Seminar Smart Services & IoT

Thema: Entwicklung eines Prototypen auf IBM Bluemix

Betreuer:

M.Sc. Sebastian Bader (AIFB) , Rainer Hochecker (IBM)

Prof. Dr. York Sure-Vetter,
Institut für Angewandte Informatik und
Formale Beschreibungsverfahren (AIFB)

Sebastian Dahnert, Matrikel-Nr.: 1999526

Marcel Hofmann, Matrikel-Nr.: 2069630

Daniel Schlotthauer, Matrikel-Nr.: 1721248

Inhalt

Abbildungsverzeichnis

Motivation

Blockchain, die Technologie hinter der Kryptowährung Bitcoin, hat seit der Veröffentlichung des Whitepapers “Bitcoin: A Peer-to-Peer Electronic Cash System” im Jahr 2008 eine rasante Entwicklung erlebt. Die Evolution der Technologie umfasst sowohl technologische Optimierung, als auch Erweiterungen des Anwendungsspektrums. Ursprünglich entwickelt, um die doppelte Ausgabe digitaler Währungen zu verhindern, kann Blockchain nach finanziellen Transaktionen das gesamte Internet revolutionieren. Swan bezeichnet Blockchain, nach dem PC, dem Internet, dem Smartphone und den sozialen Netzwerken, als das fünfte disruptive Computerparadigma (Swan 2015, S. xi). Weitreichende Auswirkungen auf Geschäftsmodelle, Volkswirtschaften, Gesellschaft und sogar Umwelt zeichnen sich allmählich ab. Intelligente Verträge, autonome Organisationen und sogar staatliche Angebote werden mittlerweile basierend auf einer Blockchain angeboten. Eines der vielversprechenden Anwendungsgebiete ist der IoT-Kontext. Sensoren sammeln permanent Daten über Prozessschritte, Kundeninteraktionen und Transaktionen. Authentifizierung, Kommunikation und selbst Transaktionen zwischen den Geräten sollen spätestens seit dem IoT-Blockchain Start-up IOTA blockchain basiert ermöglicht werden. So können Maschinen untereinander Vertrauen schaffen und unabhängig vom Menschen intelligente Verträge abschließen.

Die vorliegende schriftliche Ausarbeitung hat das Ziel einen konkreten Anwendungsfall der Blockchain als Bestandteil einer IoT-Anwendung darzustellen und eine konkrete Implementierung in Form einer Demo vorzustellen.

Einleitung

Im Rahmen des Seminars Smart Services & IoT in Kooperation mit IBM sollte ein Prototyp als Showcase entwickelt werden, der die Fähigkeiten der IBM Cloud (ehem. IBM Bluemix) veranschaulicht und dabei nach Möglichkeit auch mobil einsetzbar ist (bspw. im Vertrieb). Die IBM Cloud ist eine sog. Platform-as-a-Service (PaaS). Über die IBM Cloud können Entwickler auf mehr als 130 Cloud-Services zugreifen, um mobile Apps und Webanwendungen zu entwickeln. Die zahlreichen Analysewerkzeuge werden zudem ergänzt durch Services von Drittanbietern. IBM Cloud unterstützt mehrere Programmiersprachen und Frameworks, u.a. Java, Node.js, Go, PHP, Python, Ruby Sinatra, Ruby on Rails und kann auch andere Sprachen wie Scala durch den Einsatz von Buildpacks unterstützen (vgl. für diesen Absatz: IBM Cloud Docs 2017).

Die offene Aufgabenstellung eröffnete eine Vielzahl an Anwendungsmöglichkeiten, die zunächst spezifiziert und eingegrenzt werden mussten. Zur Unterstützung des Brainstormings wurde uns von IBM eine Liste mit Use Cases in verschiedensten Anwendungsgebieten und Entwicklungsstufen bereitgestellt, das IBM Watson IoT Center in München besucht sowie mit mehreren Mitarbeitern von IBM Rücksprache gehalten. Die Entscheidung fiel dann auf eine IoT-Tracking-Demo, die der dual-studierende Masterand Paul Wenzel bei IBM entwickelt hat. Diese Anwendung sollte so erweitert werden, dass gewisse Datensätze aus dem Tracking-Device in einer Blockchain persistiert werden. Im

Folgenden wird daher zunächst auf den Use Case, für den diese Demo entwickelt wurde, eingegangen und die Implementierung detailliert beschrieben. Als Abschluss werden noch weitere Möglichkeiten für den vorliegenden Use Case beschrieben und etwaige Einschränkungen diskutiert.

Use Case

Der deutsche Automobilhersteller A möchte seine fertiggestellten PKW von Stuttgart nach Peking transportieren. Hierzu beauftragt er die Logistikanbieter A, B und C. Diese sollen die fertigen Autos am Lager abholen und bis nach Peking transportieren, wobei Logistikanbieter A für den Transport innerhalb Deutschlands, Logistikanbieter B für den Schiffstransport und Logistikanbieter C für den Transport innerhalb von China verantwortlich ist. Hierbei kann es zu einer Reihe von Komplikationen und Problemen kommen, welche durch eine blockchain-basierte Lösung adressiert werden können.

In der Praxis kommt es entlang der Lieferkette häufig zu Schäden der zu transportierenden Ware. Angenommen die Stelle in Peking erhält die gelieferten PKW beschädigt. Für den Automobilhersteller ist nicht nachweisbar, welcher der Logistikanbieter für den Schaden verantwortlich ist. Dementsprechend ist die Rechtsgrundlage für Versicherung, Versicherungsnehmer und etwaige Ansprüche unklar, der anschließende Klärungsprozess langwierig und kompliziert.

Eine mögliche Lösung ist die Ausstattung der Container mit IoT-fähigen Sensoren, die in regelmäßigen Abständen die gemessenen Daten wie Beschleunigung, GPS, Temperatur u.v.m. an eine Datenbank in der IBM Cloud senden und speichern. Somit können alle beteiligten Parteien sicherstellen, wann ruckartige Bewegungen oder andere Anomalien stattgefunden haben. Dies wurde mit der Grundlage von Paul Wenzel bereits realisiert.

Diese Lösung alleine würde jedoch nicht ausreichen. Bei einer einzigen Datenbank, kontrolliert und aktualisiert von einer einzigen Partei, könnten einzelne ermittelte Einträge in der Datenbank verändert oder gelöscht werden, ohne dass dies Auswirkungen auf andere Einträge hätte. Eine Blockchain, deren Einträge mittels kryptografischen Mechanismen verschlüsselt, unveränderbar und dank Konsens-Mechanismen mit verifizierbar widerspruchsfreien Daten gefüllt sind, stellt eine potenzielle Lösung dar.

Alle Parteien haben Zugriff auf die verteilte Datenbank, mittels derer sie verifizieren können wann und an welchem Ort der Ausschlag gemessen wurde. Die Sensorik misst dabei die Beschleunigungswerte kontinuierlich, in die Blockchain selbst werden jedoch nur Messungen aufgenommen, die einen vorher definierten Schwellwert übersteigen.

Solch eine Überschreitung wird als Exception definiert. Sie enthält Informationen zu Zeitpunkt, Standort und der Art der Überschreitung. Der Aufbau des Use Cases ist in Abbildung 1 veranschaulicht.

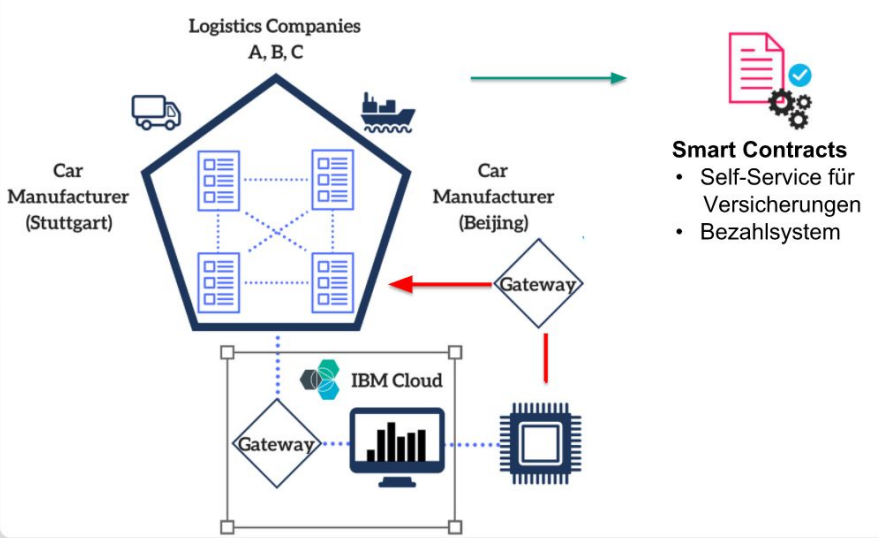

Abbildung 1: Use Case (eigene Darstellung)

In der Praxis ist es häufig so, dass ein Container an einem Hafen bzw. einem Umschlagplatz das Transportmittel und damit verbunden eventuell auch das Logistik-Unternehmen wechselt. Hier haben sowohl der bisherige als auch der neue “Shipper” ein begründetes Interesse daran, die Verantwortung für die Lieferung zu übertragen. So kann weder der bisherige Shipper für Schäden nach der Übergabe verantwortlich gemacht werden, noch der neue Shipper für vor der Übergabe entstandene Schäden.

Jenseits der Unveränderbarkeit der persistierten Daten gibt es weitere Argumente, die für den Einsatz der Blockchain sprechen: So können in diesem Use Case weitere Parteien wie Versicherer und Zollbehörden in das Netzwerk eingebunden werden, um mittels definierter Smart-Contracts und Self-Service-Tools den bürokratischen Aufwand während des Lieferprozess und der Abwicklung eines Schadenfalls zu reduzieren.

Im Folgenden werden die Teilnehmer, Assets und Transaktionen sowie deren Funktionen beschrieben:

Teilnehmer

- Dispatcher: Auftraggeber, erstellt Sendungen
- Recipient: Empfänger, bestätigt Erhalt der Sendung
- Shipper: Logistikunternehmen, transportiert Sendungen, übergibt sie u.U. an einen anderen Shipper oder übernimmt sie von einem anderen Shipper
- Insurer: Versicherungsunternehmen
- Device: zeichnet Sensordaten auf und überträgt sie an die IBM Cloud und die Blockchain

Assets

- Contract: Vereinbarung zwischen einem Dispatcher, einem Recipient und mindestens einem Shipper, beinhaltet ein vereinbartes Ankunftsdatum
- Shipment: transportiertes Gut, ist einem Contract, einem Insurer und einem Device zugeordnet. Hat einen Status (CREATED, IN_TRANSIT, RELEASED, ARRIVED) und eine Liste mit (eventuell) aufgezeichneten Exceptions

Transaktionen

- ShipmentException: Der Sensor hat einen Messwert erfasst, der über dem Schwellwert liegt. Eine kurze Beschreibung der Exception wird zusammen mit GPS-Koordinaten, Timestamp und Transaktions-ID im Feld des Shipments gespeichert.
- ShipmentReceived: Die Sendung ist beim Empfänger angekommen. Damit wird der Status des Shipments auf ARRIVED gesetzt, Änderungen sind damit nicht mehr möglich.
- ShipmentRelease: Die Sendung befindet sich an einem Umschlagplatz und muss an den nächsten Shipper übergeben werden. Hierzu gibt der aktuelle Shipper die Sendung frei (Status wird auf RELEASED gesetzt).
- ShipmentOvertake: Der neue Shipper übernimmt die freigegebene Sendung mit den gleichen Parametern wie bei einem ShipmentRelease (shipper_old, shipper_new). Dabei wird überprüft, ob die Sendung überhaupt freigegeben wurde und ob der Parameter “shipper_old” auch dem tatsächlich bisher zuständigen Shipper entspricht. Der Status wird wieder auf IN_TRANSIT gesetzt.

Implementierung

Im Folgenden wird der generelle Aufbau des Projekts näher beschrieben. Zunächst wird der aktuelle Stand von Paul Wenzel aufgezeigt und dann darauf eingegangen, wie diese Anwendung um die Anbindung an die Blockchain erweitert wurde. Anschließend wird die Modellierung des Use Cases im Hyperledger Composer Playground beschrieben. Zum Schluss werden die Demo-Anwendungen sowie die Umgebung, in welcher das Projekt läuft erläutert.

Grundlagen

Als Grundlage für die Demo-Anwendung diente eine IoT-Demo, welche von Paul Wenzel entwickelt wurde. In dieser Anwendung wird ein Sensor für einen Schiffscontainer mit Hilfe einer App simuliert, welche in Ionic entwickelt wurde. Die App sammelt von den im Handy verbauten Sensoren Informationen zur Beschleunigung (*acceleration*), der Orientierung (*gyroscope*) und dem Standort (*location*). Die Daten werden in einem hohen Intervall über MQTT an die Watson IoT Plattform gesendet, wo sie in einer Cloudant-Datenbank persistiert werden (siehe Abbildung 2).

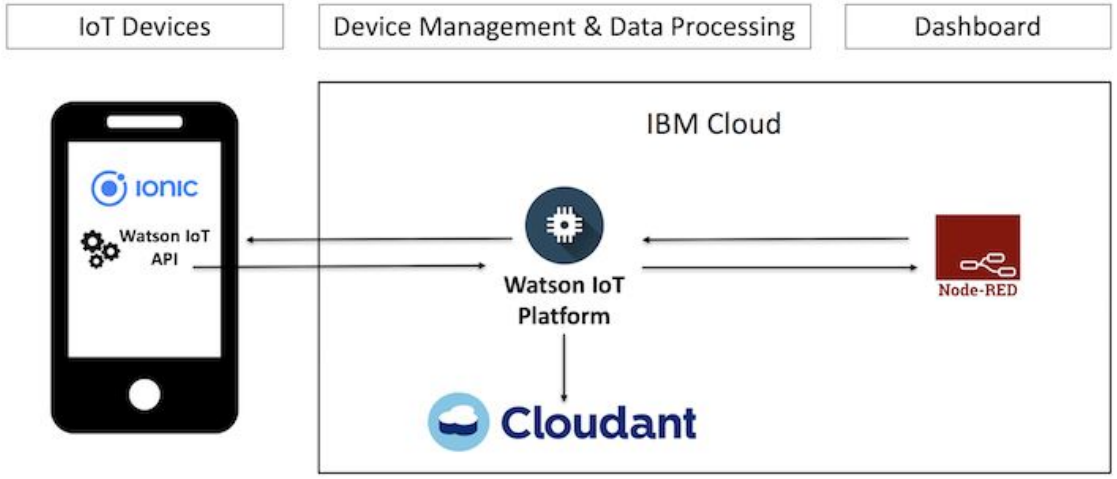

Abbildung 2: Architektur der ursprünglichen Demo-Anwendung (Hyperledger Shipment Container App Architecture 2018)

Zur Darstellung wurde eine Oberfläche mit Node-RED entwickelt, welche auf die Daten der Watson IoT Plattform zugreift (Hyperledger Shipment Dashboard 2018). Zusätzlich gibt es die Möglichkeit von der Web-Oberfläche aus, ein Bild auf dem Handy zu machen, was auch über MQTT geregelt ist. Eine ausführliche Beschreibung der implementierten Funktionalitäten ist im entsprechenden Repository (Hyperledger Shipment Container App 2018) einsehbar.

Erweiterung der Container App

Diese Anwendung wurde um eine Anbindung an die Hyperledger Blockchain erweitert. Im Gegensatz zur Watson IoT Plattform ist es in der Blockchain nicht praktikabel die Sensordaten mit so einer hohen Auflösung zu übertragen. Dies würde die Größe der Blockchain schnell hoch treiben und den Rechenaufwand unnötig steigern. Daher ist es nötig nur die Ausnahmen in die Blockchain zu schreiben und die Prüfung der Ausnahmen in den Client zu übernehmen. In der Vorlage von Paul Wenzel fand diese in Node-RED statt, was wiederum ein Vertrauen in die nicht verifizierbare Implementierung dort verlangt. Da dies dem Sinn und Zweck der Verwendung einer Blockchain widerspricht, müssen die Daten vom Gerät direkt in die Blockchain geschrieben werden, denn unter der Annahme, dass alle Parteien dem verwendeten Sensorgerät vertrauen, wird so kein zusätzliches Vertrauen benötigt.

Zur Anbindung wurde der Service *BlockchainService* programmiert (Hyperledger Shipment Container App Blockchain Service 2018). Die Funktion *checkException* prüft immer, wenn neue Sensordaten verfügbar sind, ob gewisse Schwellwerte über- oder unterschritten wurden. Für die Demonstration wurde lediglich die Orientierung beachtet, also geprüft, ob nur die Beschleunigung in z-Richtung einen hohen Wert hat. Es ist an der Stelle denkbar die Prüfung zu erweitern, beispielsweise indem der Standort geprüft wird (Abweichung von geplanter Route), oder eine schnelle Drehung erkannt wird (über das Gyroskop). Wird eine Schwellwertüberschreitung festgestellt wird diese zunächst nur gespeichert.

Alle fünf Sekunden wird unabhängig davon geprüft, ob eine gespeicherte Ausnahme vorliegt. Ist dies der Fall wird über ein POST-Request die Ausnahme an einen Composer-REST-Server geschickt (Generating a REST API | Hyperledger Composer 2018). Dieser verlangt in der Demo keine Authentifizierung, jedoch ist es über die Option *multiuser* möglich eine Authentifizierung über die Wallet zu erzwingen. Teil der Exception sind neben der Nachricht der Ausnahme noch der Standort und die genaue Uhrzeit der Schwellwertüberschreitung.

Hyperledger Blockchain

Um schnell und mit wenig Aufwand kleine Business Networks zu implementieren, wird der Hyperledger Composer Playground angeboten. Darin ist es möglich, eigene Business Networks zu modellieren oder auf Beispiel-Modelle zurückzugreifen. Die Entwicklung findet dann direkt im Browser statt und kann darin auch getestet werden. Unter dem “Define”-Tab wird dabei das Modell definiert und im “Test”-Tab kann es dann instanziiert werden und der Verlauf der Blockchain-Einträge eingesehen werden. Der Hyperledger Composer Playground ist jedoch nur optional, Business Networks können mit beliebigen anderen Editoren definiert werden und müssen dann über die Kommandozeile deployed werden.

Der Grundlegende Aufbau und das Vorgehen bei der Entwicklung eines Business Networks ist im Playground Tutorial (Playground Tutorial | Hyperledger Composer 2018) sehr ausführlich beschrieben. Aus diesem Grund wird im Folgenden nur auf die wesentlichen Punkte eines Business Networks eingegangen.

In der Modelldatei (Hyperledger Shipment Business Network Model File 2018) werden die Teilnehmer, Assets, deren Attribute sowie Transaktionen und Events definiert. Im *Permissions File* (Hyperledger Shipment Business Network Permissions File 2018) werden Zugriffsregeln für die verschiedenen Gruppen des Business Network definiert. Standardmäßig sind hier zwei Zugriffsregeln für den Network-Admin definiert, der vollen Zugriff auf alle Teile des Business Networks hat.

In der Skriptdatei (Hyperledger Shipment Business Network Script File 2018) wird der sog. Chaincode in JavaScript definiert. Für jede in der Modelldatei definierten Transaktion werden hier die Abläufe implementiert und teilweise ein Event erzeugt. So wird bpsw. beim Auslösen der *shipmentReceived*-Transaktion der Status des Shipments aktualisiert und der Vorgang in der Konsole ausgegeben. Zusätzlich wurde hier noch die *SetupDemo*-Transaktion definiert. Mit Hilfe dieser Transaktion werden mehrere Teilnehmer und Assets instanziiert, um schnell die Funktionalität des Netzwerks zu testen.

Bei der Instanziierung (sowohl über die *SetupDemo*-Transaktion als auch über den Composer Playground unter *“Test”*) werden dann alle im Modell definierten Attribute und Regeln überprüft.

Optional kann dem Business Network maximal eine Query-Datei hinzugefügt werden, die den gezielten Abruf von Einträgen der Blockchain mittels Queries ermöglicht. Aus zeitlichen und Komplexitätsgründen im Bezug zur Aufgabenstellung wurde in diesem Projekt jedoch darauf verzichtet.

Demo-Oberfläche

Um die in der Blockchain implementierten Funktionalitäten zu demonstrieren wurde eine weitere Ionic-App entwickelt (Hyperledger Shipment Demo 2018). In dieser Anwendung sind im wesentlichen zwei Szenarien abgedeckt: Erstens, die Möglichkeit alle in der Blockchain gespeicherten Informationen zu einem Shipment einzusehen, inklusive der Exceptions und zweitens, der Prozess der Übergabe des Shipments von einem Lieferanten an den nächsten.

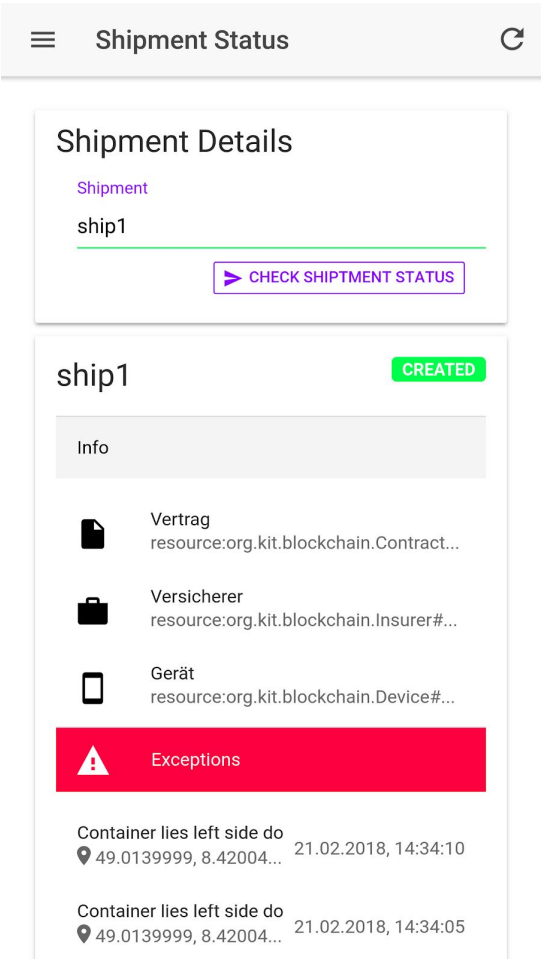

Abbildung 3: Detailansicht einer Lieferung (eigene Darstellung)

Abbildung 3 zeigt die Detailansicht eines Shipments. Durch einen Klick auf ein Element (bspw. Vertrag oder Versicherer) können zusätzliche Informationen angezeigt werden. Die Informationen hier werden ebenfalls über den Composer-REST-Server abgerufen. Während diese Seite der App geöffnet ist, wird zusätzlich eine Websocket-Verbindung mit dem REST-Server aufgebaut, welche es ermöglicht auf Events in der Blockchain in Echtzeit zu reagieren. Wird also eine Exception vom Container gemeldet, löst der Chain-Code ein *ShipmentExceptionEvent* aus, welches wiederum in der Demo-App dafür sorgt, dass dies angezeigt wird und der Inhalt aktualisiert wird.

Auf der *Transfer*-Seite, welche über das seitliche Menü geöffnet werden kann, wird der Prozess abgedeckt, um ein Shipment von einem Lieferanten an den nächsten zu

Übergeben. Hier wurde erneut, um die Demo einfach zu halten, auf eine Authentifizierung verzichtet. Als Pseudo-Authentifizierung gilt hier die Angabe der ID eines Lieferanten in der Seitenleiste. Entspricht diese ID dem Lieferanten des Shipment wird die Option angezeigt, das Shipment freizugeben. Die Prüfung der ID findet dann zusätzlich im Chain-Code statt. Ist das Shipment hingegen im Status "*RELEASED*", wird angeboten, dass Shipment zu übernehmen (*overtake*). Wie auf der Detailansicht wird hier ein Websocket verwendet um über Änderungen an der Blockchain in Echtzeit zu informieren.

IBM Cloud

Während die Ionic-Apps jeweils lokal ausgeführt werden (wobei die Demo-Anwendung prinzipiell auch auf einem Server gehostet werden kann), laufen sowohl das Node-RED-Dashboard, als auch die Blockchain auf der IBM Cloud. Während das Node-RED-Repository mit nur einem Klick in der IBM Cloud deployed werden kann (siehe hierzu Hyperledger Shipment Container App ReadMe-File 2018), bedarf es bei der Blockchain etwas mehr Aufwand.

Die IBM bietet zwar eine "fertige" Blockchain-Plattform in der IBM Cloud an (IBM Blockchain Platform 2018), jedoch ist diese aus mehreren Gründen nicht für diese Demo geeignet. Im Gegensatz zu anderen Services der IBM Cloud ist die Blockchain nicht mit einem kostenlosen Kontingent verfügbar, und muss erst manuell freigeschalten werden. Außerdem würde der Einsatz einer Blockchain-as-a-Service wiederum ein viel größeres Vertrauen in IBM als Institution verlangen, ein Aspekt der bewusst vermieden werden soll.

Stattdessen wird der Einsatz in einem mit Kubernetes verwalteten Cluster empfohlen. Für den Zweck der Demo wurde eine Installationsanleitung der IBM verfolgt, welche mit wenig Aufwand eine Hyperledger-Umgebung instanziiert. Nach Abschluss der Installation sind folgende Komponenten verfügbar (IBM Blockchain 2017):

- Zwei "Fabric Peer Nodes" (jeweils eine für die Organisationen "org1" und "org2"
- Ein "Orderer Node"
- Drei "Fabric Certificate Authorities" (jeweils eine pro Peer)
- Composer Playground (Oberfläche zum Bearbeiten und Ausrollen des Chain-Codes)
- Composer-REST-Server (zur Anbindung der Ionic-Apps)

Für die Demo ist dieser Ansatz vollkommen ausreichend. Es wird eine vollständige Blockchain verwendet, welche auf einem Remote-Server ausgeführt wird. Zudem wird die Verifikation von bereits mehreren getrennten Peers durchgeführt. In einer realen Anwendung hingegen sollte die Konfiguration des Installationsskripts jedoch genau überprüft werden und allen Geschäftspartnern das Netzwerk zur Verfügung gestellt werden, sodass sie mit eigenen Peers an der Blockchain teilnehmen können.

Diskussion

Während des Projekts hat sich immer klarer herausgestellt, dass Blockchain kein vollkommenes Vertrauen schaffen kann. Jedoch muss an irgendeiner Stelle immer ein Mindestmaß an Vertrauen herrschen, insbesondere im IoT-Kontext. Die Bitcoin-Blockchain stellt ein abgeschlossenes System dar, dessen Anwendungsspektrum sich auf den Kauf und Verkauf von Kryptowährung bezieht. Im IoT-Kontext wird einerseits das Anwendungsspektrum erweitert, andererseits erweitern sich die Berührungspunkte mit der realen Welt. Beispielsweise muss man der zugrundeliegenden Hardware, beziehungsweise dem Hersteller des Sensors und “IBM”, die das Netzwerk initial aufsetzt, vertrauen. Blockchain kann hier die Situation nur verbessern, aber die Notwendigkeit des Vertrauens nicht komplett aus der Welt schaffen.

Darüber hinaus ist fraglich, ob große Unternehmen geeignet sind Blockchain-as-a-Service anzubieten. Ein großer Vorteil der Technologie ist die Abschaffung des Single-Point-of-Failure. Bei einem einzigen Unternehmen, das die Blockchain orchestriert ist jedoch genau diese Gefahr gegeben. Die vorgestellte Architektur (vgl. Abbildung 1) zeigt den idealen Fall, in der die IBM nicht für das Persistieren der Datenbank verantwortlich ist, sondern die Hardware direkt die verteilte Datenbank mit Daten füllt. Dennoch bleibt die Frage der Notwendigkeit einer blockchainbasierten Lösung, denn im geschäftlichen Umfeld ist das wertschöpfende Netzwerk normalerweise bekannt. In Kombination mit Gesetzen und Verträgen, die generell ein Substitut für Vertrauen darstellen, verliert der Vertrauensaspekt zunehmend an Gewicht um den Einsatz der blockchain zu rechtfertigen. Hinzu kommt, dass durch das “Aufzwingen” der Blockchain die Beziehung zu Geschäftspartnern gestört werden kann. Zudem hat sich während des Projekts herausgestellt, dass viele der Anwendungsfälle für Blockchain den Großteil ihres Mehrwertes durch die Digitalisierung, beziehungsweise Automatisierung des Prozesses an sich schaffen und sich der zusätzliche Mehrwert der Blockchain in Grenzen hält.

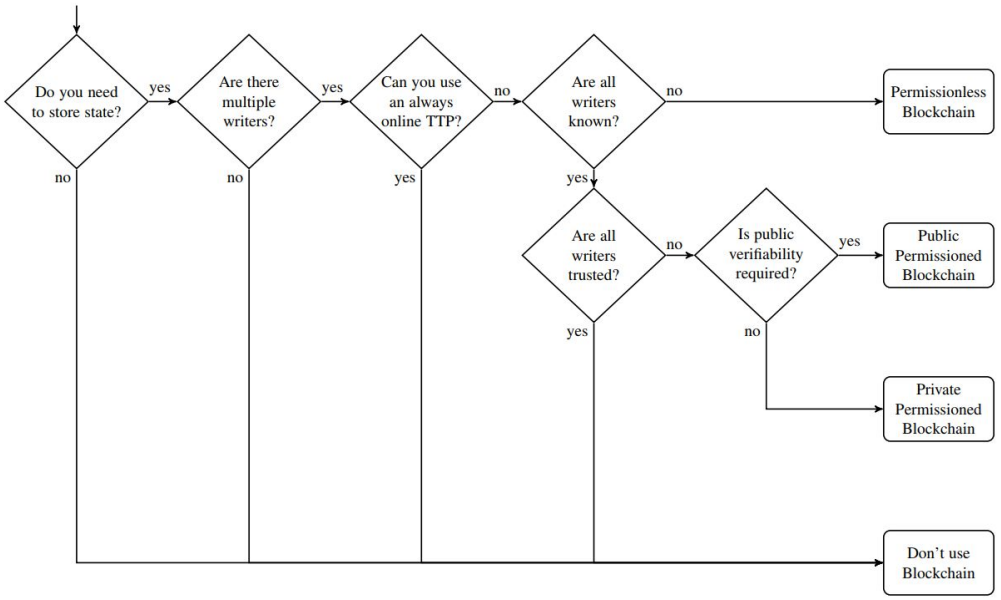

Abbildung 4: "Do you need a Blockchain?" (Wüst, Gervais 2017)

Abschließend lässt sich feststellen, dass die Umsetzung von blockchainbasierten Anwendungen im IoT-Kontext, dank der Cloud-Plattform der IBM verhältnismäßig einfach durchzuführen ist. Zumindest lassen sich einfache Prototypen umsetzen, was die zahlreichen Use-Cases von Start-ups und etablierten Unternehmen unterstreichen. Jedoch ist es sehr schwierig, einen konkreten Fall zu finden, der einen Einsatz einer Blockchain rechtfertigt (vgl. hierzu Abb. 4). Bei einem Abklingen des derzeitigen Blockchain Hypes könnte es sein, dass weniger Kunden proaktiv in Erfahrung bringen wollen, wie sie die Blockchain-Technologie einsetzen können, sich aber gleichzeitig bis dahin einige wenige blockchain-rechtfertigende Anwendungsfälle etablieren werden.

Ausblick

Eine potenzielle Erweiterung des Anwendungsfalls ist die Verifikation von Transaktionen durch einen oder mehrere Peers. Die Endorsement-Policies werden verwendet, um einen Peer darüber zu informieren, wie entschieden werden muss, ob eine Transaktion ordnungsgemäß unterstützt wird. Wenn ein Peer eine Transaktion empfängt, ruft er den VSCC (Validierungssystem-Chaincode) auf, der dem Chaincode der Transaktion als Teil des Transaktionsvalidierungsflusses zugeordnet ist, um die Gültigkeit der Transaktion zu bestimmen. (vgl. für diesen Abschnitt: Endorsement policies - hyperledger-fabricdocs master documentation 2017)

Ein weiterer Ausbau der Anwendung könnten sogenannte "Channels" sein. Dabei handelt es sich um private Kanäle in denen vertrauliche Informationen zwischen vorher autorisierten Peers ausgetauscht werden. In der aktuellen Applikation wären vertrauliche Informationen wie Preise der Logistikdienstleister oder Guthaben der Accounts vollkommen transparent und jedem zugänglich. Im geschäftlichen Umfeld ist dies in der Regel unüblich. Mittels privater Kanäle können solche Informationen problemlos und diskret ausgetauscht werden.

Quellenverzeichnis

Endorsement policies - hyperledger-fabricdocs master documentation 2017, http://hyperledger-fabric.readthedocs.io/en/release/endorsement-policies.html, abgerufen am 13.11.2017

Generating a REST API | Hyperledger Composer 2018, https://hyperledger.github.io/composer/integrating/getting-started-rest-api.html, abgerufen am 04.01.2018

Hyperledger Composer Playground 2018, https://composer-playground.mybluemix.net/login, abgerufen am 12.02.2018

Hyperledger Shipment Business Network Model File 2018, https://github.com/m2hofi94/hyperledger_shipment-business_network/blob/v0.1/models/org.kit.blockchain.cto, abgerufen am 18.02.2018

Hyperledger Shipment Business Network Permissions File 2018, https://github.com/m2hofi94/hyperledger_shipment-business_network/blob/v0.1/permissions.acl, abgerufen am 18.02.2018

Hyperledger Shipment Business Network Script File 2018, https://github.com/m2hofi94/hyperledger_shipment-business_network/blob/v0.1/lib/logic.js, abgerufen am 18.02.2018

Hyperledger Shipment Container App 2018, https://github.com/m2hofi94/hyperledger_shipment-container_app, abgerufen am 18.02.2018

Hyperledger Shipment Container App Architecture 2018, https://github.com/m2hofi94/hyperledger_shipment-container_app/blob/master/documentation/architecture.png, abgerufen am 18.02.2018

Hyperledger Shipment Container App Blockchain Service 2018, https://github.com/m2hofi94/hyperledger_shipment-container_app/blob/v0.1/src/providers/blockchain/blockchain.ts, abgerufen am 18.02.2018

Hyperledger Shipment Container App ReadMe-File 2018, https://github.com/m2hofi94/hyperledger_shipment-container_app/blob/v0.1/README.md, abgerufen am 18.02.2018

Hyperledger Shipment Dashboard 2018, https://github.com/m2hofi94/hyperledger_shipment-dashboard, abgerufen am 18.02.2018

Hyperledger Shipment Demo 2018, https://github.com/m2hofi94/hyperledger_shipment-demo, abgerufen am 18.02.2018

IBM Blockchain 2017, https://ibm-blockchain.github.io/, abgerufen am 28.12.2017

IBM Blockchain Platform 2018, https://www.ibm.com/blockchain/platform/, abgerufen am 21.02.2018

IBM Cloud Docs 2017, https://console.bluemix.net/docs/overview/ibm-cloud.html#overview, abgerufen am 04.12.2017

Playground Tutorial | Hyperledger Composer 2018, https://hyperledger.github.io/composer/tutorials/playground-tutorial.html, abgerufen am 08.02.2018

Swan, M. (2015). Blockchain: Blueprint for a New Economy, O´Reilly Media, Sebastopol, CA.

Wüst, K., Gervais, A. (2017). Do you need a Blockchain?. IACR Cryptology ePrint Archive, 2017, 375

Eidesstattliche Erklärung

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, 21.02.2018

_____

Sebastian Dahnert

_____

Marcel Hofmann

_____

Daniel Schlotthauer