Transaktionen

Ture Claußen, 1531067, ture.claussen@stud.hs-hannover.de und Jannes Neemann, 1530893, jannes.neemann@stud.hs-hannover.de

Fakultät IV, Abteilung Informatik, Hochschule Hannover, Ricklinger Stadtweg 120, 30459 Hannover

Zusammenfassung. Schlüsselwörter:

Inhaltsverzeichnis

1 Einführung 2
2 Struktur und technische Umsetzung einer Transaktion 3
2.1 Komponenten einer Transaktion 3
2.2 Typen von Transaktionen 3
2.3 Serialisierung 4
3 Aufbau einer Transaktion 4
3.1 Nonce 4
3.2 Gas 4
3.3 Value und Data 6
3.4 Signatur 6
4 Transaktionsabwicklung 7
4.1 Propagation 7
4.2 Speicherung 7
5 Ausblick 7

1 Einführung

Das Wort Transaktion stammt von dem lateinischen Wort *transigere* ab, welches im übertragenden Sinne mit 'durchführen', 'vollführen' oder 'abmachen' (Geschäft) übersetzt werden kann. [1] Dieser Wortsinn besteht auch weiterhin im technischen und wirtschaftlichen Bereich, jedoch gibt es noch spezifischere Abgrenzungen. In der Wirtschaft ist es ein Vorgang bei dem Waren und Forderungen ausgetauscht werden. [5, S. 18 f.] In der Informatik ist es im Zusammenhang mit Datenbanken eine unteilbare, *atomare*, Abfolge von Anweisungen, die einen Übergang von einem konsistenten Zustand in einen Anderen beschreibt. [6, S.520]

Ethereum ist ein "transaktionsbasierter Automat"(*transaction-based state machine*). Somit sind Transaktionen ein grundlegender Baustein von Ethereum im Allgemeinen und ihnen kommt eine ähnliche Bedeutung wie ACID Transaktionen bei. Der Automat speichert seinen Zuständ σ_t in der Blockchain, eine Transaktion T ist Argument der Zugstandsübergangsfunktion Υ, die von *externen Akteuren (EA)* angestoßen wird und diesen gespeicherten Zustand σ_t in einen neuen, gültigen Zustand σ_{t+1} überführen soll: $\sigma_{t+1} = \Upsilon(T, \sigma_t)$. Im Falle eines Konsens des Netzwerkes wird diese Zustandsveränderung durchgeführt beziehungsweise gespeichert.

Im Kontrast zu Kryptowährungen wie Bitcoin ist der Umfang des Automaten bzw. des Protokolls bei Ethereum deutlich geweitet, denn Zweck ist nicht nur die Schöpfung, Speicherung und der Austausch eines digitalen Zahlungsmittels [8], sondern eine allgemeine dezentrale Rechenmaschine, ein "Weltcomputer". Daher bestehen bei Ethereum auch an Transaktionen andere technische und konzeptionelle Anforderungen, die im Folgenden erläutert werden. [10, S. 1-4]

2 Struktur und technische Umsetzung einer Transaktion

Die Komponenten welche eine Transaktion in Ethereum ausmachen sind vergleichbar mit denen eines Briefes. Sie besitzen einen Empfänger sowie eine Frankierung, welche die Transportkosten zum Empfänger bezahlen. In einen Brief kann man z.B. Geld oder einen Text verschicken. Auch Transaktionen in Ethereum können Geld in Form von Ether und einen Text in Form von Nutzdaten verschicken. Im Weiteren wir die allgemeine Struktur und technische Umsetzung einer Transaktion vorgestellt.

2.1 Komponenten einer Transaktion

Transkationen enthalten laut ihrer offiziellen Definition [10, S. 4] folgende Datenfelder:

nonce: Ein Skalar welcher gleich der Anzahl der vom EOA versendeten Transaktionen ist. Der Nutzen wird in 3.1 erläutert.

gasPrice: Ein Skalar der angibt, wie viel Wei man pro Einheit *Gas* bezahlt, die bei der Gesamtheit aller Berechnungen die während der Ausführung der Transaktion anfallen (S. 3.2)

gasLimit: Ein Skalar der die maximal Anzahl an *Gas* angibt, die während der Ausführung der Transaktion verbraucht werden darf. Dieser Betrag muss im Voraus bezahlt werden.

to: Die 160-Bit Adresse des Empfängers.

value: Skalar der die Menge Wei angibt, die der Empfänger erhält.

v,r,s: Komponenten der ECDSA-Signatur (S. 3.4), um den Sender der Transaktion zu bestimmen

init: Ein Byte-Array unbegrenzter Länge, welches nur bei einer Kontrakterzeugung verwendet wird und den kompilierten Sourcecode des Kontrakts enthält

data: Ein Byte-Array unbegrenzter Länge, welches die Nutzdaten des Kontrakts enthält

2.2 Typen von Transaktionen

Es gibt genau zwei Typen von Transaktion. Transaktionen die eine Nachricht von einem Account[1] zu einem anderen überträgt („message calls“ [10, S. 4]) oder Transaktionen die einen neuen Kontrakt erzeugen („contract creation“ [10, S. 4])). Mit Nachricht ist dabei der Inhalt von den Feldern *value* und *data* gemeint.

Bei massage call Transaktionen enthält das *to* Feld die öffentliche Adresse eines EOA oder eines Kontrakts. Zusätzlich besteht die Option *value* und *data* zusetzen. Speziell ist das *data* Feld von Bedeutung, wenn das Transaktionsziel ein Kontrakt ist, denn in diesem Feld wird der konkrete Funktionsaufruf in Bytecode gespeichert und wird von dem Kontrakt ausgeführt.

[1] Mit Account ist hier ein EOA oder ein Kontrakt gemeint

Die Besonderheit bei contract creation Transaktionen ist, dass die Empfängeradresse die Nulladresse (*0x0*) ist. Diese Adresse ist keinem Account zugewiesen und dient ausschließlich als „kontrakterzeugungs Adresse“ [3].

Ein spezieller Typ von Transkation ist eine interne Transaktion. Diese treten nur ausgehend von einem Kontrakt aus auf. Sie werden auch nicht in der Blockchain aufgefasst. Beispielsweise kann eine Transkation die an einen Kontrakt gerichtet ist, eine Funktion aufrufen, welche dem Sender einen Etherbetrag zurücksendet. Diese eigentlich Transaktion wird als interne Transaktion gewertet und nur der Funktionsaufruf wird in der Blockchain dokumentiert.

2.3 Serialisierung

Da Ethereum ein Weltcomputer ist und Daten somit über die ganze Welt verschickt werden, müssen diese kompakt, effizient und einheitlich verschickt werden. Dabei wird das Kodierungsverfahren *Recursive Length Prefix (RLP)* verwendet.Es ist ein relativ simples und deterministisches Kodierungsverfahren, dessen einzige Aufgabe es ist Struktur zu kodieren. Mit Struktur sind (verschachtelte) Arrays von binären Daten gemeint. Die Methode nimmt dabei genau einen Parameter entgegen ein sogenanntes „Item“. Ein Item ist ist dabei ein String oder eine Liste von Items. Wichtig bei der Kodierung ist die Länge des Inhalts, denn je nach Länge gibt es bestimmte Regeln. In jeder Regel wird spezifiziert, welche zusätzlichen Bytes gesetzt werden. So wird zum Beispiel ein Byte, welches den Wert zwischen *0x00* und *0x7f* hat nicht verändert und das Byte ist das RLP Encoding des Bytes. Ist das Item eine Zeochenlägen mit einer Länge von 0-55 Byte besteht das RLP kodierte Resultat aus dem Byte *0x80* plus die Länge des Strings gefolgt von den Zeichen des Strings. Zum Beispiel würde *Ether* kodiert zu *[0x85, 'E', 't', 'h', 'e', 'r']* werden bzw. *[0x85, 0x45, 0x74, 0x68, 0x65, 0x72]*, wenn man die Zeichen in ihren ASCII Hex-Code übersetzt.

3 Aufbau einer Transaktion

3.1 Nonce

3.2 Gas

Gas ist ein zentraler konzeptioneller Lösungsansatz im Rahmen von Ethereum. Da Ethereum turing-vollständig ist [10, S. 1], ergibt sich unter anderem das sogenannte "Halteproblem". Dieses besagt, dass im Voraus nicht vorhergesagt werden kann, ob das Programm einer Turing-Maschine jemals zu einem Ende kommt. [4, S.70] Um die Funktionalität des Netzwerks zu gewährleisten, wird die Laufzeit einer jeden Zustandsveränderung der Blockchain, sprich Transaktion, durch Gas begrenzt.

Gas ist eine eigenständige Währung innerhalb von Ethereum, dessen Einheit einen Rechenschritt in der EVM bemisst [7, S. 9:3], wobei für jeden Opcode die Kosten in Gas spezifiziert werden. [10, S. 25 ff.] Gas ist also eine Gebühr

für Rechnenaufwand. Zusätzlich werden auch Kosten für die Nutzung von persistentem Speicher miteinbezogen. Es gilt sogar das Inverse: Wird durch eine Transaktion persistenter Speicher freigegeben, werden Rabatte gewährt.

Die maximale Gebühr einer Transaktion wird durch die Kombination der Datenfelder *gasPrice* und *gasLimit* angegeben. Die resultierende Gebühr *gasPrice*$\times$ *gasLimit* wird bei Erstellung der Transaktion in voller Höhe vom Konto abgezogen. Nach Bestätigung der Transaktion wird nicht genutztes Gas zu dem angegebenen Preis in Ethereum zurückerstattet.

Somit gilt es im Voraus abzuschätzen wie hoch der Rechenaufwand sein wird. Je mehr Ressourcen des Weltcomputers in Anspruch genommen werden, desto höher die Gebühr. Gerade wegen des Halteproblems kann dies aber nur grob vorgenommen werden, eine robuste Programmierung von *Smart Contracts* ist essentiell. Ein erster Anhaltepunkt dafür sind zunächst die intrinsischen Kosten einer Transaktion. Das ist der Overhead der allein durch die Transaktion und deren Inhalt betsteht. Diese intrinsischen Kosten g_0 lassen sich mit auf Basis folgender Grundlage berechnen.

$$g_0 \equiv \sum_{i \in T_i, T_d} \begin{cases} G_{datazero} \text{ if } i = 0 \\ G_{txdatanonzero} \text{ otherwise} \end{cases} + \begin{cases} G_{txcreate} \text{ if } T_t = \emptyset \\ 0 \text{ otherwise} \end{cases} + G_{transaction}$$

Also steigen die Kosten einer Transaktion mit der Größe des Feldes *data* an und $G_{transaction}$ bestimmt den Basiswert an Gas für eine Transaktion, welcher sich im Jahr 2020 auf 21000 beläuft. Generell sollte das *gasLimit* tendenziell zu hoch angelegt sein, da Transaktionen mit unzureichendem Gas einfach abgebrochen werden *(out-of-gas Exception)*. In diesem Fall wird keine der begonnenen Veränderungen am Zustand gespeichert.

Preis und Latenz Gas kann bewusst nur mit Ether erworben werden, da die Gas-Preise möglichst unabhändig von den Preisschwankungen (von Ether) sein sollen. Der *gasPrice* kann frei gesetzt werden, auch ein Wert von 0 ist gültig. Ein Richtwert für den Wert lässt sich durch Werkzeuge wie ETH Gas Station ermitteln, welche vergangene Transaktionen im *Ledger* betrachten und daraus Richtwerte ermitteln.

Dort wird auch ein Umstand kenntlich, denn die Höhe des Gas-Preises scheint maßgeblich über die Latenz zu entscheiden, also die Zeit bzw. Zahl der Blöcke zwischen Veroeffentlichung einer Transaktion und ihrer Inkludierung in einem Block. Übersteigt der *gasPrice* das Mittel der anderen Transaktionen im *mempool* so steigt die Wahrscheinlichkeit in nächsten Block bearbeitet zu werden. Diese Korrelation schwindet allerdings, sobald die Durchsatzfähigkeit des Netzwerkes erreicht ist. Unter Betrachtung aller Transaktionen im Zeitraum vom 01.03.2020 00:00:17 UTC (Block 9581792) bis 31.03.2020 23:59:57 UTC (Block 9782601) mit dem Python Werkzeug *ethereum-etl* [2] ergibt sich aktuell folgender Durchsatz T_{max} pro Block: [9]

$$T_{max} = \frac{blockGasLimit}{transactionMedianGas} = \frac{9817880}{80000} = 122.72$$

Bei kurzzeitig stark erhöhter Anzahl an Transaktionen wie beispielsweise bei einem *ICO*, werden teilweise um ein vielfaches höhere Transaktionskosten gezahlt, um möglichst schnellen Zugriff auf die Wertanlagen zu erhalten. [7, S. 9:6 f.] Da das *gasLimit* H_1 eines Blocks nur durch die Miner nach erfolgreichem Schürfen eines Blockes das alte Limit um maximal $P(H)_{H1}$ um $\frac{P(H)_{H1}}{1024}$ erhöht oder verringert werden. Dies soll eine Zentralisierung der Rechenleistung auf weniger Miner verhindern. Gleichzeitig limitiert dies die Fähigkeit viele Transaktionen in einem kurzen Zeitintervall zu verarbeiten. Es zeigt sich außerdem, dass sich

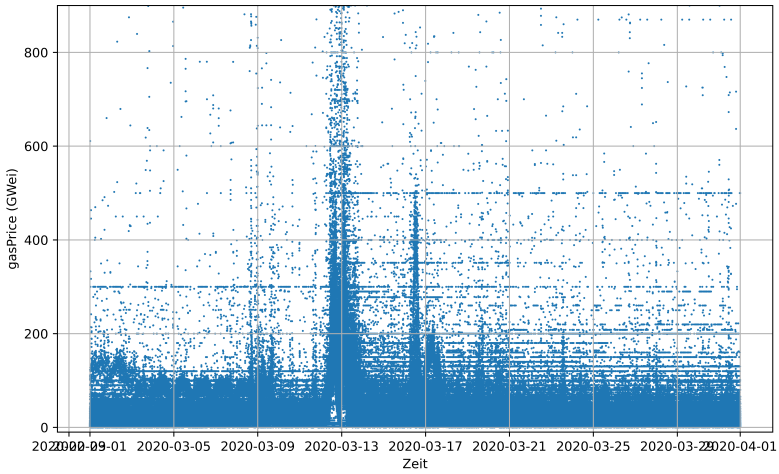

Abb. 1. gasPrice nach Tag im Monat März [9]

Anreiz und Spieletheorie

3.3 Value und Data

3.4 Signatur

ECDSA

Multisignaturen

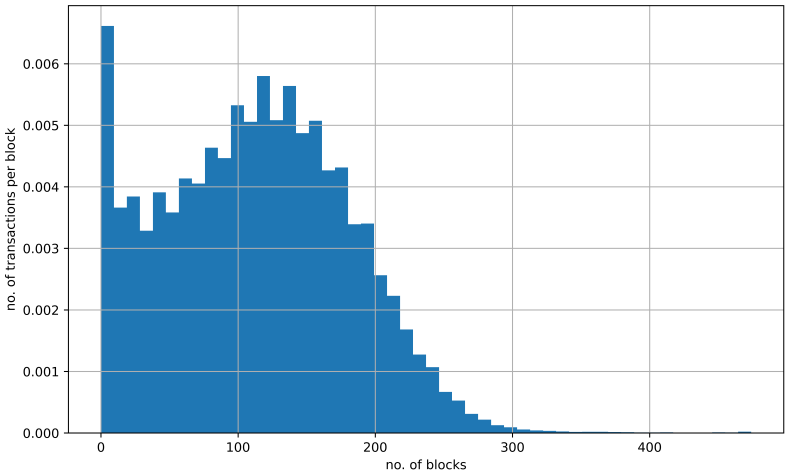

Abb. 2. Verteilung der Zahl an Transaktionen pro Block [9]

4 Transaktionsabwicklung

4.1 Propagation

4.2 Speicherung

5 Ausblick

Literatur

1. Transigere - Translation from Latin into German | PONS. https://en.pons.com/translate/latin-german/transigere
2. Blockchain-etl/ethereum-etl. Blockchain ETL (Apr 2020)
3. Antonopoulos, A.M., Wood, G.: Mastering Ethereum: building smart contracts and DApps. O'Reilly, Sebastopol, CA, first edition edn. (2019), oCLC: ocn967583559
4. Davis, M.: Computability and Unsolvability. Courier Corporation (Apr 2013)
5. Ehrlicher, W.: Kompendium der Volkswirtschaftslehre. Vandenhoeck & Ruprecht (1975)
6. Herold, H., Lurz, B., Wohlrab, J., Hopf, M.: Grundlagen Der Informatik. Pearson, third edn. (2017)
7. M.Spain, M.Foley: OASIcs-Tokeneconomics. Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany (2019)
8. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System
9. Neemann, J., Claussen, T.: Appendix: Scripts. https://github.com/campfireman/SEM-ethereum-transactions, library Catalog: github.com
10. Wood, G.: Ethereum/yellowpaper. 2019-10-20 (Oct 2019)