Attacken auf Computersysteme, die Bedrohungen können oft sehr vielfältig sein. Ohne ein IDS hat man keine Möglichkeit herauszufinden, wie lange ein Eindringling unbemerkt blieb, wie und wann er seinen Angriff ausführte oder welcher Schaden dadurch entstand. Die Hauptziele eines IDS sind also:

- Benachrichtigung des Admins/Sicherheitsbeauftragten im Falle eines Angriffs oder das Ergreifen von aktiven Gegenmaßnahmen
- eine juristische Verwertbarkeit der gesammelten Daten (den Angriff betreffend)
- die Erkennung von Verlusten (bestes Beispiel: Daten!)
- der Schutz vor zukünftigen Angriffen durch die Auswertung der gesammelten Daten bei einem (simulierten) Angriff. [9]

Wie oben schon erwähnt ist das Ziel des Einsatzes von IDS ist eine frühzeitige Erkennung von Attacken, um den möglichen Schaden zu minimieren und den Angreifer zu identifizieren. Nachdem

getane Sicherheitsverletzungen durch die Analysekomponenten erkannt wurden, werden die Reaktionskomponenten des IDS veranlasst entsprechende Reaktionen durchzuführen. Es kann grundlegend zwischen passiven und aktiven Reaktionen unterschieden werden. Passive Reaktionen liefern lediglich Informationen an den Nutzer des Intrusion Detection Systems und überlassen diesem dann die Aufgabe, weitere Maßnahmen zu ergreifen. Aktive Reaktionen umfassen im Gegensatz das automatische oder halbautomatische Auslösen von direkten Aktionen gegen den Angriff. Darunter fällt das Blockieren von Netzwerkdiensten, die Benachrichtigung umgebender Systeme oder die Sammlung zusätzlicher Informationen.
Es gibt auch Unterscheidungen, ein Network-based IDS versucht den Paketverkehr im Netz aufzuzeichnen und zu analysieren, während Host-based IDS nur für einen Rechner operieren, beides mit Vor- und Nachteilen. [9]

3.1.2 Honey Pot-Systeme

Ein Honigtopf (engl. *honeypot*) ist eine Einrichtung, die einen Angreifer vom eigentlichen Ziel ablenken soll und ihn in einen Bereich hineinziehen soll, der ihn sonst nicht interessiert hätte. Ein Honeypot ist also konkret ein schlecht abgesicherter Server, der bestimmte Netzwerkdienste eines Rechnernetzes simuliert, jedoch aus Sicht des Betreibers 'nicht nötig' ist und keine richtigen Daten oder Dienste bereithält. Honeypots werden vorrangig dazu eingesetzt, um Informationen über das Angriffsmuster und das Angreiferverhalten zu erhalten. Erfolgt durch den Angreifer ein Zugriff auf so einen Honey Pot, werden alle damit verbundenen Aktionen protokolliert und ein Alarm ausgelöst. Das abgekapselte reale Netzwerk bleibt vom Angriff möglichst verschont, da es besser gesichert ist als der Honeypot. Die Idee hinter dem Einsatz von Honeypot-Systemen ist in einem Netzwerk einen oder am besten mehrere Honeypots zu installieren, die keine vom Anwender oder anderen Kommunikationspartnern benötigten Dienste bieten und so im Normalfall niemals angesprochen werden. Ein Angreifer, der das Netzwerk auf Sicherheitslücken untersucht, wird den schlecht gesicherten Honeypot als Angriffsziel bevorzugen, und so kaum Schaden anrichten. [9]

3.1.3 Application Firewall

Eine Application Firewall ist eine spezielle Art einer Firewall den Input, Output oder Zugriffe auf Systemdienste protokolliert und diese gegebenenfalls blockiert, falls ein Verstoß gegen die Firewall-Policy auftritt. Die Firewall ist dafür ausgerichtet, den ganzen Netzwerkverkehr (bis zum Application Layer) zu kontrollieren. Wieder gibt es die Unterscheidung Network-/Host-based. [9]

3.2 Dezentrale Systeme sicherer machen

An sich lässt sich diese Frage so beantworten: Mit Authentifikation, Autorisierung, Verschlüsselung und Session Management lässt sich der Sicherheitsgrad einer Anwendung allgemein erhöhen. Diese Themen werden genauer, auch in Bezug auf das Umfeld, in späteren Punkten besprochen.
Ein Framework, dass die genannten Punkte vereint ist etwa Apache Shiro. Hier wird eine umfassende API bereitgestellt. Der Fokus liegt hierbei natürlich auf Webanwendungen, aber auch auf mobilen Applikationen (App-Entwicklung). Vergleichlich ist etwa Spring Security, welches als Teil des bekannten Spring-Frameworks Features wie etwa Authentifikation/Autorisierung und Schutz vor den häufigsten Angriffen im Enterprise-Bereich. [11, 12]

4 Authentication, Authorization, Accounting

Damit eine sichere Kommunikation gewährleistet werden kann, ist eine Authentifizierung der kommunizierenden Parteien erforderlich. In vielen Fällen erfordert sie auch die Sicherstellung der Nachrichtenintegrität und unter Umständen auch der Vertraulichkeit. Da es in der Umfeld-Firma Loxone mehr > 1 Kunden mit n IoT-Komponenten, die diesem unterliegen, verwalten wird, muss es klare Zuweisungen geben, welchem User welche Sensoren, Roboter, etc. gehören. [13]

4.1 Beschreibung der Grundlagen

4.1.1 Authentisierung

Unter Authentisierung versteht man den Nachweis der eigenen Identität. Dabei kann es sich um die Identität einer Person (eines Benutzers, Clients) oder auch um die einer Programminstanz handeln. Wird dagegen eine angegebene Identität überprüft, so spricht man von Authentifizierung, welche anschließend erklärt wird. Auch hier kann es um die Identität eines Menschen oder eines Programms gehen. Ein Beispiel: Ein Anwender, der sich an einem Computer anmeldet, gibt dafür seinen Nutzernamen und zusätzliche, eindeutig unterscheidbare Informationen (typischerweise ein Passwort) an. Damit authentisiert er sich gegenüber dem Anmeldeprogramm und dieses authentifiziert infolgedessen den Anwender. [13]

Ich gebe vor, die Person xy zu sein.

4.1.2 Authentifizierung

Die eigene Identität zu belegen oder die Identität eines anderen zu überprüfen, stellt eine einfache Aufgabe dar. Auch wenn alle anderen Merkmale unbekannt sein sollten, so können sich zwei Menschen immer noch anhand eines gemeinsamen Geheimnisses (Shared Secret) gegenseitig ihre Identität nachweisen. Solch ein gemeinsames Geheimnis kann bei zwei Personen beispielsweise ein Losungs- bzw. in den meisten Fällen ein Passwort sein.
Authentifizierung und Nachrichtenintegrität sind aufeinander angewiesen. Um in der Folge die Integrität der Datennachrichten sicherzustellen, die nach der Authentifizierung ausgetauscht werden, ist es gängige Praxis, Kryptografie mit geheimen Schlüsseln zu verwenden, indem zufällige Schlüssel, sogenannte Sitzungsschlüssel generiert werden. Ein Sitzungsschlüssel ist ein gemeinsamer, geheimer Schlüssel, der aus Gründen der Integrität und möglicherweise auch der Vertraulichkeit zur Verschlüsselung von Nachrichten verwendet wird. Ein derartiger Schlüssel wird im Allgemeinen nur benutzt, solange es den Kanal gibt. Bei der Schließung des Kanals wird sein zugehöriger Schlüssel verworfen (bzw. auf sichere Art zerstört).
Die Authentifizierung kann wie genannt auf Grundlage eines gemeinsamen geheimen Schlüssels basieren, oder auch über den Ansatz eines KDC, eines Key Distribution Centers, in späteren Punkten genauer erklärt. Eines der Probleme bei der Verwendung eines gemeinsamen geheimen Schlüssels zur Authentifizierung ist die Skalierbarkeit. Bei vielen Hosts in einem verteilten System müssen diese eine unnötige Vielzahl an geheimen Schlüsseln nutzen. Eine gerade angesprochene Alternative ist ein zentralisierter Ansatz wie ein Key Distribution Center. Dieses KDC nutzt gemeinsam mit jedem der Hosts einen geheimen Schlüssel, aber die Hosts untereinander benötigen nicht mehr paarweise geheime Schlüssel, was klar ersichtlich eine Verbesserung darstellt.

Eine andere Art der Authentifizierung läuft über öffentliche Schlüssel. Hierbei brauchen die kommunizierenden Parteien keinen gemeinsamen Schlüssel zu kennen. Ein Benutzer erzeugt ein Schlüsselpaar, welches aus einem öffentlichen und einem privaten Schlüssel besteht. Der Öffentliche dient zur Verschlüsselung und der Private zur Entschlüsselung. [13]

Ich bestätige, dass die Person xy ist.

4.1.3 Autorisierung

Neben Authentisierung und Authentifizierung sind auch die Begriffe Autorisierung und Zugriffskontrolle wesentlich für jedes Sicherheitskonzept. Beim Vorgang der Autorisierung wird festgelegt, mit welchen Berechtigungen Benutzer auf Ressourcen im Netzwerk zugreifen dürfen. Netzwerkdienste, die diese Ressourcen anbieten, führen im Allgemeinen eine Zugriffskontrolle durch. Dabei prüfen sie, ob der zugreifende Benutzer autorisiert ist. Dementsprechend wird der Zugriff auf die Ressource erlaubt, verweigert oder nur eingeschränkt gewährt. Damit das funktioniert, muss ein Dienst wissen, welchem Anwender er auf welches Objekt welche Art von Zugriff erlauben kann. Welche Autorisierungsinformationen ein Dienst dafür benötigt und wo diese hinterlegt sind, hängt von dem betrachteten Dienst ab. [13]

In meinem System hat Person xy die erforderlichen Rechte, um auf den Bereich zugreifen zu können. (oder ggf. nicht)

4.2 Benutzerverwaltung mit LDAP

Lightweight Directory Access Protocol (LDAP) ist ein TCP/IP-basiertes Directory-Zugangsprotokoll, das sich im Internet und auch in lokalen LANs als Standardlösung für den Zugriff auf Netzwerk-Verzeichnisdienste für Datenbanken, E-Mails, Speicherbereiche und andere Ressourcen etabliert hat. Das LDAP-Protokoll unterstützt die für die Kommunikation erforderlichen Funktionen zwischen LDAP-Client und LDAP-Server. Dazu gehören die Anmeldung am Server, die Suchabfrage nach allen Informationen zu einem bestimmten Benutzer, die Modifikation der Daten wie beispielsweise die Änderung eines Passworts und die Replikation der Daten zwischen verschiedenen Directories. Zu den Authentifizierungs- und Kontrolloperationen gehören das Anmelden, Abfragen und das Abbrechen der Abfrage, zu den Abfrageoperationen die Suchabfrage, das Lesen und Vergleichen und zu den Update-Operationen das Hinzufügen, Löschen und Ändern der Eintragungen. LDAP setzt direkt auf TCP/IP auf und arbeitet auf Client-Server-Basis. Ein Verzeichnis in diesem Sinne ist mit einer Datenbank zu vergleichen, jedoch mit einem speziellen Aufbau. Die abgelegten Daten werden in den meisten Fällen um ein Vielfaches öfter gelesen als geschrieben, sprich das Protokoll ist optimiert auf schnelle Lesezugrife. [14]

Viele verschiedene Hersteller bieten ihre eigene Implementierung eines LDAP-Servers an, jeder mit Unterschieden in der Speicherung der Daten und der Zugrif darauf, sowie andere Funktionen, die je nach Ausrichtung variieren. Bekannte Namen sind hier etwa Microsoft Active Directory, Novell eDirectory sowie das freie OpenLDAP. OpenLDAP ist der bekannteste Open Source-LDAP-Server, und verfügt über eine lange Historie in der Unix-Welt und ist weitgehend plattformunabhängig.

Wie erwähnt ist LDAP ein Client-Server-Protokoll, ein oder mehrere (Failover zwecks Ausfallsicherheit) LDAP-Server beinhalten die Daten des LDAP-Verzeichnisbaums. Viele kleine bis mittelgroße Netzwerke verwenden eine solche Lösung um ihr zentrales Userverzeichnis aufzubauen. Ein Client verbindet sich mit einem LDAP-Server und stellt eine Anfrage, ob die eingegebenen Benutzerdaten valide sind, auch BIND Request genannt. Dieser BIND-Request wird übertragen, um den Autorisierungsstatus der Clientverbindung zu verändern oder die Anfrage an einen anderen LDAP-Server weiterzuleiten. [15]

4.3 Möglichkeiten zur Implementierung/alternative verteilte Authentifizierungsdienste

4.3.1 Kerberos

Bei Kerberos handelt es sich um eine sichere Methode, mit der sich Anfragen an ein Service im Netzwerk authentifizieren lassen. Kerberos stellt Nutzern ein verschlüsseltes *Ticket* zur Verfügung, mit dem sich Nutzer wiederum an einem bestimmten Service anmelden können. Der Vorteil besteht darin, dass das Passwort des Nutzers nicht über das Netzwerk geschickt werden muss. Im Rahmen einer sicheren Umgebung sollte daher eine Mischung aus Kerberos/LDAP bevorzugt werden, wobei Kerberos die Authentifizierung übernimmt. Eine kurze Beschreibung, wie Kerberos funktioniert: [16]

- Der User möchte auf einen Server zugreifen. Man weiß, dass der entsprechende Dienst ein Kerberos-Ticket benötigt.
- Um dieses Ticket zu erhalten, muss sich der User zunächst beim Authentication-Server (AS) authentifizieren. Der AS erstellt einen Session Key (zugleich ein sogenannten Verschlüsselungschlüssel), der auf dem Passwort und einer zufälligen Zeichenfolge (die den jeweils angefragten Service darstellt) basiert. Der Session Key ist im Grund ein 'Ticket für das Ticket'.
- Diesen Session Key wird anschließend an den Ticket-Granting-Server (TGS) weitergeschickt. Der TGS kann ein anderer Server als der AS sein – muss es aber nicht. In jedem Fall handelt es sich um einen anderen Service. Der TGS liefert anschließend das eigentliche Ticket, das wiederum an den ursprünglichen Service geschickt werden kann.
- Der jeweilige Service nimmt das Ticket an oder lehnt es ab. Bei einer Annahme kann der Nutzer auf den entsprechenden Dienst zugreifen.

- Das erhaltene Ticket ist mit einem Zeitstempel versehen. Solange die vorgegebene Nutzungdauer nicht abläuft, kann der Nutzer zusätzliche Anfragen über dieses Ticket stellen, ohne dass er sich erneut am AS und TGS authentifizieren muss. Je kürzer dieser Zeitraum ist, desto geringer ist die Chance, dass ein Ticket missbraucht werden kann. Kürzere Zeiträume bedeuten aber auch einen Mehraufwand beim Verlängern.

Der tatsächliche Ablauf ist deutlich komplizierter, je nach Implementierung kann das Vorgehen für den Nutzer abweichen. [16]

5 Disaster Recovery

Disaster Recovery (dt. auch Katastrophenwiederherstellung), im Folgenden auch DR genannt, beschreibt die Vorbereitung und Reaktion auf sogenannte Katastrophen, die abgespeicherte Daten und Lauffähigkeit eines IT-Systems betreffen. In diesem Bereich der Sicherheitsplanung ist mit negativen Ereignissen all das gemeint, was den Betrieb eines Unternehmens gefährdet. Hierzu gehören Cyberattacken, Infrastrukturausfälle ebenso wie Naturkatastrophen. DR umfasst beispielsweise Schritte zur Wiederherstellung von Server oder Mainframes mit Backups oder ferner die Bereitstellung von LANs für die unmittelbaren geschäftlichen Bedürfnisse.
eine Katastrophe kann vielerlei Ausmaß haben. Jede einzelne davon hat primäre und sekundäre Auswirkungen, die sich in direkte Schäden, korrumpierte oder unzugängliche Daten niederschlägt. Das eigene IT-Netzwerk ist verschiedensten Gefahren ausgesetzt, die in den schlimmsten Fällen auch ohne jegliche Vorwarnung auftreten können. Einige Beispiele:

- Feuer, Brand im Serverraum, Wasserrohrbruch
- Sonstige Naturkatastrophen
  Sind ebenso zu berücksichtigen, speziell bei hoher Sicherheitsstufe!
- Sicherheitsprobleme, Viren, Cyberattacken, Datendiebstahl
- Hardware- und Softwareausfälle
- Stromausfall, ...

Ein geeignetes Maß an Fehlertoleranz ist zu wählen, damit das System nicht beim kleinsten Problem ausfällt. Mit Fehlertoleranz bekommt der Administrator nichts vom Fehler mit, da er selbsständig ausgebessert wird. [17]

5.1 Sicherheitslevels & Backup-Strategien

Die SHARE-Gruppe hat im Zuge der Definition des 'Traditional Disaster Recovery' sieben verschiedene Stufen ausgewiesen. [17]

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

Unternehmen mit einer Tier 0-Lösung haben keinen Disaster Recovery Plan. Es gibt keine Backups, auch keine Informationen über das System und keine Dokumentation. Die Zeit, die benötigt wird um ein solches System wiederherzustellen ist unvorhersehbar, es kann sogar sein, dass es unmöglich ist zum Normalzustand zurückzukehren. [17]

5.1.2 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 befindet. Diese Methode Backups zu transportieren heißt PTAM, ausgeschrieben 'Pick-up Truck Access Method'. Einige Tage bzw. Wochen Datenverlust müssen befürchtet werden, dafür sind die Sicherungsdateien geschützt außerhalb des Geländes aufbewahrt. [17]

5.1.3 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'). Diese Lösung benötigt immer noch einige Stunden oder Tage zur Wiederherstellung, jedoch ist die Gesamtdauer besser vorhersehbar. [17]

5.1.4 Tier 3: Electronic vaulting

Der Ansatz Tier 3 baut auf Tier 2 auf. Hinzufügend dazu werden kritische Daten elektronisch abgekapselt von den weniger prioren. Als Ergebnis ist hier weniger Dateiverlust, sollte eine Katastrophe eintreten. [17]

5.1.5 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 Festplatten. Immer noch sind mehrere Stunden Datenverlust möglich. [17]

5.1.6 Tier 5: Transaction integrity

Tier 5 wird verwendet, wenn es zwingend erforderlich ist, dass Daten konsistent sind zwischen Produktivsystem und dem Wiederherstellungs-Remoteserver. [17]

5.1.7 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. Tier 6 erfordert eine Form von Disk Mirroring zu einem Remoteserver. [17]

5.1.8 Tier 7: Highly automated, business integrated solution

Das höchste Level nimmt all die Hauptkomponenten von Tier 6 zusammen und fügt Automation hinzu. Desaster werden automatisch erkannt von Geräten außerhalb des eigenen Computersystems. Außerdem wird die Wiederherstellung automatisch ausgeführt, was sozusagen den kompletten Wiederherstellungsprozess beschleunigt. Die Ausfälle belaufen sich auf wenige Minuten oder Sekunden. [17]

5.2 Disaster Recovery Plan

Ein Disaster Recovery Plan, im Folgenden auch DRP genannt, dokumentiert konkret Richtlinien, Verfahren und Maßnahmen, um die Störung eines Unternehmens im Falle eines Desasters zu begrenzen, und möglichst innerhalb eines bestimmten Zeitrahmens wieder zurück zum Normalzustand überzugehen. Wie bei einer Katastrophe macht das Ereignis die Fortführung des normalen Geschäftsbetriebs unmöglich. Genannter Plan sollte ein Teil eines jeden Standard- Projektmanagementsprozess sein. Ein Disaster Recovery Plan muss Desasteridentifikation, Kommunikationsrichtlinien, das Koordinieren der Prozesse, etwaige Ausweichmöglichkeiten, Prozesse, um so schnell 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. [17]

5.3 Best Practice für die vorgegebenen Anforderungen

Wenn Systeme 24/7 verfügbar sein müssen, wird wie hier eine Clusterlösung herangezogen. Hier unterscheidet man zwischen zwei Varianten, Failover-Clustering und 'echtem' Clustering. Bei der Failover-Technologie mit zwei oder mehreren Netzwerkdiensten übernimmt ein Zweitsystem bei einseitigem Ausfall des Primärsystems. Echtes Clustering, sprich ein Aktiv/Aktiv-Cluster, wird bezeichnet einen Rechnerverbund, in dem mehrere (meistens > 2) Nodes gleichzeitig aktiv sind. Da sie '100% der Zeit' verfügbar sind (aus Sicht des Kunden), gibt es per se keine Katastrophen, die die Business Continuity beeinträchtigen. Der einzige Nachteil: Solche Lösungen sind meistens sehr teuer in Aufbau und Wartung, und daher eher nur von großen Unternehmen mit großem Budget zu bewerkstelligen.
Für Systeme, die keine 100%-ige Verfügbarkeit benötigen, also nicht den hier gegebenen Anforderungen entsprechen, oder das Budget es nicht anders vorsieht, ist DR die bessere Wahl. Eine typische Lösung ist es ein identes System aufzubauen, es für etwaige Einsatzfälle zu warten und es als Failover zu verwenden. Der Wechsel auf dieses System verlangt einen Eingriff des Administrators, während dieser Zeit bis zur Inbetriebnahme 'steht der Betrieb'. Dies kann auch über eine Heartbeat-Anfrage geschehen, die überprüft, ob Server verfügbar sind. Die Rückkehr auf den Normalzustand kann mithilfe dieser Methode relativ schnell wiedererlangt werden, und generiert weniger Kosten als eine vollständige Cluster-Lösung. Die billigste Variante (aus Hardware-Sicht) ist es auf Fehler zu reagieren, nachdem sie passiert sind. Eine USV (Unterbrechungsfreie Stromversorgung) oder ein RAID-System (Redundant Array Of Independent Disks) sind jedoch relativ kostengünstige Maßnahmen, um ein System vor Katastrophen zu schützen.

Zur weiteren Definition der verschiedenen Failover: [17]

- Active/Active:
  Maximale Ausfallsicherheit, Cluster
- Hot Site:
  Sogenannter 'Failover', gleicher Datenstand auf allen Servern, erfordert Synchronisation.
- Warm Site:
  Ebenso ein Failover, Synchronisierung jedoch nur nach einiger Zeit in regelmäßigen Abständen, was Datenverlust von z.B. einem Tag mit sich bringt.
- Cold Site: Ein Failover, der händisch gestartet werden muss, Neuheitsgrad der Daten kann auch länger zurückliegen. Nur als Notfallplan! [17]

6 Algorithmen und Protokolle

6.1 Techniken zur Prüfung und Erhöhung der Sicherheit von dezentralen Systemen

Im Rahmen des Projektumfelds gibt es gewisse Risiken, und eine davon ist das Verändern von ungesicherten Nachrichten, die innerhalb des Netzwerks durch unbefugten Zugriff abgefangen & manipuliert werden. Hier greift eindeutig der Ansatz, die Kommunikation zu verschlüsseln. [10]

6.1.1 Verschlüsselung

Um nicht die eigentliche Nachricht im abfangbaren Klartext zu übertragen, wendet man einen Schlüssel an, der aus der Nachricht einen sogenannten Chiffretext generiert. Der Empfänger dieses codierten Textes besitzt einen Dechiffrierschlüssel, um die Nachricht in ihren Ausgangszustand zurück zu verwandeln. Der Standard entspricht der symmetrische Verschlüsselung, wo das gleiche Geheimzweichen für das Ver- und Entschlüsseln benutzt wird. Das Problem beim Austausch des symmetrischen Schlüssel ist, dass andere Teilnehmer dieses abfangen können, und daraufhin den Chiffretext ohne Probleme ebenso entschlüsseln können. Für dieses Problem gibt es eine Lösung, die sogennannte asymmetrische Verschlüsselung. Hier gibt es für das En- und Decoding einen eigene Schlüssel. Wobei der Verschlüsselungskey für jeden zugänglich ist aber nur der, der den Entschlüsselungkey besitzt, ist in der Lage, die Nachricht zu entschlüsseln. [9]

6.1.2 Symmetrische Verschlüsselung

Die Verschlüsselungsverfahren, die mit *einem* geheimen Schlüssel arbeiten, der zum Ver- und Entschlüsseln dient, nennt man symmetrische Verfahren oder Secret Key-Verfahren. Fast alle symmetrischen Verfahren sind auf ressourcenschonende Umgebungen optimiert. Sie zeichnen sich durch geringe Hardwareanforderungen, geringen Energieverbrauch und einfache Implementierung aus.
Dieser eine Schlüssel in der symmetrischen Kryptografie muss bei der Ver- und Entschlüsselung vorhanden sein. Diese Verfahren sind schnell und bei entsprechend langen Schlüsseln bieten sie auch eine relativ hohe Sicherheit. Der schwierigste Aspekt liegt in der Schlüsselübergabe zwischen den beiden Kommunikationspartnern. Vor der sicheren Datenübertragung mit Verschlüsselung müssen sich die Kommunikationspartner auf den Schlüssel einigen und austauschen. Wenn der Schlüssel den selben Kommunikationspfad nimmt, wie die anschließend verschlüsselten Daten, dann besteht die Gefahr, dass ein Angreifer in Besitz des Schlüssels gelangt, wenn er die Kommunikation abhört. Wenn der Angreifer den Schlüssel hat, dann kann er nicht nur die Daten entschlüsseln, sondern auch selber Daten verschlüsseln, ohne das es die Kommunikationspartner bemerken. Knackpunkt ist der unsichere Schlüsselaustausch und die Authentifizierung. Sicher ist die Schlüsselübergabe nur dann, wenn sich zwei Personen persönlich treffen und den Schlüssel austauschen oder der Schlüssel einen anderen Weg nimmt.
Advanced Encryption Standard, kurz AES, ist eine symmetrische Block-Chiffre mit flexibler Block- und Schlüssellänge. AES besitzt eine Standardmäßige Blocklänge von 128 Bit und Schlüssellängen von 128 Bit, 192 Bit und 256 Bit. Wieviele Runden absolviert werden hängt von der Schlüssellänge ab. Derzeitiger Standard 10 Runden bei einer Schlüssellänge von 128 Bit, 12 Runden bei 192 Bit und 14 Runden bei 256 Bit. [9]

6.1.3 Asymmetrische Verschlüsselung

In der asymmetrischen Kryptografie arbeitet man nicht mit einem einzigen Schlüssel, sondern mit einem Schlüsselpaar. Bestehend aus einem öffentlichen und einem privaten Schlüssel. Man bezeichnet diese Verfahren als asymmetrische Verfahren oder Public-Key-Verfahren. Üblich sind auch die Bezeichnungen Public Key-Kryptografie und Public Key-Verschlüsselung.
Ein fundamentales Problem der Kryptografie ist, dass sich die Kommunikationspartner auf einen gemeinsamen Schlüssel verständigen müssen. Man bezeichnet das als Schlüsselaustauschproblem. Asymmetrische Verschlüsselungsverfahren arbeiten mit Schlüsselpaaren. Ein Schlüssel ist der öffentliche Schlüssel (Public Key), der andere ist der private Schlüssel (Private Key). Dieses Schlüsselpaar hängt über einen mathematischen Algorithmus eng zusammen. Daten, die mit dem öffentlichen Schlüssel verschlüsselt werden, können nur mit dem privaten Schlüssel entschlüsselt werden. Deshalb muss der private Schlüssel vom Besitzer des Schlüsselpaares geheim gehalten werden. Der konkrete Anwendungsfall sieht so aus: Will der Sender Daten verschlüsselt an den Empfänger senden, benötigt er den öffentlichen Schlüssel des Empfängers. Mit dem öffentlichen Schlüssel können die Daten verschlüsselt, aber nicht mehr entschlüsselt werden (Einwegfunktion). Nur noch der Besitzer des privaten Schlüssels, also der richtige Empfänger kann die Daten entschlüsseln. Wichtig bei diesem Verfahren ist, dass der private Schlüssel vom Schlüsselbesitzer absolut geheim gehalten wird. Kommt eine fremde Person an den privaten Schlüssel muss sich der Schlüsselbesitzer ein neues Schlüsselpaar besorgen.
An RSA kommt man im Zusammenhang mit asymmetrischen Verfahren einfach nicht vorbei. Im Vergleich zum Diffie-Hellman-Schlüsselaustausch eignet sich RSA auch für die Verschlüsselung und als Signaturverfahren. Der RSA-Algorithmus (Ron Rivest, Adi Shamir und Leonard Adleman) basiert auf dem Faktorisierungsproblem. Asymmetrische Verfahren benötigen viel mehr Rechenleistung als symmetrische Verfahren. Wenn man RSA und AES miteinander vergleicht, dann ist RSA ungefähr um den Faktor 1.000 langsamer als AES. [9]

6.1.4 SSL/TLS-Protokoll

SSL ist ein Protokoll, das der Authentifizierung und Verschlüsselung von Internetverbindungen dient. SSL schiebt sich zwischen die Anwendungsprotokolle und den Transportprotokollen. Ein typisches Beispiel für den Einsatz von SSL ist der gesicherte Abruf von vertraulichen Daten über HTTP und die gesicherte Übermittlung von vertraulichen Daten an den HTTP-Server. In der Regel geht es darum, die Echtheit des kontaktierten Servers durch ein Zertifikat zu garantieren und die Verbindung zwischen Client und Server zu verschlüsseln. SSL ist äußerst beliebt und das Standard-Protokoll bzw. die Erweiterung für Anwendungsprotokolle, die keine Verschlüsselung für sichere Verbindungen mitbringen. Das ursprüngliche SSL ist inzwischen veraltet und in TLS aufgegangen. Obwohl man heute in der Regel TLS verwendet spricht man trotzdem immer noch von SSL.
Es ist jedoch ein Fakt, dass, wie das Projektumfeld vorgibt, das Internet der Dinge vorrangig auf unzuverlässige Datenübertragung via UDP aufsetzt, um möglichst schnell und in kurzen Abständen Daten liefern zu können ohne möglichst Rechenaufwand bzw. um Overhead zu sparen. Die Nachrichten sollen jedoch trotzdem sicher ankommen, hierfür gibt es DTLS, ein Ansatz, der TLS über UDP implementiert. Hier wird dafür gesorgt, dass alle Datenpakete sicher und integer ankommen, großer Overhead wird hier vermieden. [18, 19]

6.2 Grundlagen Lastverteilung

Load Balancing ist kein neues Konzept im Server- und Netzwerkbereich, und besonders wie hier im Firmenbereich mit vielen unterschiedlichen, gleichzeitigen Usern eine wichtige Technologie. Bei Server Load Balancing wird der Netzwerkverkehr über verschiedene Server Ressourcen verteilt. Anfangs wurden Load Balancer als reine Lastenverteiler verwendet, doch heute sind Load Balancer schon so weit entwickelt, dass sie zusätzliche Funktionen wie Healthchecks, Optimierung von Datenflüssen, etc. zur Verfügung stellen.
Im Webhostingbereich wird Load Balancing typischerweise für die Verteilung von HHTP-Verkehr auf mehrere Server (Nodes) eingesetzt. Durch die Verteilung des Dienstes auf mehrere Server schützt man sich zusätzlich vor Hardwareausfall, da bei Ausfall eines Servers der Netzwerkverkehr einfach über Andere erfolgen kann. Das Hauptziel ist es, den Netzwerkverkehr bei hoher Nachfrage auf alle vorhandenen Server aufzuteilen, um die bestmögliche Performance zu gewährleisten. Der User weiß normalerweise nichts über vorhandene Backend- bzw. Backupserver, für ihn scheint es so als ob nur ein einziger Server hinter einem Dienst steht. Durch das Verwenden von mehreren Servern für so eine Website entstehen aber Herausforderungen in den Bereichen Skalierbarkeit, Verwaltbarkeit und Verfügbarkeit. Load Balancing löst zusätzlich viele dieser Probleme. Durch die bessere Skalierbarkeit können laufend weitere Instanzen hinzugefügt werden, sofern sie benötigt werden. Der Load Balancer delegiert dann je nach Schema die Anfragen einfach auch auf den neuen Server. Die bessere Verwaltbarkeit zeichnet sich dadurch aus, dass einzelne Komponenten offline genommen werden können, ohne dass das System komplett offline gehen muss. Sämtliche Anfragen werden auf die anderen Server weitergeleitet. Wie bereits argumentiert resultiert dass allgemein in einer besseren Verfügbarkeit.
Nach dieser kurzen Einführung mit Webservern wird klar, wieso es auch auf Applikationsebene einer Lastenverteilung bedarf. Das Schema und die Vorgehensweisen bleiben gleich, das genannte Beispiel kann 1:1 auf eine beliebige verteilte Anwendung umgemünzt werden. [20]

6.3 Load Balancing-Algorithmen

Zur Verwirklichung, wie Anfragen verteilt werden, gibt es unterschiedliche Schemen.

6.3.1 Round Robin

Round-Robin ist eines der einfachsten Verfahren zur Lastenverteilung. Ein Load Balancer weist jedem Server der Reihe nach eine Verbindung zu. Dieses Verfahren kann eine gleichmäßige Verteilung der Last nicht sicherstellen, da jede Verbindung über unterschiedliche lange Zeiträume bestehen kann. Infolgedessen können manche Server mehr gleichzeitig aktive Verbindungen haben, als andere. Da Round-Robin eine sehr einfache Methode zur Lastverteilung ist, werden sehr wenige Ressourcen seitens des Load Balancers benötigt. Als Ergebnis davon ist der Algorithmus nur dann sehr effektiv, wenn der Lastverteilungsalgorithmus viel Rechenzeit benötigt. [20]

6.3.2 Least Connections

Jede neue Anfrage wird dem Server mit den geringsten gleichzeitig aktiv vorhandenen Verbindungen zugesandt. Der Load Balancer muss hierbei die Anzahl dieser Verbindungen jedes Servers zu

jeder Zeit festhalten. Diese Methode ist eine der effektivsten und öftesten verwendet in verschiedenen Anwendungsbereichen, wie beispielsweise DNS oder dem Web. Der Hauptgrund hierfür sind die einfache Verständlichkeit und Anwendbarkeit der Methode. Least Connections kann dann von Nutzen sein, wenn zwei oder mehr Server mit gleicher Ausstattung unterschiedlich stark belastet werden. [20]

6.3.3 Weighted Distribution

Da unterschiedliche Server von der Hardware her unterschiedliche ausgestattet sein können, kann mithilfe dieser Methode eine Angabe der Leistung der einzelnen Server in Relation zueinander erfolgen, indem jedem Server eine gewisse Gewichtung zugewiesen wird. In der Praxis wird diese Methode in Kombination mit anderen Methoden, wie Least Connections oder Round-Robin angewandt. Sollen nun weitere Server hinzugefügt werden im Fall von Weighted Round Robin die Anfragen wie gewohnt der Reihe nach gewichtet verteilt. Im Fall von Weighted Least Connections ist lediglich auf die Anzahl der aktuell verbundenen Clients und die Gewichtung zu achten. Diese Methode ist besonders dann geeignet, wenn bereits bestehende, womöglich leistungsschwächere Hardware mit leistungsstärkerer kombiniert werden soll. [20]

6.3.4 Response Time

In der Annahme, dass eine schnelle Reaktion eines Servers eine gute Performance zur Folge hat, wird die Reaktionszeit eines Servers vom Load Balancer gemessen und anhand dieser dann ein Server ausgewählt. Für eine effiziente Lastenverteilung muss diese Antwortzeit über einen längeren Zeitraum gemessen werden, was eine kurze Verzögerung mit sich bringt. Aufgrund des hohen Komplexitätgrades dieser Methode, könnte diese alleine nicht die beste Möglichkeit zur Lastenverteilung bieten. Der Algorithmus funktioniert hingegen in Kombination mit anderen Load Balancing-Methoden sehr annehmbar. [20]

6.3.5 Server Probe

Auf jedem Server rennt im Hintergrund ein Programm, welches die aktuelle Auslastung des Servers an den Load Balancer in regelmäßigen Zeitabständen weiterleitet. Auf diese Weise hat der Load Balancer stets Zugriff auf Daten, wie die aktuelle Auslastung der CPU, aber auch den verfügbaren Arbeits- und Festplattenspeicher. [20]

6.3.6 Kombiniert

Die Kombination von zwei oder mehreren Methoden kann eine besseres Erfassen der Serverlast ermöglichen. Hierzu können beispielsweise die Methoden Response Time und Least Connections zusammen verwendet werden. Die Verfahren können vom Load Balancer wieder mit entsprechender Gewichtung verwendet werden. [20]

6.3.7 Zufällig

Der denkbar einfachste Alogrithmus ist, dass der zu verwendende Server von einem Zufallszahlengenerator am Load-Balancer festgelegt wird. Sollten nun innerhalb eines kurzen Zeitraumes viele Anfragen zustande kommen, so werden diese zufällig verteilt, was in etwa eine grobe gleichmäßige Verteilung der Anfragen gewährleistet. [20]

6.4 Session-basiertes Load Balancing

Hier wird der Ansatz verfolgt, dass bei mehreren Anfragen vom selben User der bereits zugewiesene Server weiter verwendet wird. Dies spart einen erneuten Rechenaufwand, um die Anfrage einem neuen Server zuzuweisen, kann jedoch darin resultieren, dass aufgrund der 'sticky session' zu viele Anfragen auf einmal zugeteilt werden, sofern der Zeitraum, wie lange eine Session maximal dauern darf zu lange eingestellt ist. [20]

6.5 Load-Balancing-Frameworks

6.5.1 Apache Hadoop

Apache Hadoop ist ein Framework, dass das verteilte Verarbeiten von großen Datenmengen in vielen Clustern bereitstellt. Das gut skalierbare Design erlaubt es, dass sowohl ein paar wenige, aber auch mehrere 1000 Server mühelos verwendet werden können. Das Framework selbst wurde so entworfen, dass Fehler auf Applikationsebene entdeckt und beseitigt werden können, was infolgedessen eine hohe Verfügbarkeit des Services im Cluster sicherstellt. Hadoop verfügt über verschiedene Module, die nach Wahl hinzugefügt werden können, etwa das Hadoop Distributed File System oder etwa MapReduce. [20]

7 Konsistenz und Datenhaltung

Um bei der Firma Loxone die Smart Homes verlässlich steuern lassen zu können, sind Inkonsistenzen bei den gesammelten Daten tabu, diese wirken sich direkt auf den Kunden aus. Wie man diesem Problem mit Transaktionen entgegenwirken kann, ist hier beschrieben.

7.1 Grundlagen & Erklärung

7.1.1 CAP-Theorem

Das CAP-Theorem beschreibt das Verhältnis zwischen Konsistenz, Verfügbarkeit und Partitionstoleranz indem es besagt, dass nur davon zwei in einem verteilten System gleichzeitig angeboten werden können.
Consistency zählt zu den Sicherheitseigenschaften und beschreibt zusammengefasst, dass eine Anfrage auch eine integere Antwort erhält. Als Beispiel für Consistency können aus dem Umfeld zwei Staubsaugerroboter genommen werden. Wenn zwei Roboter gleichzeitig eine Stelle am Boden bearbeiten wollen, dürfen nicht beide die Antwort bekommen, dass der Weg frei ist und somit losfahren darf.
Availability: In verteilten Systemen muss für eine kontinuierliche Verfügbarkeit, jede Anfrage, die ein nicht fehlerhaften Server erhält, eine Antwort bekommen.
Partition Tolerance: Wenn man jedoch die Daten und Logik auf mehrere Server verteilt besteht die Gefahr von *partition forming*. Dies passiert, wenn sie in verschiedenen Gruppen aufgeteilt werden und wegen eines Fehlers nicht mehr miteinander kommunizieren können. Kein Fehler, außer ein Totalausfall, soll eine Antwort verfälschen. Wenn man somit eine Partition im Netzwerk hat, verliert man entweder Consistency, weil man Änderungen auf beiden Partitionen erlaubt oder Availability, weil man einen Fehler entdeckt und das System offline gehen muss, um den Fehler zu beheben. [21]

7.2 Verschiedene Transaktionsprotokolle

7.2.1 2-Phase-Commit

Der Zwei-Phasen-Commit ist eine Methode zum Koordinieren und Sicherstellen einer Transaktion zwischen mindestens zwei Ressourcenmanagern. Es gewährleistet Datenintegrität bei der Sicherstellung, dass entweder eine Transaktion von allen Ressourcemanager committed werden oder bei allen ein Rollback stattfindet. Die 1. Phase des Zwei-Phasen-Commit wird Abstimmungsphase oder Vorbereitungsphase genannt, die 2. Phase Entscheidungsphase oder Commit/Abort-Phase. [21]

7.2.2 3-Phase-Commit

Der Drei-Phasen-Commit verhindert das große Problem des Zwei-Phasen-Commit, dass Teilnehmer nach einem Absturz des Koordinators teilweise nicht mehr zu einer eindeutigen Entscheidung kommen können. In der Praxis wird der 3PC nicht häufig eingesetzt, da der Zustand bei dem 2PC nicht sehr oft vorkommt. Beim 3PC genügen die Zustände des Koordinators und der Teilnehmer folgender Bedingungen:

- Keinem einzelnen Zustand ist es möglich direkt in COMMIT/ABORT-Zustand zu gelangen.
- Keinem einzelnen Zustand ist es unmöglich eine endgültige Entscheidung zu treffen (Übergang in den COMMIT-Zustand) [21]

7.2.3 2-Phase-Lock

Das Zwei-Phasen-Sperrprotokoll ist eine Methode zur Synchronisierung von Zugriff auf aufgeteilte Daten. Bei einer Schreibsperre kann nur der sperrende Prozess auf das Objekt zugreifen und somit dieses auch ändern. Bei einer Lesesperre können mehrere Prozesse das gesperrte Objekt lesen.
Die konservative Art des 2PL verhindert einen Deadlock. Dies wird umgesetzt indem bei Beginn der Transaktion alle benötigten Sperren auf einmal gesetzt werden. Die strikte Art ist die häufiger verwendete Art von 2PL. Hier werden alle gesetzten Write-Locks erst am Ende der Transaktion freigegeben. [21]

7.2.4 Long-duration Transaction

Sehr viele Businesstransaktionen sind Long-term-/Long-running-/Long-duration-Tranaktionen. Diese laufen meist wie der Name sagt für eine lange Zeit, sogar für Stunden oder Tage. Daher werden meistens Timelimits gesetzt. Nach Ablauf des Limits geschieht ein Rollback. [21]

7.3 Transaktionskonflikte

7.3.1 ACID

Das Transaktionsprinzp sollte dem ACID-Schema folgen. [21]

- Atomicity
  Transaktion wird entweder ganz oder gar nicht ausgeführt
- Consistency
  Datenbank ist vor Beginn und nach Beendigung einer Transaktion jeweils in einem konsistenten Zustand
- Isolation
  Nutzer, der mit einer Datenbank arbeitet, sollte den Eindruck haben, dass er mit dieser Datenbank alleine arbeitet
- Durability
  Nach erfolgreichem Abschluss einer Transaktion muss das Ergebnis dieser Transaktion 'dauerhaft' in der Datenbank gespeichert werden.

7.3.2 Probleme im Mehrbenutzerbetrieb und Lösungsansätze

Folgendes wird durch ACID-konformen Transaktionen verhindert

- Inkonsistentes Lesen: Nonrepeatable Read
- Abhängigkeiten von nicht freigegebenen Daten: Dirty Read
- Das Phantom-Problem
- Verlorengegangenes Ändern: Lost Update

Im 2PC-Betrieb kann wie bereits kurz angesprochen nach dem Vote der Koordinator ausfallen. Die Teilnehmer müssen daraufhin nach einem Timeout aborten. Das ist jedoch quasi Rückgängigmachen einer bereits getroffenen Entscheidung.
Im 3PC-Fehlerfall (Ausfall des Koordinators und k-1 Teilnehmer) sind alle weiteren Teilnehmer im Zustand 'Ready'. Die ausgefallenen Teilnehmer können nur in den Zuständen 'Ready', 'Abort' oder 'Pre-Commit' sein, weswegen die Transaktion abgebrochen werden kann. Sollte einer der Teilnehmer im Zustand 'Pre-Commit' oder 'Commit' sein, so muss ein neuer Koordinator das Protokoll fortsetzen. [22]

Literatur

[1] Burkhard Hampl und Simon Wortha. Cloud computing-ausarbeitung systemtechnik 2015/16. https://elearning.tgm.ac.at/mod/resource/view.php?id=48953. zuletzt besucht: 16.05.2016.

[2] Margaret Rouse. Definition - microsoft azure (windows azure). http://searchcloudcomputing.techtarget.com/definition/Windows-Azure. zuletzt besucht: 16.05.2016.

[3] Margaret Rouse. Definition - internet der dinge (internet of things, iot). http://www.searchnetworking.de/definition/Internet-der-Dinge-Internet-of-Things-IoT. zuletzt besucht: 16.05.2016.

[4] whatis.com. Definition - interprocess communication (ipc). http://whatis.techtarget.com/definition/interprocess-communication-IPC. zuletzt besucht: 16.05.2016.

[5] Selina Brinnich und Niklas Hohenwarter. Service-oriented architecture. https://elearning.tgm.ac.at/mod/resource/view.php?id=45143. zuletzt besucht: 16.05.2016.

[6] TU Clausthal. Remote procedure call. http://zach.in.tu-clausthal.de/teaching/programming_0506/literatur/linuxfibel/rpc.htm. zuletzt besucht: 16.05.2016.

[7] Anja Austermann. Rmi – verteilte programmierung unter java. http://www.cul.de/data/jbuilderpr.pdf. zuletzt besucht: 16.05.2016.

[8] ITWissen. Mom (message oriented middleware). http://www.itwissen.info/definition/lexikon/message-oriented-middleware-MOM.html. zuletzt besucht: 16.05.2016.

[9] Karic/Adler. Security. https://elearning.tgm.ac.at/mod/resource/view.php?id=44595. zuletzt besucht: 16.05.2016.

[10] Gemalto. Securing the internet of things (iot). http://www.safenet-inc.com/data-protection/securing-internet-of-things-iot/#. zuletzt besucht: 16.05.2016.

[11] Apache Foundation. Apache shiro. http://shiro.apache.org/. zuletzt besucht: 16.05.2016.

[12] Spring. Spring security. http://projects.spring.io/spring-security/. zuletzt besucht: 16.05.2016.

[13] Ernhofer/Kopec. Autorisierung und authentifizierung. https://elearning.tgm.ac.at/mod/resource/view.php?id=44594. zuletzt besucht: 16.05.2016.

[14] itwissen.info. Ldap (lightweight directory access protocol). http://www.itwissen.info/definition/lexikon/lightweight-directory-access-protocol-LDAP-LDAP-Protokoll.html. zuletzt besucht: 16.05.2015.

[15] CE-Team. Diplomarbeit competence editor. https://github.com/mgoebel-tgm/CompetenceEditor/blob/development/doc/diploma_thesis/CompetenceEditor_Diplomarbeit.pdf. zuletzt besucht: 16.05.2016.

[16] Margaret Rouse. Definition - kerberos. http://www.searchsecurity.de/definition/Kerberos. zuletzt besucht: 16.05.2016.

[17] Braendli/Weinberger. Synchronisation und konsistenz. https://elearning.tgm.ac.at/mod/resource/view.php?id=48957. zuletzt besucht: 16.05.2016.

[18] ElKo. Tls - transport layer security. http://www.elektronik-kompendium.de/sites/net/1706131.htm. zuletzt besucht: 16.05.2016.

[19] Thomas Kothmayr. Security for the internet of things. http://kothmayr.net/research/dtls/. zuletzt besucht: 16.05.2016.

[20] Koelbl/Taschner. Load balancing. https://elearning.tgm.ac.at/mod/resource/view.php?id=46895. zuletzt besucht: 16.05.2016.

[21] Goebel/Perny. Transaktionen und nebenläufigkeit. https://elearning.tgm.ac.at/mod/resource/view.php?id=46942. zuletzt besucht: 16.05.2016.

[22] Uni Magdeburg. Vdm 4. http://www.cs.uni-magdeburg.de/iti_db/lehre/vfdb/folien/vdm-4.pdf. zuletzt besucht: 16.05.2016.