fabrico

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

Additive Fertigung mit Distributed Ledger Technologie

Entwicklung eines Wertschöpfungsnetzwerkes auf Basis eines Distributed Ledger für die verteilte computergestützte Fertigung und Lizenzierung von Bauteilen

1. Studienarbeit – 5. Semester
Kurs: TIF20A
Betreuer: Prof. Dr. Jan M. Olaf
Jennifer L. Krüger, Fabian Zaremba

1 Themenabgrenzung

Ziel der vorliegenden Arbeit ist die Absicherung von Fertigungsprozessen (z.B. 3D-Druckdaten zur additiven Fertigung oder CNC Maschinendaten) inklusive Lizenzierungsmöglichkeit für Hersteller. Dabei wird ein Wertschöpfungsnetzwerk aus Dateneignern (Originalherstellern), Druckdienstleistern und perspektivisch auch Datenlogistikern mit Hilfe eines im Rahmen dieser Arbeit speziell entwickelten Distributed Ledgers aufgebaut.

Am Studienzentrum für Informatik der DHBW Lörrach dient die vorhandene Forschungsplattform für Additive Fertigung und 3D Topometrie als Grundlage für die experimentelle Erprobung des entwickelten Prototypen. Die vorhandene Plattform bietet die Möglichkeit einer Ansteuerung über Ethernet Schnittstelle (GCODE via TCP).

2 Theoretische Grundlagen

2.1 Distributed Ledger und Blockchain

Die Blockchain stellt eine technische Lösung zur Datenverbreitung innerhalb einer dezentralen verteilten Netzwerkinfrastruktur dar. Diese ermöglicht es, die ausgehändigten Daten nachvollziehbar, aber vor allem manipulationssicher im gegenseitigen Konsens der einzelnen Teilnehmer (hier auch als Nodes bezeichnet) zu verwalten.

Dabei werden einzelne Anfragen sowie Blöcke (die mehrere Anfragen enthalten können) mittels kryptografischer Verfahren, wie beispielsweise Hashfunktionen oder digitalen Signaturen gesichert und zu einer Kette verknüpft – die sogenannte „Blockchain“. Dennoch kann es inbesondere durch Fehlimplementierungen, ungeschützten Netzwerkprotokollen oder mangelhaft gesicherte Endanwendungen zu Problematiken innerhalb der Blöcke per se, aber auch der Chain als Gesamtes kommen [BSI].

Ein Distributed Ledger stellt im weitestgehenden Sinne eine Art Datenbank dar, welche auf allen teilnehmenden Nodes zeitgleich hinterlegt wird und alle Informationen des Netzwerkes enthält, insbesondere eine einheitliche Reihenfolge für verarbeitete Anfragen.

Somit hat jeder Netzwerkteilnehmer eine vollumfängliche Kopie dieser Datenbank. Im Falle eines neuen Nodes, wird vor der ersten Interaktion im Netzwerk selbst eine solche Kopie abgespeichert.

2.2 Crash Fault Tolerance vs. Byzantine Fault Tolerance

Die verschiedenen Fault Tolerance Kategorien beschreiben die Resistenz eines verteilten Computersystems gegenüber Fehlern, die absichtlich oder unabsichtlich von Teilnehmern des Netzwerkes verursacht werden.

Crash Fault Tolerance beschreibt hierbei den trivialen Fall, in dem ein verteiltes System das Ausfallen eines Knotens / Nodes keine Beeinträchtigung des Gesamtsystems verursacht. Sendet der Teilnehmer bedingt durch einen Programmierfehler oder in böswilliger Absicht falsche Daten ist der Gesamtzustand des Systems undefiniert.

Die von Leslie Lamport et al. beschriebene Allegorie des „Byzantine General Problem“ [LaShPe82] verdeutlicht die Problemstellung, in einem verteilten System mit asynchroner Kommunikation eine gemeinsame Entscheidung zu finden. Hier wird davon ausgegangen, dass nur bei einem gleichzeitigen Angriff verschiedener Generäle, die eine Stadt belagern, die Schlacht gewonnen werden kann.

Die Byzantine Fault Tolerance[1] Eigenschaft beschreibt also die Eigenschaft, dass ein verteiltes System auch bei Präsenz von einer definierten Anzahl fehlerhafter oder bösartiger Teilnehmer ein korrektes Ergebnis bzw. eine gemeinsame Entscheidung zu finden.

2.3 Konsensmechanismen

Konsensverfahren sind Algorithmen, die die Aufgabe besitzen, sicherzustellen, dass alle Nodes eines verteilten Computersystems untereinander einen gemeinsamen Beschluss fassen.

Unterschieden werden diese Mechanismen auf Grund ihrer Eigenschaften (bspw. Byzantine Fault Tolerance, Safety- und Liveliness Properties) und Anforderungen (z.B. an Netzwerklatenzen), sowie durch die eingesetzten Algorithmen.

Im Folgenden werden zwei weit verbreitete Algorithmen vorgestellt, die in den bekannten Kryptowährungen Bitcoin und Ethereum eingesetzt werden. Außerdem wird der Konsensmechanismus Paxos näher erläutert, der in abgewandelter Form die Grundlage für den Konsensmechanismus des vorliegenden Prototypen bildet.

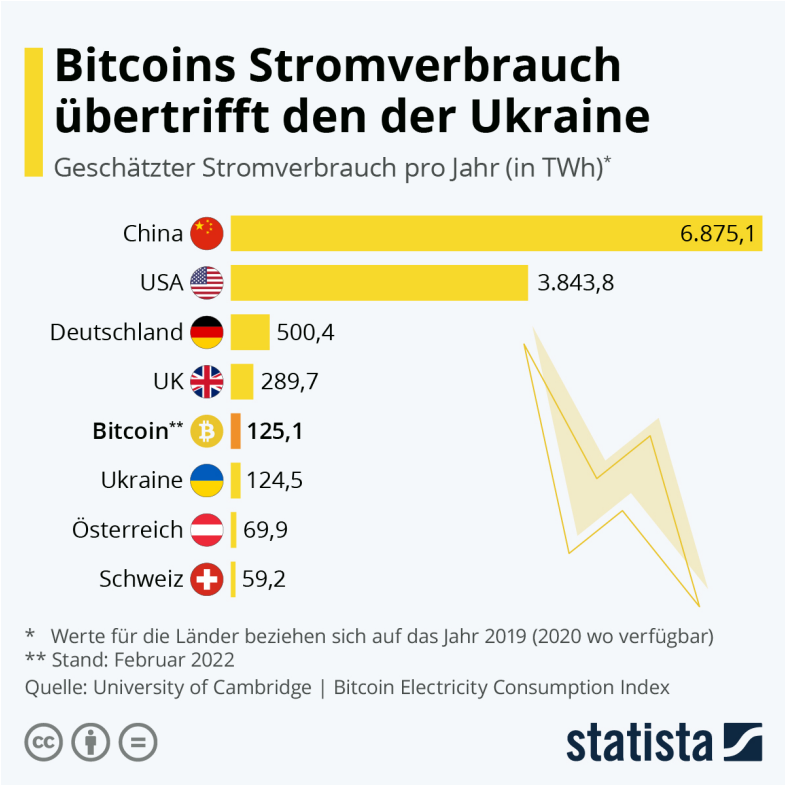
1 Byzantine Fault Tolerance ≡ BFT

2.3.1 Proof of Work

Proof of Work (PoW) ist ein Konsensmechanismus, der hauptsächlich in Blockchain-basierten Kryptowährungen wie Bitcoin verwendet wird. Algorithmen auf dieser Basis erfordern, dass jeder Teilnehmer, der einen Block in die Blockchain aufnehmen möchte, eine komplexe mathematische Aufgabe löst (in der Regel basieren auf Hashfunktionen). Dieser Vorgang wird auch als "Mining" bezeichnet wird. Diese Aufgabe ist so konstruiert, dass sie sehr rechenintensiv ist und es eine gewisse Zeit dauert, um sie zu lösen. Sobald ein Teilnehmer die Aufgabe erfolgreich gelöst hat, kann er den neuen Block in die Blockchain aufnehmen und erhält dafür einen Reward, in der Regel wieder in Form der jeweiligen Kyptowährung.

Der Zweck des PoW ist es, das Hinzufügen von Blocks zur Blockchain zu regulieren und Doppelausgaben zu verhindern, indem es sicherstellt, dass jeder Block, der in die Blockchain aufgenommen wird, von einem Teilnehmer erstellt wurde, der viel Rechenleistung und damit auch Kosten aufgewendet hat. Es ist auch eine Methode um die Sicherheit der Blockchain zu gewährleisten, da es theoretisch gesehen sehr teuer und schwierig ist, eine Mehrheit der Rechenleistung in einem PoW-Netzwerk zu kontrollieren, und dadurch die Mehrheit der Blöcke in der Chain zu manipulieren. Ein Angriff auf eine Mehrheit der Rechenleistung zur Manipulation des Konsens wird auch Sybil Attack genannt [Dou02].

Ein großter Nachteil der PoW basierten Konsenverfahren bildet sich in ihrem Stromverbrauch ab – durch die hohe Intensivität der benötigten Rechenleistung entsteht ein sehr hoher Energieverbrauch. So hat beispielsweise laut dem Bitcoin Electricity Consumption Index der University Cambridge das Bitcoin Netzwerk Anfang 2022 mehr Energie verbraucht als die gesamte Ukraine:



2.3.2 Proof of Stake (PoS)

Proof of Stake (PoS) ist ein Konsensmechanismus, der hauptsächlich in Blockchain-basierten Kryptowährungen verwendet wird. Im Gegensatz zu Proof of Work, bei dem Rechenleistung (meist durch Mining) investiert wird, um neue Blöcke in die Blockchain aufzunehmen, basiert PoS auf dem Besitz einer bestimmten Menge der Kryptowährung, die als "Stake" bezeichnet wird.

In einem PoS-System werden die Teilnehmer, die neue Blöcke erstellen dürfen, auf der Grundlage ihres Stake ausgewählt. Je größer der Stake eines Teilnehmers ist, desto höher ist die Wahrscheinlichkeit, dass er ausgewählt wird.

Wie bei PoW erhält der Teilnehmer, der einen neuen Block errechnet / erstellt eine Belohnung.

PoS stellt sicher dass Blöcke nur von Teilnehmern erstellt werden können, die eine signifikante Menge an Kryptowährung besitzen und „staken“, und somit ein Eigeninteresse hat das Netzwerk

und die Kryptowährung zu unterstützen. PoS ist außerdem energieeffizienter als PoW, da es keine hohen Rechenleistungen benötigt.

Allerdings eignet sich PoS nicht für verteilte Netzwerke, in denen keine Vorverteilung von Werten bzw. Währung besteht, oder keine Währungstransaktionen vorgesehen sind.

2.3.3 Paxos

Paxos wurde als Konsensmechanismus von Leslie Lamport entwickelt und 1998 erstmals in der ursprünglichen Version veröffentlicht. Es ist heute ein wichtiger Bestandteil von vielen verteilten Systemen wie Datenbanken und verteilten Dateisystemen.

Das Paxos-Verfahren besteht aus mehreren Phasen. Im Allgemeinen bzw. in der ursprünglichen Variante umfasst das Verfahren die Phasen Prepare, Promise, Accept und Response. In der Prepare-Phase schlägt ein Teilnehmer einen Wert vor, der in das System aufgenommen werden soll.

In der Promise-Phase stimmen andere Teilnehmer dem Vorschlag zu oder lehnen ihn ab. In der darauf folgenden Accept / Response Phase bestätigen die Teilnehmer daraufhin, dass sie den Wert akzeptiert haben und dass er nun Teil des Systems ist. [Lam98].

Generell bietet Paxos die Garantien, dass nur gültige Werte gelernt werden können (Trivial), dass gelernte Werte immer im System einheitlich sind (Agreement) und das System Fortschritt macht, so lange eine ausreichende Anzahl von Teilnehmern nicht fehlerhaft ist (Liveliness). [Lam06]

Ein wichtiger Aspekt von BFT Varianten von Paxos ist die Fähigkeit, mit fehlerhaften oder bösartigen Teilnehmern umzugehen, sowie mit Phasen in denen Nodes nicht oder nur mit hoher Latenz erreichbar sind.

Paxos existiert in verschiedene Varianten (z.B. Multi Paxos für mehrere Leader) und Erweiterungen des ursprünglichen Algorithmus. Der ausgewählte Konsensmechanismus für das vorliegende Projekt – SmartBFT nach [BaMaMeTo21] – basiert ebenfalls auf Paxos und stellt eine Weiterentwicklung auf Basis früherer Verbesserungen / Analysen dar. [CaLi99] [SoBe12]

2.4 Kryptographische Grundlagen

2.4.1 Public Key Infrastructure

Public Key Infrastructure (PKI) bezeichnet ein System, das verwendet wird, um digitale Zertifikate und öffentliche Schlüssel zu verwalten und zu verteilen. Mit Hilfe der Signaturen und Zertifizierungen von öffentlichen Schlüsseln kann im Zusammenspiel mit kryptographischen Protokollen sichergestellt werden, dass die Kommunikation über ein Netzwerk sicher und authentisch ist.

Mit Hilfe einer zentralen Zertifizierungsstelle (Certificate Authority), kann die Identität von Personen, Unternehmen oder Geräte überprüft werden, die ein Zertifikat beantragen. Bei erfolgreicher Prüfung des Certificate Signing Request (CSR) wird ein Zertifikat ausgestellt, das ihre Identität einem öffentlichen Schlüssel zuordnet.

Das Zertifikat kann unterschiedlich kodiert werden (üblich sind X.509, DER und PEM). Es enthält Informationen wie den Namen des Besitzers, mögliche Hostnamen und IP-Adressen, seine öffentliche Schlüssel und eine digital signierten Informationen über die Gültigkeit des Zertifikates. Das Zertifikat wird wiederum mit dem Schlüssel der Zertifizierungsstelle signiert, so dass eine Vertrauensstellung entsteht.

2.4.2 Transport Layer Security (TLS)

Das Transport Layer Security Protokoll (aktuell in Version 1.3) bietet einen standardisierten Weg, um einen abhörsicheren und manipulationssicheren Kommunikationskanal zwischen zwei Endpunkten über eine nicht vertrauenswürdige Verbindung herzustellen.

Es stellt den Nachfolger des Secure Socket Layer Protokolls (SSL) dar, und bietet mit definierten, sicheren Kombinationen von kryptographischen Primitiven (sogenannte Cipher Suites) zusammen mit einer PKI eine standardisierten sicheren Kommunikationskanals an.

Außerdem werden bei Einsatz der entsprechenden Versionen und Cipher Suites zusätzliche Eigenschaften wie zum Beispiel Forward Secrecy garantiert – eine nachträgliche Entschlüsselung des Netzwerkverkehrs wird damit auch bei einem kompromittierten privaten Schlüssel unmöglich. [RFC8446]

2.5 Elliptische Kurven

2.5.1 Grundlagen

Elliptic Curve Cryptography (ECC) ist eine Methode der Kryptographie, die auf der Mathematik von elliptischen Kurven in einem definierten (diskreten) Restklassenkörper basiert.

Grundlage für ECC bildet eine gemeinsam definierte (öffentliche) Kurve, sowie ein Punkt auf dieser Kurve als öffentlichen Schlüssel.

Um eine Nachricht zu verschlüsseln, wird sie mit dem öffentlichen Schlüssel multipliziert, um einen verschlüsselten Punkt zu erzeugen. Der Empfänger der Nachricht besitzt dann einen privaten Schlüssel, mit dem er den verschlüsselten Punkt entschlüsseln kann, indem er ihn mit dem privaten Schlüssel multipliziert.

Einer der Vorteile von ECC ist, dass es für dieselbe Sicherheit wie traditionelle Kryptographie-Methoden wie RSA, aber mit viel kürzeren Schlüssellängen arbeitet. Daher ist es für Anwendungen wie die sichere Übertragung von Daten in drahtlosen Netzwerken und die Verschlüsselung von mobilen Geräten geeignet, wo die Rechenleistung begrenzt ist. Es ist auch robust gegen Brute-Force Angriffe, da in der Regel größere Schlüssellängen benötigt werden um einen Schlüssel zu erraten.

Theoretische und praktische Grundlagen für den Einsatz von ECC finden sich unter anderem in [Wer13]. Im Folgenden werden kurz die in der vorliegenden Arbeit eingesetzten Verfahren vorgestellt.

2.5.2 EdDSA / Ed25519 Signaturverfahren

Das EdDSA Edwards Digital Signature Algorithm basiert auf Curve25519 (eine definierte elliptische Kurve) und bietet mit einer Abwandlung des Schnorrs Signatur Algorithmus (DSA) auf Basis einer elliptischen Kurve eine sichere und hochperformante Möglichkeit, beliebige Daten mit asymmetrischen Schlüsselpaar zu signieren. Die Signatur erfordert den privaten Schlüssel, während die Verifizierung mit dem öffentlichen Schlüssel erfolgen kann.

Ed25519 bietet ein Sicherheitsniveau von 128bit und wird dadurch (ausreichend große Quantencomputer ausgenommen) für traditionelle Kryptographie als sicher eingeschätzt. Die verwendeten Schlüsselpaare sind kleiner im Vergleich zu anderen Signaturmethoden.

[RFC8032]

2.5.3 Curve25519 / X25519 Schlüsselaustausch

Das X25519 Verfahren zum Schlüsselaustausch basiert ebenfalls auf Curve25519 (hier aber in der Montgomery Version). Es handelt sich um eine auf dieser elliptischen Kurve basierende Version des Diffie-Hellmann Protokolles zum Schlüsselaustausch, dass auch ohne elliptische Kurven verwendet werden kann (in einem klassischen Restklassenkörper).

Dabei wird unter Verwendung von privaten und öffentlichen Schlüsseln beider Parteien ein geheimer Schlüssel gebildet, der von einem passiven Angreifer im Netzwerk nicht ermittelt werden kann.

Im Vergleich zu anderen Implementierungen des Diffie-Hellmann Protokolls ist das Verfahren performanter und effizienter, sowie resistenter gegen kryptografische Side-Channel Angriffe.

[RFC7748]

2.5.4 Elliptic Curve Integrated Encryption Scheme (ECIES)

Es existieren verschiedene, divergierende Implementierungen bzw. Verfahren zur Verschlüsselung mit Hilfe von elliptischen Kurven im Rahmen einer hybriden Verschlüsselung, auch ECIES genannt. Verbreitet ist der Ansatz aus [ANSI-X9.63]:

Nach Definition der verwendeten elliptischen Kurve / Restklassenkörpers wird vom Absender ein temporäres Schlüsselpaar generiert. Analog zum Schlüsselaustausch bei X25519 wird durch Multiplikation des privaten, temporären Schlüssels mit dem öffentlichen Schlüssel des Empfängers ein geheimer symmetrischer Schlüssel generiert.

Der Absender verschlüsselt mit einem vorher definierten symmetrischen Verschlüsselungsverfahren (z.B. AES oder ChaCha20), idealerweise unter Einsatz eines Authenticatied Encryption (AE/AD) Cipher oder eines zusätzlichen Message Authentification Codes (MAC) und dem so generierten symmetrischen Schlüssel die Daten.

Der Absender erhält dann den öffentlichen temporären Schlüssel gemeinsam mit den verschlüsselten Daten, und kann diese Operation entsprechend umgekehrt wiederholen um die Daten mit seinem privaten Schlüssel zu entschlüsseln.

2.6 Netzwerkkommunikation

2.6.1 REST API

Representational State Transfer (REST) bezeichnet ein Paradigma dar, mit dem Application Programming Interfaces (APIs) für die Kommunikation zwischen Clients und Server auf Basis des Hypertext Transfer Protocols (HTTP) entwickelt werden können.

Anfragen über das HTTP-Protokoll (wie GET, POST, etc.) können damit an einen Server gestellt werden, um Daten abzufragen oder zu ändern.

Eine REST API besteht aus einer Reihe von Endpunkten, die über einen Uniform Resource Locator (URL) zugänglich sind. Jeder Endpunkt kann dazu verwendet werden, bestimmte Funktionen auf dem Server aufzurufen.

Beispielsweise kann ein Endpunkt verwendet werden, um Daten aus einer Datenbank abzufragen, während ein anderer Endpunkt verwendet werden kann, um Daten in die Datenbank einzufügen.

REST APIs sind damit sehr flexibel und können von den meisten gängigen Programmiersprache und Betriebssystemen aus verwendet werden. Ein wichtiger Bestandteil von REST APIs ist die semantische Verwendung von HTTP Methoden und Statuscodes. Diese Codes signalisieren den Erfolg oder Misserfolg einer Anfrage und geben Auskunft über den Zustand einer Anfrage.

Beispielsweise gibt ein HTTP-Statuscode 200 OK an, dass die Anfrage erfolgreich war, während ein HTTP-Statuscode 404 Not Found anzeigt, dass die angeforderte Ressource nicht gefunden wurde.

[Fie20]

2.6.2 Protocol Buffers & gRPC

Protocol Buffers (auch bekannt als Protobuf) ist ein Open Source Austauschformat / Serialisierungsformat für beliebige strukturierte Daten und umfasst zugehörige Libraries bzw.

Plugins für diverse Programmiersprachen. Das Format wurde von Google entwickelt, um effizienter und plattformübergreifender Datenstrukturen und -services zu erstellen und zu übertragen.

Es ermöglicht strukturierte Daten in ein binäres Format zu kodieren und zu dekodieren, um diese kompakt und mit breiter Kompatibilität zu übertragen. Die so kodierten Daten sind damit kompakter und effizienter als Textformate wie XML oder JSON. Aber nicht menschenlesbar ohne die Verwendung einer entsprechenden Bibliothek.

gRPC ist ein Open Source Remote Procedure Calls (lRPCs), das ebenfalls von Google mit Fokus auf Performance und Sicherheit entwickelt wurde. Es basiert auf Protocol Buffers als Austauschformat (kann aber auch andere Formate unterstützen) und ermöglicht es, RPC Services auf verschiedenen Plattformen und mit verschiedenen Programmiersprachen zu erstellen und aufzurufen.

Mit gRPC können entfernte Funktionen / Prozeduren mithilfe von Service- und Methodenbeschreibungen im Protobuf Format aufgerufen werden. Serialisierung und Deserialisierung erfolgt dabei automatisch.

2.6.3 mDNS / DNS Service Discovery (DNS-SD)

mDNS (Multicast Domain Name System) ist ein Protokoll, das es ermöglicht, Domain Name System Anfragen im selben lokalen Netzwerk (Multicast Domäne) ohne einen zentralen DNS-Server aufzulösen. Multicasts werden in der Regel an Netzwerkgrenzen gefiltert, wodurch sich dieses Protokoll nur für lokale Netzwerke eignet.

DNS-Based Service Discovery (DNS-SD) ermöglicht es, Dienste anhand spezieller DNS Anfragen zu finden. Dienste können dabei nach Instanznamen, Dienstnamen und lokale Domäne eingeordnet und mit zusätzlichen Informationen in Form von TXT Records versehen werden. Die Auflösung erfolgt inklusive Port und Netzwerkprotokoll (UDP, TCP).

In Kombination ermöglichen es die beiden Protokolle, Geräte automatisch und ohne manuelle Konfiguration anhand ihrer Hostnamen und bereitgestellten Dienste aufzufinden.

[RFC6762] für Multicast DNS & [RFC6763) für DNS-SD

3 Praktische Realisierung

3.1 Resultate

- Erfolgreiche Entwicklung eines Prototypen für ein verteiltes System mit Go (golang.org), inkl. Integrationstests mit fehlerhaften & ausgefallenen Nodes
- Basierend auf der Library für den Konsensalgorithmus *SmartBFT*[2], von IBM entwickelt als Byzantine Fault Tolerant Konsensmechanismus für HyperLedger Fabric (kein Einsatz von Fabric im Projekt).
- Entwicklung eigener Certificate Authority (P2P Zertifikate mit Ed25519 Keys)
  - Verwendung eines zentralen Schlüsselpaars für Signaturen im Konsensverfahren sowie Verschlüsselung mit ECIES
- Netzwerkkommunikation und Remote Procedure Calls mit gRPC
  - Gesicherter Austausch von Nachrichten für Konsensverfahren (Stream)
  - Gesicherter Austausch von Nachrichten für Fabrikationsdaten (Stream)
- Node Discovery statisch (CLI) oder im LAN mit mDNS & DNS-SD
- Webinterface mit REST API
  - Statische HTML / JS Anwendung basierend auf Bootstrap 5
  - Auslieferung an Client integriert in API HTTP Server

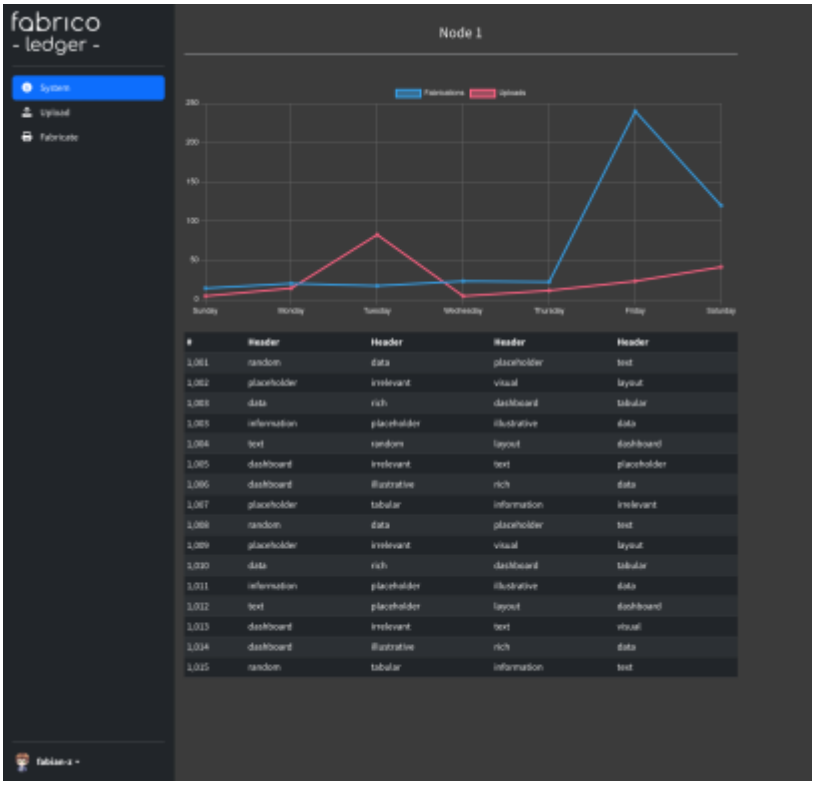
3.2 Webbasiertes User Interface

Für die Webentwicklung eines User-Interfaces wurde eine statische Hypertext Markup Language[3], sowie JavaScript Anwendung implementiert, welche auf Bootstrap 5 basiert. Die Kommunikation mit dem verteilten System erfolgt via lokaler HTTP Verbindung mit dem zugrundeliegenden Node Prototypen via REST API.

2 Siehe: https://github.com/SmartBFT-Go/consensus

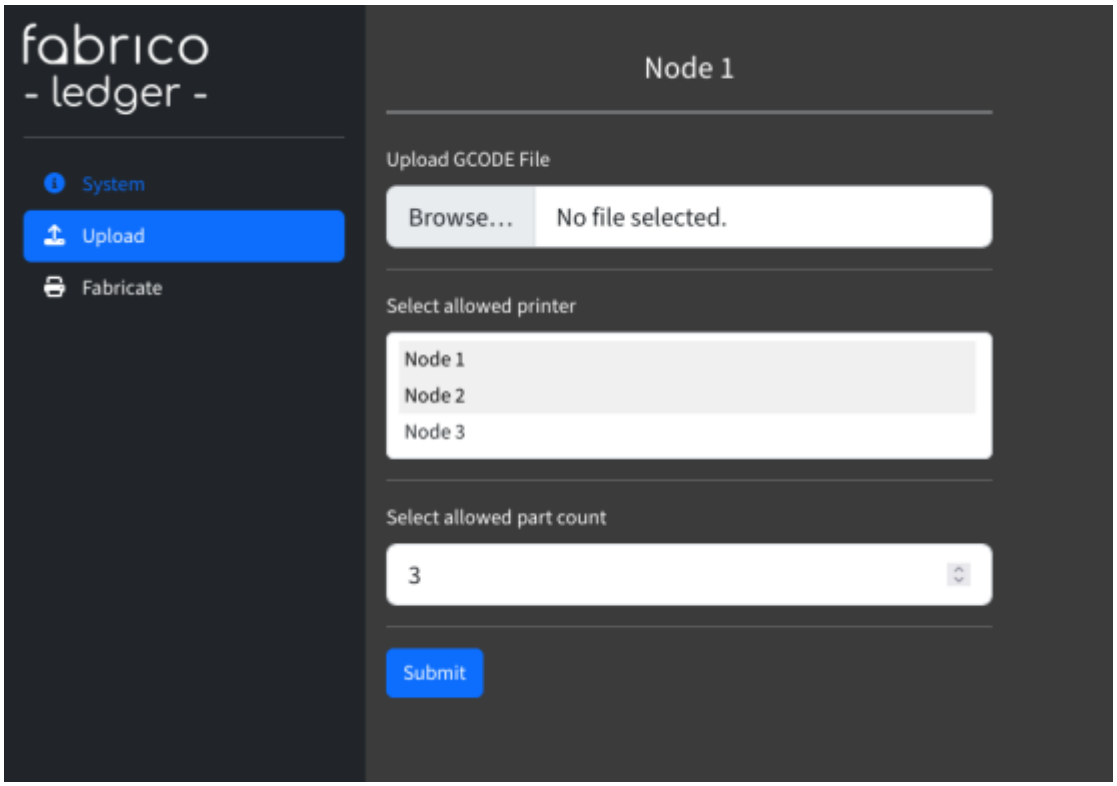
3 Hypertext Markup Language ≡ HTML

In der Systemstatus Ansicht (Abbildung 1) kommt Chart.JS zum Einsatz, um zukünftig Kennzahlen des aufgebauten verteilten Netzwerkes zu visualisieren. Im vorliegenden Prototypen ist die Visualisierung mit statischen Testdaten ausgeführt.

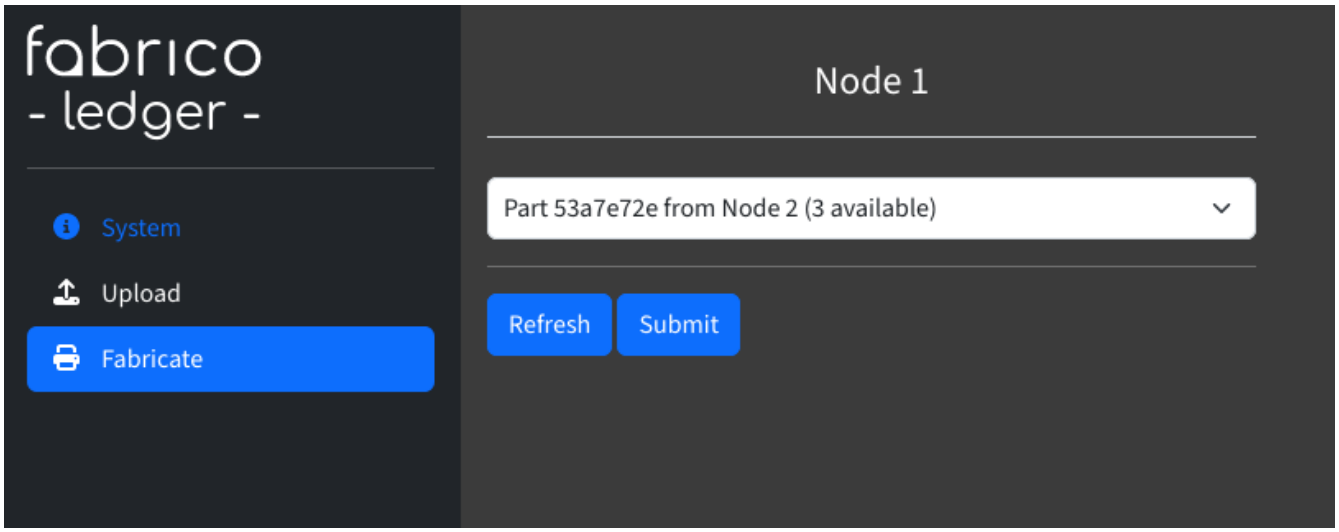


Das User Interface ermöglicht es dem Bediener, über ein nutzerfreundliches „Upload“ Interface Fertigungsdaten hinzuzufügen und für bestimmte Nodes zu lizenzieren (Auswahl von Nodes und Anzahl der erlaubten Produktionen).

Im vorliegenden Prototypen werden die Fertigungsdaten lokal auf dem ursprünglichen Node gespeichert und bei Bedarf über eine GRPC Schnittstelle zur Verfügung gestellt.



Das „Fabrication“ Interface erlaubt eine Anzeige der aktuell für den verwendeten Node lizenzierten Produktionsdaten sowie die Anzahl der verbleibenden Produktionen / Verwendungen. Nach Auswahl können diese Produktionsdaten direkt an das konfigurierte computergestützte Fertigungssystem versendet werden.



4 Ausblick

- Node Discovery im WAN via Distributed Hash Tables (Vgl. BitTorrent)
- Ausarbeitung Zertifikatsprüfung (Dynamische Zuordnung ohne Hostname Abhängigkeit)
- Ausarbeitung Webbasiertes User Interface
- Interface für die Datenlogistik
  - Konzeptioneller Ersatz einer physisch niedergelassenen Logistik
  - Einsatz von ECIES zur Sicherstellung der Vertraulichkeit und Lizenzierung
- Überwachung des laufenden Fertigungsprozesses ermöglichen (Abhängig von spezifischer Fertigungsweise)

5 Quellen- und Literaturverzeichnis

Alle Internetquellen – sofern nicht anders angegeben – verstehen sich als zuletzt abgerufen / aktualisiert am 15.01.2023.

Kürzel	Quelle / Literatur
[BSI]	BSI (o.D.): *Blockchain macht Daten praktisch unveränderbar*, in: https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Informationen-und-Empfehlungen/Technologien_sicher_gestalten/Blockchain-Kryptowaehrung/blockchain-kryptowaehrung_node.html
[LaShPe82]	Lamport, Shostak, Pease (1982): The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems. **4** (3): 382–401. https://doi.org/10.1145%2F357172.357176
[Tran2017]	Tran, Alexandra (2017): *An introduction to byzantine fault tolerance and alternative consensus*, in: https://ancapalex.medium.com/a-cursory-introduction-to-byzantine-fault-tolerance-and-alternative-consensus-1155a5594f18
[Dou02]	Douceur, John R (2002): "The Sybil Attack" Peer-to-Peer Systems. Lecture Notes in Computer Science. Vol. 2429. pp. 251–60. https://www.doi.org/10.1007/3-540-45748-8_24
[Lam98]	Lamport, Leslie (1998): The Part-Time Parliament. ACM Transactions on Computer Systems. 133 – 169. https://doi.org/10.1145%2F279227.279229
[Lam06]	Lamport, Leslie (2006): Fast Paxos, Distributed Computing 19, 2. 79-103
[BaMaMeTo21]	Barger, Manevich, Meir, Tock (2021): A Byzantine Fault-Tolerant Consensus Library for Hyperledger Fabric https://arxiv.org/abs/2107.06922, sowie https://github.com/SmartBFT-Go/consensus/wiki/Algorithm
[CaLi99]	Castro, Liskov (1999): Practical Byzantine Fault Tolerance http://pmg.csail.mit.edu/papers/osdi99.pdf
[SoBe12]	Sousa, Bessani (2012): From Byzantine Consensus to BFT State Machine Replication: A Latency-Optimal Transformation http://www.di.fc.ul.pt/~bessani/publications/edcc12-modsmart.pdf
[Wer13]	Werner, Annette (2013): Elliptische Kurven in der Kryptographie

	Springer Verlag, ISBN 978-3540425182
[RFC8032]	Internet Research Task Force (2017): RFC8032 / Edwards-Curve Digital Signature Algorithm (EdDSA) https://www.rfc-editor.org/rfc/rfc8032
[Val19]	Filippo Valsorda (2019): Using Ed25519 signing keys for encryption https://words.filippo.io/using-ed25519-keys-for-encryption/
[RFC7748]	Internet Research Task Force (2016): Elliptic Curves for Security https://www.rfc-editor.org/rfc/rfc7748.html (Curve25519, X25519 with Montgomery Form)
[ANSI-X9.63]	American National Standards Institute (2001): "ANSI - X9.63 Public Key Cryptography for the Financial Services Industry Key Agreement and Key Transport Using Elliptic Curve Cryptography"
[Fie20]	Fielding, Roy Thomas (2000). "Representational State Transfer (REST)". Architectural Styles and the Design of Network-based Software Architectures http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
[RFC8446]	Internet Engineering Task Force (2018): The Transport Layer Security (TLS) Protocol Version 1.3 https://www.rfc-editor.org/rfc/rfc8446
[RFC6762]	Internet Engineering Task Force (2013): Multicast DNS https://www.rfc-editor.org/rfc/rfc6762.html
[RFC6763]	Internet Engineering Task Force (2013): DNS-Based Service Discovery https://www.rfc-editor.org/rfc/rfc6763 (DNS-SD)