FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Analýza protokolu IOTA a jeho případy užití
Student:	Bc. Pavel Beran
Vedoucí:	Ing. Petra Pavlíčková, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem práce je analyzovat protokol IOTA včetně jeho výhod a nevýhod se zaměřením především na oblast Internet of Things. Současně s tím zanalyzovat možné případy užití protokolu a služby, které díky němu mohou vzniknout. V poslední řadě implementovat jádro jednoho z probraných případů užití.

1) Zanalyzujte Internet of Things, jeho dopady a problémy, se kterými se potýká.
2) Zanalyzujte protokol IOTA a vysvětlete hlavní podstatu jeho fungování.
3) Zhodnoťte vhodnost protokolu IOTA pro využití v rámci Internet of Things, vzhledem k jeho specifickým požadavkům na technologie.
4) Prozkoumejte možné případy užití protokolu IOTA, případně v kombinaci s IoT a obchodní modely a aplikace, které díky nim mohou vzniknout.
5) Zanalyzujte možnosti vývoje pomocí protokolu IOTA.
6) Vyberte jednu uvedenou aplikaci a technicky ji zanalyzujte a implementujte jádro daného systému s použitím protokolu IOTA.
7) Implementaci ekonomicky zhodnoťte a doporučte další vývoj/postup.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 11. prosince 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Analýza IOTA protokolu a jeho případy užití

Bc. Pavel Beran

Katedra Softwarového Inženýrství
Vedoucí práce: Ing. Petra Pavlíčková, Ph.D.

29. dubna 2019

Poděkování

Mé poděkování patří paní Ing. Petře Pavlíčkové, Ph.D. za konzultace při zpracovávání této práce a za její celkové vedení.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 29. dubna 2019 .....

Odkaz na tuto práci

Beran, Pavel. *Analýza IOTA protokolu a jeho případy užití.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Internet věcí je jedním z moderních konceptů, který však stále hledá technologie, o které by se v budoucnu opíral. Důvodem jsou především jeho rozsáhlé požadavky, které na dané technologie klade. Jednou z těchto technologií, které si kladou za cíl stát se standardem v rámci Internetu věcí, je protokol IOTA. Jedná se o jednu z implementací distribuované účetní knihy. V rámci práce je nejprve představena samotná technologie a koncepty, na kterých je založena. Následně je protokol analyzován z hlediska použitelnosti v rámci Internetu věcí a to především z hlediska specifických vlastností, které by měl splňovat. Zároveň jsou představeny nové obchodní modely a aplikace, které díky vlastnostem protokolu mohou vzniknout. Jedna z těchto aplikací je následně implementována a tato implementace je využita ke zhodnocení možností vývoje nad danou technologií. Práce tedy představuje nový protokol IOTA a hodnotí jej jako technologii pro IoT, projekt jako celek a zároveň jako technologii určenou pro stavbu dalších aplikací.

Klíčová slova IOTA protokol, analýza, distribuovaná účetní kniha, Internet věcí, Node.js

Abstract

The Internet of Things is one of the modern concepts that are still searching for its future main technologies. The reason for this are mainly its extensive technological requirements. One of these technologies that aims to become one of the IoT standards is IOTA protocol. It is one of the implementations of distributed ledger. In the first part this thesis presents the protocol itself and the main technologies and concepts it is based on. After that the protocol is analysed, especially its usability for the Internet of Things due to its specific requirements. The thesis also presents some new business models and applications that can be created thanks to the protocol. One of these applications is implemented in the last part and the implementation serves as a basis for the analysis of the development upon the protocol. Overall the thesis presents the IOTA protocol and analyses it as a new technology for IoT, as a project itself and also as a technology intended to be used for application development.

Keywords IOTA protocol, analysis, distributed ledger, Internet of Things, Node.js

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Internet se od svého vzniku v šedesátých, či sedmdesátých letech (názory na rok vzniku internetu se často různí, o čemž se zmiňuji v kapitole o stručné historii internetu) poměrně zásadně vyvinul. Z jednoduché komunikační sítě pro vyvolená zařízení se stal hromadně používanou technologií, bez které by si velká část populace nedokázala život představit.

V současné době je na internetu více než 4 miliardy uživatelů a denně je na internetu vyprodukováno více než 2,3 miliardy gigabytů dat. Tento objem dat s každým dalším rokem roste a tempo růstu se zrychluje. Toto je hlavním důvodem raketového vzestupu oborů jako jsou obory zabývající se analýzou a zpracováním Velkých dat (Big data - datové soubory takové velikosti, která neumožňuje jejich zpracování v normálním čase), nebo například strojového učení. Vzestup těchto oborů nevychází, jak se občas mylně domnívá, z růstu výpočetní kapacity hardware. Ten se naopak postupně zpomaluje a Moorův zákon již několik posledních let neplatí. Vychází ale hlavně právě z exponenciálního růstu vyprodukovaných dat ve světě. Díky tomu se stroje mohou učit na velkém počtu ukázkových dat a tím se efektivně zlepšovat. Stále však nejsou kapacity na efektivní zpracování všech produkovaných dat a reálně využitých je z tohoto důvodu ve výsledku pouhý zlomek.

S internetem a produkovanými daty je úzce spojen obor informačních technologií, který se stal velkým fenoménem posledních několika let a to Internet věcí (Internet of Things). Základní myšlenkou IoT je propojení fyzických zařízení pomocí jedné sítě, nebo skupiny více propojených sítí. Zařízení potom budou moci komunikovat, vyměňovat si data a na jejich základě provádět určité úkony. Podle odhadů by mělo být v roce 2020 v IoT připojeno kolem 30 miliard zařízení. Budoucím cílem je to, aby prakticky každá věc byla připojena do IoT, sbírala data o sobě a o okolí a tyto data poté někam odesílala, případně přijímala data a na jejich základě se rozhodovala, co dělat. Z toho je očividné, že v dohledné budoucnosti budou internetem, případně dalšími sítěmi, proudit obrovské objemy dat, nesrovnatelné s tím, co jimi proudí nyní. Toto je také

jedna z překážek, s kterými se IoT potýká. Aktuální standardní komunikační technologie nejsou schopné efektivně zvládat tyto objemy dat. A toto je pouze jeden z problémů, který musí IoT do budoucna nějakým způsobem vyřešit, pokud má být dosaženo toho, co se očekává.

Vedle IoT je dalším fenoménem poslední doby blockchain [1] a další jiné implementace decentralizovaných účetních knih. Jednou z nich je takzvaný Tangle, tedy orientovaný acyklický graf vytvořený společností IOTA. Tangle a technologie s ním spojené by díky svým vlastnostem měly být vhodné jako protokol právě pro IoT. IOTA se snaží svůj IOTA protokol založený právě na technologii orientovaného acyklického grafu ustanovit jako nepsaný standard pro Internet věcí. Protokol je zatím v testovací fázi a velkou část vývoje má stále před sebou. IOTA Foundation, nezisková organizace stojící za protokolem, s ním má ale poměrně veliké plány a jak už bylo avizováno, usilují o to, aby se z něj stal standard pro IoT. Z tohoto důvodu jsem se rozhodl v rámci práce zaměřit na IOTA protokol, jakým způsobem funguje a jestli je vhodným řešením pro Internet věcí.

[1] Českým ekvivalentem slova blockchain byl dle volby na portálu blockchain.cz zvolen výraz „bločenka“

Kapitola 1

Cíle a metodika

1.1 Cíle

Hlavními cíli práce je provést analýzu protokolu IOTA a možností jeho využití v rámci Internetu věcí, probrat možné případy užití daného protokolu a nové obchodní modely a aplikace, které mohou díky němu vzniknout a v poslední řadě vybrat jednu z těchto aplikací a implementovat jádro systému, který bude aplikace využívat.

Za účelem analýzy využití IOTA protokolu pro Internet věcí je nutné zanalyzovat Internet věcí jako takový a zaměřit se primárně na hlavní koncepty o které se opírá. U těchto konceptů a požadavků, které klade na využívané technologie zanalyzovat, jestli IOTA požadavky splňuje a na základě toho rozhodnout o vhodnosti projektu pro použití v rámci IoT ekosystému.

Jelikož je protokol ještě ve vývoji a jeho finální podoba se všemi funkcemi by měla být až v budoucnosti, je také potřeba zaměřit se na roadmapu projektu.

Za účelem vyvinutí jádra aplikace na protokolu IOTA bude potřeba zanalyzovat i možnosti vývoje na této technologii. Aktuálně je ve vývoji několik knihoven pro různé jazyky a zároveň vzniká přímo nový funkcionální jazyk pro tento protokol. Bude tedy třeba rozhodnout, jaké technologie budou pro vývoj využity a jejich volbu odůvodnit.

Veskrze jsou cíle práce především analytického a rešeršního charakteru. Jelikož se jedná o velmi novou technologii, využívající poměrně neznámé a nové principy, je analýza rozhodně na místě. Primárním cílem práce je tedy seznámit čtenáře s protokolem IOTA a dát mu vhled do toho, jak funguje a proč by měl, případně neměl být využíván pro Internet věcí.

1.2 Metodika

Metodika na základě které bude tato diplomová práce vznikat přímo vychází z cílů. Práce bude mít tři hlavní části a to část rešeršní, analytickou a imple-

mentační.

První, rešeršní část by měla seznámit čtenáře s historií hlavních technologií o kterých se bude dále v práci mluvit, tedy internetu samotného, následně Internetu věcí a nakonec IOTA protokolu. Tam kde je to potřeba by také mělo být vysvětleno, jak daná technologie funguje, tedy na čem je založená. Tato část by měla sloužit jako podklad pro další analytickou část.

Analytická část, která bude navazovat na rešerši by měla již probírat jednotlivé otázky, které byly vytyčeny v rámci cílů práce. Hlavními tématy tedy budou problémy s kterými se Internet věcí potýká a v návaznosti na to analýza IOTA protokolu a jeho vhodnosti jako standardu pro Internet věcí. Poté budou probrány jednotlivé ukázkové obchodní modely a aplikace, které díky protokolu mohou vzniknout.

V poslední hlavní části, tedy části implementační se bude implementovat jádro jedné z aplikací z analytické části. Tato část bude mít také svou analytickou část, která se ale bude už zabývat přímo otázkami spjatými s vývojem, tedy architekturou systému, použitými technologiemi a dalšími věcmi potřebnými pro vývoj. Následně bude vysvětlena implementace jako taková, což bude obohaceno ukázkami kódu, případně pseudokódů a dalších věcí, které čtenáři dají alespoň částečný vhled do implementace. Na závěr kapitoly bude samozřejmě zhodnocena implementace a doporučeny další kroky pro vývoj aplikace.

1.3 Doba vzniku práce

Jelikož je IOTA velmi mladá technologie a celkový ekosystém spjatý s decentralizovanými technologiemi, kryptoměnami a internetem věcí se vyvíjí velmi rychle, je třeba upozornit na to, kdy tato práce vznikla, jelikož je možné, že část informací, které budou rozebírány, již nebudou platné v době, kdy tuto práci čtete. Z tohoto důvodu zde uvádím, že všechny informace v práci uvedené jsou platné ke dni 30.11.2018 a informace týkající se implementace, tedy knihovny a způsoby práce s nimi jsou platné ke dni 31.12.2018.

Kapitola 2

Rešeršní část

2.1 Internet věcí

2.1.1 Internet

2.1.1.1 Historie internetu

Ačkoliv se dodnes vedou spory o přesném roce, kdy vznikl internet tak, jak ho známe dnes, jeho historie, ačkoliv časově poměrně dlouhá, není zase tak složitá a dá se jednoduše shrnout na pár řádcích. O to se také v následující části pokusím.

První zmínky o něčem, co by mělo připomínat internet, tedy možnost sociální interakce pomocí síťování, vznikly v roce 1962. Za zprávami, které pojednávaly o „Galaktické síti“ („Galactic network“) stál J.C.R. Licklider z MIT. Podařilo se mu přesvědčit své profesní okolí o důležitosti tohoto konceptu. Pocit nutnosti vyvinout něco podobného ještě podpořil strach ze Sovětského svazu a toho, že by jednoduše mohl vyřadit celý americký telefonní systém, který byl v té době hlavní komunikační technologií. Galaktická síť by umožňovala komunikaci i v případě napadení telefonního systému. Následně v roce 1964 vydala RAND[2] skupina, v rámci armády spojených států, pojednání zabývající se technologií přepínání paketů (packet switching). Tato technologie umožnila přenášená data rozdělit na části, takzvané pakety a ty pak jednotlivě posílat různými cestami, což zajistí lepší dostupnost a horší napadnutelnost sítě. Během toho probíhal návrh a diskuze o první síti nazvané ARPANET. V roce 1968 byl její návrh značně upraven a začal vývoj prvních komponent. První připojené zařízení bylo poté v roce 1969 a jednalo se o počítač UCLA (Kalifornská univerzita v Los Angeles). Na konci roku k ARPANETu byly připojené již čtyři zařízení a s postupem času jejich počet narůstal. S rostoucím počtem ale přestával být původní komunikační protokol vyhovující, především z důvodu nemožnosti řešit chyby při přenosech, které v důsledku

[2] Americká nezisková výzkumná skupina

mohly zastavit celou síť. Z tohoto důvodu vyšlo v roce 1974 pojednání od Vinta Cerfa a Boba Kahna, kde byl popsán princip, z kterého následně vznikl TCP/IP protokol, tak jak ho známe dnes. Internet se postupně technologicky vyvíjel, ale hlavní krok přišel v roce 1991, kdy Tim Berners-Lee v CERNu vytvořil HTML (jazyk pro tvorbu webových stránek) a HTTP (protokol určený pro výměnu hypertextových HTML dokumentů), což dalo vzniknout systému World Wide Web (www), který nám umožňuje internet využívat tak, jak jsme na to dnes zvyklí, tedy především vytvářet a propojovat jednotlivé webové stránky. Od tohoto okamžiku dochází již pouze k minoritním zlepšováním, ale samotná podstata internetu a www byla dokončena právě v roce 1991. [1]

2.1.1.2 Technologie internetu

Když se budeme zabývat otázkou, na jakých technologiích je internet založen a jak funguje, musíme se zabývat dvěma částmi, které spolu ale velmi blízce souvisí - hardwarovou infrastrukturou a protokoly. Tyto dvě části spolu plní primární a vlastně jedinou funkci internetu, kterou je přenos dat mezi dvěma zařízeními v něm. Internet využívá velkou spoustu různých protokolů, které, ačkoliv jsou často samy o sobě velmi zajímavé, také většinou plní pouze velmi specifický úkol, například jako protokol SMTP, který umožňuje posílání e-mailů. Z tohoto důvodu se zaměřím v této kapitole pouze na opravdu základní protokoly, které jsou k přenosu dat nezbytné.

Když se zaměříme na samotný přenos dat, mluvíme o posílání dat z jednoho zařízení do druhého. Obě tyto zařízení musí být v síti identifikovány unikátním identifikátorem. V tomto případě IP adresou, jejíž formát, ani další specifické informace nejsou pro pochopení důležité. K přenosu se používá hlavní skupina protokolu nazvaná TCP/IP. Tuto skupinu rozdělujeme do čtyř vrstev:

1. Aplikační vrstva - protokoly pro specifické aplikace, jako například SMTP
2. Transportní vrstva - poskytuje kanál pro přenos dat a jejich případnou kontrolu
3. Síťová vrstva - zajišťuje směrování přenášených dat
4. Vrstva síťového rozhraní - zajišťuje samotný přenos, tedy zakódování a přístup k přenosovému médiu

Při komunikaci je zpráva v případě potřeby rozdělena na více částí a ty jsou posílány odděleně. Každá část ale při přenosu nejprve projde všemi vrstvami, od aplikační do vrstvy síťového rozhraní. V každé části je s daty proveden specifický úkon a většinou jsou data „obalena“ hlavičkou specifickou pro danou vrstvu, která nese informace důležité k doručení. Následně je obalený kus dat poslán po internetu a na druhé straně prochází opět vrstvami, ale v opačném

sledu, tedy od síťového rozhraní po aplikační vrstvu. Toto je velice stručně popsaný princip přenosu dat s použitím TCP/IP protokolu. K přenosu ale potřebujeme ještě druhou část a to hardwarovou infrastrukturu.

Hardwarová infrastruktura je vlastně fyzickou kostrou internetu, která je využívána k veškeré komunikaci. Základem sítě je takzvaná Internetová páteř. Jedná se o velkou skupinu navzájem propojených sítí, které jsou většinou provozovány vládními nebo akademickými celky, případně velkými technologickými subjekty. Tato páteř se stará především o hlavní pokrytí světa a přenos dat mezi kontinenty. V případě, že je třeba poslat data z jednoho zařízení do druhého, je potom hlavním využívaným zařízeným takzvaný směrovač (router). Toto zařízení má uloženou směrovací tabulku, která obsahuje jednotlivé IP adresy zařízení a k nim odpovídající porty, kam data poslat. V případě, že pro danou adresu nemá odpovídající port, pošle data na výchozí adresu, na které je většinou další router. Toto umožňuje vytvořit hierarchii routerů a v důsledku toho hierarchii sítí. Pokud jsou tedy odeslána data a mají být doručena na určitou adresu, jsou postupně posílána mezi routery v hierarchii směrem nahoru, dokud není nalezen router, který ve své tabulce danou adresu má a adresa je tedy v jeho subsíti. Toto je v podstatě ve zkratce hlavní princip internetu.

Internet jako takový využívá samozřejmě pro své fungování velkou řadu protokolů, které nebyly zmíněny, jelikož jako takové nejsou důležité pro tuto práci. Hlavní důležitá informace je ta, že internet jako takový je jedna velká síť, složená ze subsítí, které umožňují přenos dat mezi zařízeními v nich. K tomuto je využívána určitá hardwarová infrastruktura a především skupina protokolů, která definuje, jak jednotlivé komponenty pracují a jak jsou data posílána. Je zároveň důležité sledovat stav sítě a chápat, že existují určité metriky, jako například rychlost přenosu, které ovlivňují celkovou použitelnost sítě. [2]

2.1.2 Internet věcí

Internet věcí je poslední dobou velmi skloňované téma, avšak stále není ucelená definice, která by říkala, co přesně IoT je. Názory se velmi liší a na jedné straně jsou představy, že celý Internet věcí je prakticky pouze nejnižší fyzická vrstva, která umožňuje síťové propojení zařízení se senzory a na straně druhé potom představa Internetu věcí jako všeobjímajícího konstruktu, prostupujícího nejenom technologiemi, ale i celou společností a světem. Mezi těmito dvěma protipóly je potom spousta dalších definic, které se přiklánějí spíše k jedné, nebo druhé straně mince.

Z tohoto důvodu budu v rámci této práce vycházet z dokumentu Towards a definition of the Internet of Things (IoT), od organizace IEEE, což je nezisková organizace, která si klade za cíl technologický rozvoj v oblasti elektrického a elektronického inženýrství a v oblastech s nimi souvisejících. Tento dokument byl předložen veřejnosti v roce 2015 a jeho cílem je blíže specifikovat oblast IoT. Veřejnost byla v rámci vydání tohoto dokumentu vyzvána, aby

se na definici spolupodílela a na základě toho by se měl dokument rozrůstat a upravovat. Ačkoliv tedy dokument ani definice v něm obsažené nejsou finální, budu z něj v rámci této práce vycházet, jelikož se dle mého názoru jedná o jednu z nejdetailnějších a nejlépe strukturovaných definic této problematiky a zároveň je jejím autorem organizace IEEE, tedy instituce, která stojí za již nespočtem podobných obecně uznávaných definic a standardů.

Pokud tedy budeme vycházet z výše uvedeného dokumentu, budeme brát IoT jako obor, který se zabývá propojením různých technologií a společenských oborů. Ve velice obecném smyslu tedy praktickým propojením světa v rámci jedné sítě. Jelikož se jedná o opravdu široký záběr, rozdělujeme IoT do několika hlavních oblastí, které mohou být reprezentovány jako jednotlivé vrstvy IoT. Nejnižší vrstvou je oblast týkající se **systémové architektury a technologií, které vůbec umožňují Internet věcí**. Zde mluvíme o protokolech, senzorech, napájení a dalších nejdůležitějších technologiích pro samotné fungování ekosystému. Nad touto vrstvou je vrstva **softwarové architektury**. Do ní spadají především operační systémy, middleware, big data, API a další technologie, které umožňují udržovat ekosystém z hlediska softwarového. Nad ní se nachází poslední technologická vrstva a to vrstva **služeb a aplikací**. Sem spadají už samotná řešení postavená nad technologiemi. Řešení, která přivedou Internet věcí ke koncovému uživateli. Jedná se o samotné aplikace a služby, jako například chytrá města, aplikace běžící na domácích spotřebičích a spoustu dalších, kterými se částečně budu zabývat v budoucích kapitolách. Toto byly tři hlavní technologické vrstvy. Nad aplikační vrstvou jsou však ještě další dvě vrstvy, které jsou s IoT úzce spojené. Nejsou technického rázu, avšak z hlediska ekosystému, jeho využití a budoucnosti nejsou o nic méně důležité. Přímo nad aplikační vrstvou se nachází vrstva **obchodních modelů a ekosystému**. Sem spadají především věci týkající se obchodních modelů, využívajících aplikací z nižší vrstvy koncovými uživateli, což je v důsledku velmi důležité, protože to umožní provoz samotného ekosystému. Každý stroj v síti musí být něčím poháněn a obchodní modely by měly zajistit, že stroj, případně jeho provozovatel bude mít dostatek prostředků, aby mohl běžet. Nejvyšší a také nejvíce abstraktní vrstvou je vrstva **sociálního dopadu IoT**. Tato vrstva je už velmi vzdálená od samotných technologií. Jedná se o dopady, které bude mít Internet věcí na společnost, změny které díky těmto technologiím budou moci proběhnout a to, jak se změní fungování spousty oborů, institucí a celkově obraz světa. Jak je tedy z této hierarchie vidět, IoT v podstatě začíná u nejjednodušších senzorů a pomocí jejich integrace a hromadného využití se snaží vylepšit spoustu odvětví a celkově život tak jak ho známe.[3]

2.1.2.1 Požadavky na IoT systém

Vzhledem k tomu, že Internet věcí má být obrovský ekosystém, který má propojovat opravdu velké množství zařízení, dá se předpokládat, že po síti budou

proudit obrovské objemy dat. Každé zařízení si bude vyměňovat informace se spoustou dalších a to vše optimálně v reálném čase, aby mohly další zařízení okamžitě reagovat. Do toho všeho je potřeba tyto data zpracovávat a analyzovat pro provozovatele systémů a zároveň data určitým způsobem agregovat a prezentovat koncovým uživatelům. Toto klade poměrně specifické požadavky na infrastrukturu, protokoly, senzory a prakticky všechny technologické subjekty, které se nějakou mírou do provozu ekosystému zapojují.

Bezpečnost Hlavním požadavkem na systémy běžící v rámci Internetu věcí je bezpečnost. Vzhledem k jednoduchosti většiny systémů běžících na chytrých zařízeních, jako je například chytrý kávovar, je často bezpečnost přehlížená. První běžnou myšlenkou je fakt, že útočník nemá nic, co by z chytrého kávovaru ukradl. V nejhorším případě by přeprogramoval systém tak, že by místo espressa dělal lungo. Problémem je však fakt, že kávovar je připojený k lokální síti a tím, že se jej útočníkovi podařilo kvůli slabému zabezpečení napadnout, dostal přístup do lokální sítě a díky tomu je následně schopný napadnout i další zařízení, která už mohou pro provozovatele, nebo vlastníka sítě představovat větší problém. Například směrovač. Zároveň vzhledem k tomu, že v budoucnu bude zpracování velkých dat na o mnoho lepší úrovni, budou mít data poměrně velkou cenu. Vzhledem k tomu, kolik dat bude proudit v IoT síti, bude určitě snaha tyto data zcizit. Z tohoto důvodu bude důležité i zabezpečení samotných přenášených dat. Proto je bezpečnost jedním z hlavních požadavků, avšak nejedná se přímo o požadavek vycházející z podstaty Internetu věcí. Jde o obecný požadavek na všechny informační systémy a hardware. Pouze je zatím v rámci IoT zařízení často lehce opomínán a z tohoto důvodu jej považuji za velmi důležitý.

Propojitelnost Jelikož má v rámci sítě komunikovat spoustu zařízení, je důležité, aby byly systémy vytvořeny tak, aby se s jejich rozhraním dalo jednoduše komunikovat. V budoucnu se předpokládá, že každé zařízení bude spolupracovat s mnoha odlišnými typy systémů a dalších zařízení. Ledničky budou komunikovat s drony, rozhraními obchodů, kávovary a dalšími podobnými systémy. Z tohoto důvodu je třeba je vytvářet takovým způsobem, aby jejich propojení nebylo složité a nebyla vyžadována spousta různých komunikačních protokolů a překladačů rozhraní.

Rychlost Rychlost je jednou z dalších hlavních proměnných, které do fungování ekosystému vstupují. Jednotlivá zařízení potřebují rychle reagovat na okolní dění, nebo na další pro ně důležité informace. Z tohoto důvodu nemůže docházet k zahlcení sítě a frontám, kdy by trvalo několik sekund, nebo i více, než se dostane informace ze senzoru jednoho zařízení do druhého. Síť a všechny její části musí fungovat rychle a nezdržovat komunikaci.

Škálovatelnost Na rychlost navazuje další požadavek a tím je škálovatelnost. Jedná se o jednu z aktuálně nejdůležitějších vlastností a zároveň vlastnost, která zatím působí největší potíže při snaze o vývoj funkčního IoT. Jelikož se v budoucnu počítá s velkým počtem zařízení a zároveň s tím, že počet se bude postupně zvyšovat, je třeba mít možnost síť jednoduše škálovat. Optimálně tak, aby se síť nezpomalovala s rostoucím počtem zařízení v ní.

Spotřeba S velkým počtem zařízení se dá očekávat i velká spotřeba energie. Zde jsou dva hlavní problémy. Prvním je ekologický dopad, který bude mít provoz takto velké sítě. Výroba energie na světě zatím není v takovém stavu, abychom byli schopní produkovat její neomezené množství. Provoz IoT se může tedy nepříznivě podepsat na energetickém průmyslu a v důsledku toho na samotné Zemi. Druhým problémem jsou pak i samotná zařízení. Dá se počítat s tím, že spousta zařízení operujících v síti bude mobilní. Létající, pojízdná a další zařízení, která nebudou mít možnost být neustále připojena ke zdroji energie. Z tohoto důvodu je třeba, aby jejich spotřeba byla co nejmenší, jelikož mohou plnit svoje poslání pouze ve chvíli, kdy mohou cestovat a nemusí být na místě a čekat na dobití.

Požadavků je samozřejmě mnohem více, ale tyto považuji za hlavní. Zároveň spousta dalších požadavků se dá považovat za součást jednoho z výše vypsaných. Ve výsledku je tedy třeba, aby při vývoji systémů, protokolů a hardware pro IoT bylo bráno v potaz, že vyvinuté věci musí být dostatečně zabezpečené, jednoduše propojitelné, rychlé, škálovatelné a musí mít nízkou spotřebu. Toto vše umožní vytvořit IoT ekosystém, který bude moci správně plnit svůj cíl. [4]

2.2 IOTA

Hlavním cílem IOTA je stát se standardem v rámci Internetu věcí. Nejedná se o přenosový protokol. IOTA jako taková je schopna pracovat na jakémkoliv transportním protokolu, ať už se jedná o bluetooth, wifi, nebo cokoliv jiného. Cílem IOTA je stát se standardem pro ukládání dat, tvorbu aplikací nad sítí a především se stát hlavní technologií pro M2M (machine to machine) ekonomii. Vzhledem k tomu, že se jedná o poměrně novou technologii, obsahuje spoustu specifických názvosloví, pro která ještě nejsou ustálené české překlady. V takovém případě se tedy budou v textu často používat originální anglické výrazy.

2.2.1 Distributed ledger

IOTA je podle IOTA Foundation novou generací technologie distributed ledger (do češtiny občas překládáno doslovně, tedy jako distribuovaná účetní kniha).

Pro pochopení IOTA technologie je tedy potřeba nejprve vysvětlit samotnou technologii distributed ledger.

Jedná se v podstatě o běžnou účetní knihu, tedy něco, co zaznamenává přesun aktiv mezi subjekty formou transakcí. Tyto data mohou být zaznamenána různými formami. V blízké minulosti byly většinou účetní knihy opravdové papírové knihy. Nyní jsou většinou již data uchovávána v digitálních databázích, nad kterými jsou postavené určité systémy. Tyto technologie ale stále mají nevýhodu, kterou je fakt, že jsou data uložena na jednom místě. Jejich replikace a distribuce je potom složitá a zároveň není jednoduché ověřit integritu dat. Toto řeší právě distributed ledger. Jedná se o distribuovanou databázi, tedy databázi, která je sdílená napříč sítí. Každý subjekt v rámci sítě tedy může mít celou kopii dané databáze. Transakce se propisují do několika sekund či minut napříč celou sítí a všechny subjekty tedy vidí aktuální reálný stav. Nejznámější takovou technologií současnosti je asi blockchain. Známý je především díky své implementaci v rámci virtuální měny Bitcoin. Jedná se vlastně o implementaci distributed ledger, která umožňuje jej provozovat díky principům a algoritmům, které řeší právě integritu, bezpečnost a další otázky spojené s distribuovanou databází. IOTA je další implementací distributed ledger, není ale založena na blockchainu, nýbrž na jiné technologii. Touto technologií je orientovaný acyklický graf (directed acyclic graph, DAG), který zakladatelé IOTA nazvali Tangle.

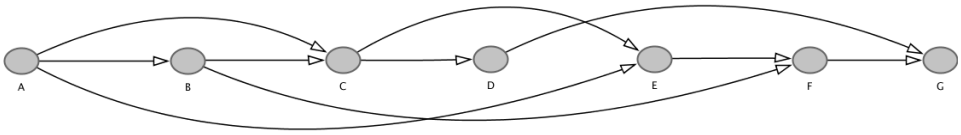
Distributed ledger jako technologie přináší novou formu bezpečnosti a integrity do databází. Vzhledem k tomu, že je databáze distribuovaná mezi velké množství subjektů, musel by útočník napadnout ve stejnou chvíli všechny, respektive alespoň 33% všech subjektů. Z tohoto důvodu se o tuto technologii aktuálně ve velkém zajímají vlády států a velké firmy, které by distribuované databáze rády využily, ať už v případě vlád například pro výběr daní, tak v případě firem například pro lepší zabezpečení svých dat.[5]

2.2.2 Orientovaný acyklický graf

Jak bylo zmíněno v předchozí kapitole, IOTA je založena na technologii orientovaného acyklického grafu, který je pracovně pojmenován Tangle. Pro porozumění základů na kterých IOTA stojí je tedy třeba nejprve vysvětlit, co v rámci grafu znamená to, že je orientovaný a acyklický.

V případě **orientovaného grafu** v rámci teorie grafů myslíme uspořádanou dvojici G = (V, A), kde V jsou všechny uzly grafu a E jsou pak hrany grafu, tedy uspořádané dvojice, které nám říkají, odkud kam hrana jde. Všechny hrany tedy mají směr odkud kam jdou, odtud orientovaný graf.[6]

Druhou vlastností tohoto grafu je **acykličnost**. Acyklický graf je takový, který v sobě neobsahuje žádné grafové cykly, tedy subgrafy, kde nalezneme uzel, který je počátečním a zároveň posledním, v rámci grafu se tedy vytvořila kružnice. Toto v rámci acyklického grafu nemůže nastat, jelikož všechny hrany jdou v podstatě jedním směrem. Hrany se tedy nevrací zpět k předchozím



Obrázek 2.1: Orientovaný acyklický graf

uzlům. Pokud označíme všechny uzly čísly od 0 do n, nikdy by neměla být hrana od uzlu s vyšší hodnotou do uzlu s nižší. Orientovaný acyklický graf, viditelný na obrázku 2.1, je tedy graf, který má směrované hrany, které jdou jedním směrem a tím pádem nevytváří cykly. Díky těmto vlastnostem je vhodným základem pro IOTA protokol.

2.2.3 Princip fungování IOTA

2.2.3.1 Základní princip

Základní myšlenkou fungování Tangle je to, že pro založení transakce v síti musí zakládající uživatel udělat práci, která potvrdí další dvě transakce. Uzly sítě, skrz které je transakce vytvářena samozřejmě kontrolují, jestli v rámci transakcí nedochází ke konfliktům s historií Tangle a pokud ano, tak nepovolí danou transakci vytvořit. Čím vícekrát byla určitá transakce potvrzena, tím důvěryhodnější je v rámci sítě. Aby byla transakce vytvořena, vytvářející uzel nejprve vybere dvě transakce podle specifického algoritmu, který bude probrán v rámci dalších kapitol. Následně je uzlem provedena kontrola, jestli tyto dvě transakce nejsou konfliktní a pokud ne, tak jsou transakce potvrzeny uzlem tím, že vyřeší určitý kryptografický úkol, podobný tomu, který se řeší v rámci potvrzování transakce Bitcoinu. Tento algoritmus je také obsahem dalších kapitol. Transakce potom odpovídají jednotlivým uzlům grafu a hrany reprezentují to, která transakce potvrzuje kterou. Orientovanost grafu tedy umožňuje zaznamenávat jednotlivá potvrzování a acykličnost je následně zajištěna tím, že nové transakce potvrzují starší a starší transakce už nové nepotvrzují. Každá transakce tedy potvrzuje další transakce pouze při svém vzniku a dál už je pouze zaznamenána v rámci účetní knihy. Na začátku byl Tangle vytvořen s iniciální transakcí, které se říká „genesis“, tato transakce dala vzniknout všem tokenům. Co a k čemu je token bude probráno v jedné z budoucích kapitol.[7]

2.2.3.2 Uzly

Ačkoliv v předchozím odstavci bylo řečeno, že transakce jsou uzly grafu, což je pravda, je pojem uzel v rámci IOTA používán v trochu jiném významu. Transakce jsou obsaženy v distribuované účetní knize, která je orientovaným

acyklickým grafem, jak už bylo několikrát řečeno. Uzly o kterých ale mluvíme jsou uzly jiného grafu a to grafu síťové topologie, která umožňuje decentralizaci. Každý uzel této topologie je zařízení, které má staženou celou kopii Tangle. Zároveň musí mít přiřazeny sousední uzly, kterým posílá transakce, které vytvořil. Jedná se o broadcast operaci, kterou se zařídí distribuce do všech uzlů. Tyto uzly tvoří základní topologii, která udržuje Tangle v chodu. Jelikož uzly mají celou kopii Tangle, říká se jim plné uzly (Full nodes). Vedle nich ještě existují takzvané lehké uzly (Light nodes). Každý lehký uzel musí být napojený na nějaký plný a pracuje fakticky pouze jako rozhraní. Komunikuje s uzlem a umožňuje uživateli pracovat se sítí, aniž by musel stahovat celou její historii.[8]

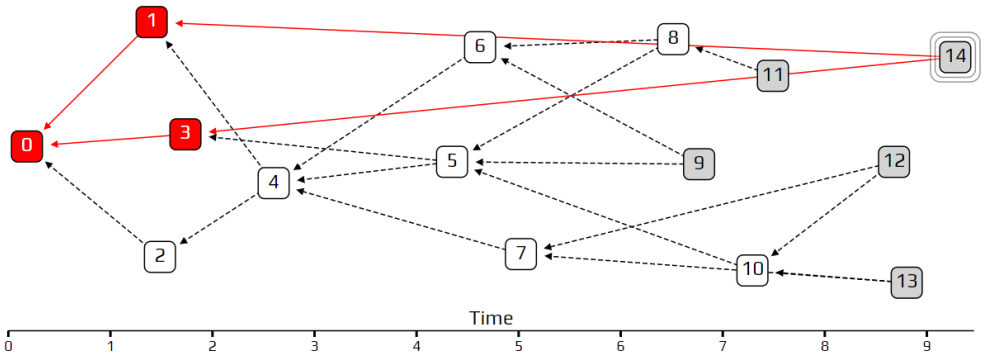
2.2.3.3 Váhy transakcí

Hlavním parametrem, který u transakce zaznamenáváme je její váha. Tyto váhy dělíme na dvě - vlastní váhu transakce a její kumulativní váhu. Vlastní váha je váha, která je transakci přidělena při jejím vzniku a odpovídá tomu, kolik práce bylo při jejím vytváření zakládajícím uzlem odpracováno. V tuto chvíli jsou povolené hodnoty pro vlastní váhu pouze 3^n, kde n je kladné celé číslo. Kumulativní váha transakce je potom součet její vlastní váhy a vlastních vah všech transakcí, které ji přímo nebo nepřímo potvrzují. S rostoucí kumulativní vahou transakce samozřejmě roste její důvěryhodnost. Vzhledem k tomuto principu jsou tedy starší transakce důvěryhodnější a jejich dvěryhodnost stále roste, jelikož každá nová transakce zvětší váhu dvou dalších, které zvětší váhu těch které potvrzují tyto dvě a v důsledku se takto zvýší váha všech předchozích transakcí, které jsou nepřímo potvrzeny nově vytvořenou. U jednotlivých transakcí ještě můžeme zaznamenávat jejich výšku a hloubku, což jsou parametry, které v tuto chvíli nejsou důležité a v případě potřeby budou vysvětleny v budoucích kapitolách.[7]

2.2.3.4 Výběr transakcí k potvrzení

Algoritmus, který vybírá zatím nepotvrzené transakce (nazývané „tips“), které mají být potvrzeny novou transakcí, je metoda Monte Carlo pomocí náhodné procházky (RWMC - Random Walk Monte Carlo - podtřída třídy algoritmů Monte Carlo pomocí Markovova Řetězce). Jádrem algoritmu pro výběr transakcí, které budou nově příchozí transakcí potvrzeny, je tedy náhodná procházka. Jedná se o proces, během kterého se v každém kroku objekt pohne určitým náhodným směrem[9]. V tomto případě při výběru transakce je v prvním kroku vybrána genesis transakce, tedy úplně první transakce v Tangle a v každém dalším kroku se výběr posune do jedné z transakcí, které transakci tohoto kroku přímo potvrzují. V případě, že by se jednalo čistě o náhodnou procházku, byla by transakce na kterou se v dalším kroku posuneme vybírána náhodně. Toto by ovšem neřešilo problém takzvaných „líných

transakcí“ (Osobní překlad, v původním znění „lazy tips“). Jde o transakce, které potvrzují transakce staré a ne nové, a to z toho důvodu, že průběžně nesledují aktuální stav Tangle. Nevidí tedy nové transakce a potvrzují ty, které jsou z jejich pohledu nejnovější, ačkoliv ve skutečnosti jsou již zastaralé a měly by být potvrzované novější. Líná transakce je vyobrazena na obrázku 2.2.



Obrázek 2.2: Líná transakce 14 potvrzující transakce staré

Na něm je vidět, že nově vzniklá transakce 14 potvrzuje transakce 0, 1 a 3, tedy jedny z nejstarších transakcí, ačkoliv by optimálně měla potvrzovat spíše například transakce 8, 9 nebo 11. Toto chování je pro síť nevhodné a je třeba mu určitým způsobem zabránit. Možností, která je využívána je trestat lazy tips tím, že budou při výběru transakcí k potvrzení upřednostňovány transakce, které se chovají správně a drží krok s aktuálním stavem sítě. V případě běžné náhodné procházky tyto špatně se chovající transakce nejsou nijak penalizovány. Zároveň ale není možné vytvořit pravidlo, že nově vznikající transakce vždy musí potvrdit dvě nejnovější transakce v síti, jelikož nelze přesně říci, kdy která transakce vznikla. Z tohoto důvodu je místo běžné náhodné procházky využita náhodná procházka vážená. Ta je založena na principu, kdy se na další transakci neposouváme s náhodnou pravděpodobností, ale na každou se můžeme posunout s určitou pravděpodobností. Zde je využita kumulativní váha transakce o které je psáno v minulém odstavci zabývajícím se vahami. Tato kumulativní váha je určujícím faktorem pro pravděpodobnost přechodu. Transakce s vyšší kumulativní vahou mají větší šanci, že se do nich v rámci náhodné procházky dostaneme. Tímto je zajištěno to, že ze staré transakce nepůjdeme rovnou na jednu z nejnovějších, která je lazy tip, jelikož potvrzuje transakci starou. Lazy tip bude mít totiž malou kumulativní váhu a z transakce tedy vždy s větší pravděpodobností přejdeme do transakce, která je bližší času vytvoření transakce, ve které se aktuálně nacházíme.[10] V rámci určování pravděpodobnosti se ještě pracuje s parametrem α, který určuje, jak velkou váhu v rámci pravděpodobnosti má právě kumulativní váha.

Pro $\alpha = 0$ máme čistou náhodnou procházku, ve které váha nehraje žádnou roli. Pro $\alpha = 1$ naopak váha hraje jedinou roli a v každém kroku přejdeme na potvrzující transakci s nejvyšší kumulativní váhou. To také není vhodné, jelikož v takovém případě by docházelo k nepotvrzení velké části transakcí. Vhodná hodnota pro parametr α je stále předmětem výzkumů. Obecně se pro určité situace používají jeho různé hodnoty.[7]

2.2.3.5 Vytvoření transakce

K tomu, aby byla transakce akceptovaná Tangle, musí obsahovat takzvanou kryptografickou nonce. Výraz nonce, používaný v kryptografii, obecně znamená číslo, které je určeno pouze k jednorázovému použití. Z tohoto důvodu je často jeho součástí aktuální časová značka. Data transakce jsou zakódována a uložena ve 2673 trytech. Tryte je ternární obdobou bajtu. Je složen z pěti tritů. Ternární logikou a důvody proč ji IOTA využívá, se budu zabývat v budoucích kapitolách této práce. Posledních 81 trytů je rezervováno právě pro nonce. K tomu aby transakce získala nonce, musí nejprve vyřešit určitý kryptografický problém. Principu, kdy k využití služby je nejprve třeba provést výpočetní úkony, se obecně říká proof-of-work a jedná se poměrně vhodný nástroj pro zamezení DoS (Denial of Service - typ útoku, kdy je generováno velké množství požadavků, tím zahlcena síť a tak znemožněn přístup ke službě ostatním uživatelům) útokům, nebo spamu.[11] Takovýmto známým systémem je například Hashcash. Algoritmus je používán především k zamezení spamu v rámci elektronické pošty. Funguje tak, že pro odeslání musí odesilatel přiložit hlavičku v určitém formátu. Hlavička se skládá z určitých dat doplněných číslem, které je na začátku nastaveno na náhodnou hodnotu. Následně je hlavička zahashována funkcí SHA-1 a pokud začíná na 20 nulových bitů, jedná se o použitelnou hlavičku. Pokud na začátku nemá alespoň 20 nulových bitů, je třeba upravit doplňující číslo a zkusit hlavičku zahashovat znovu. Toto je třeba opakovat do té doby, než je nalezena hlavička, která má požadovaný formát. Díky tomuto nelze efektivně spamovat síť, jelikož není možné generovat velké množství validních hlaviček v krátkém čase.[12]

IOTA je založena na velmi podobném principu. Nonce, které musí být přiloženo na konci zakódované transakce musí obsahovat určitý počet nulových tritů na konci. Tento počet je určen konstantou, které se říká minimum weight magnitude (MWM). V době psaní této práce je na hlavní síti nastavena na hodnotu 15. Nonce tedy musí mít na konci minimálně 15 nulových tritů. Jelikož se jedná o ternární soustavu, není možné použít klasické hashovací funkce a z toho důvodu byla IOTA vývojáři vytvořena první existující ternární hashovací funkce, která se nazývá Curl. Tato funkce vzbudila velkou kontroverzi, především kvůli výzkumu týmu z MIT, který v ní nalezl určité chyby a označil ji za nedostatečně bezpečnou a celou IOTA síť tedy za jednoduše napadnutelnou. Chyba byla následně opravena.[13]. Tato implementace proof-of-work zabraňuje mnoha typům útoku na síť. [14]

2.2.3.6 Validní transakce

V předchozích kapitolách bylo vysvětleno, jakým způsobem je transakce potvrzovaná a co znamená, že jedna transakce potvrzuje jinou. Hlavní věcí, která je ale důležitá pro práci se samotnou účetní knihou je fakt, jestli je určitá transakce brána jako potvrzená v rámci celé sítě (v rámci odlišení názvosloví pro potvrzení jedné transakce druhou a potvrzení transakce v rámci celé sítě, nazývejme toto potvrzení validitou transakce), nebo ne. Díky tomu jsme schopní například přijmout platbu. Ve chvíli co víme, že je transakce validní, máme jistotu, že proběhla a že tedy zdroje byly převedeny. Rozhodnutí, jestli je určitá transakce validní, je založeno na stejném algoritmu, jako výběr transakcí k potvrzení, tedy na metodě Monte Carlo pomocí náhodné procházky.

Při ověřování validity je na začátku vybráno pomocí RWMC sto zatím nepotvrzených transakcí a následně zjistíme, kolik z nich přímo nebo nepřímo potvrzuje ověřovanou transakci. Procento transakcí, které ji potvrzují se nazývá jistota potvrzení (Osobní překlad, v původním znění „confirmation confidence“). Kdy bereme transakci jako validní si potom určuje sám tvůrce aplikace. Typicky není nikdy jistota validity, ale vždy máme pouze jistotu potvrzení.

2.2.3.7 Koordinátor

Jednou z hlavních otázek, která se týká decentralizovaných účetních knih, je bezpečnost a především důvěryhodnost dat, která se v knize nachází. Existuje několik typů útoků, kterým může Tangle, stejně jako blockchain a další distribuované sítě, čelit. Každá technologie se před útoky brání jiným způsobem. V případě Tangle je hlavním bezpečnostní prvkem fakt, že vytvoření každé transakce vyžaduje určitou výpočetní kapacitu. Z tohoto důvodu by měla být možnost získat nadvládu nad většinou sítě nemožná. Je to způsobeno tím, že čím více transakcí vzniká, tím více transakcí je třeba vytvořit k tomu, aby jejich autor získal většinu v síti. Tím pádem narůstá i výpočetní kapacita, která je třeba k vytvoření daného počtu transakcí. Možnost vytvořit dostatečný počet podvodných transakcí k prosazení vlastních dat v síti by tedy měla být v teorii nemožná z důvodu obrovských nákladů na výpočetní sílu. Je zde ovšem určitá hranice počtu transakcí za sekundu, pod kterou není síť bezpečná, vzhledem k faktu, že pro bezpečnost je třeba, aby počet vznikajících transakcí byl znatelně větší oproti výpočetní kapacitě útočníka.

Protože ekosystém se ale nachází v poměrně raném stádiu vývoje, není síť využívána zdaleka tolik, jako by v budoucnu měla být. Počet transakcí, které vznikají je tedy poměrně malý a z tohoto důvodu nezajišťuje dostatečnou bezpečnost sítě. Pro útočníka by teoreticky nebylo tak těžké vytvořit mnoho uzlů a začít generovat velké množství podvodných transakcí, které by potom měly poměrně velkou šanci, že budou při náhodné procházce vybrány. Z tohoto

důvodu je v síti speciální uzel, provozovaný přímo IOTA Foundation, zvaný Koordinátor. Ten vytváří každé dvě minuty takzvanou milníkovou transakci (Milestone transaction). Transakce, které jsou přímo nebo nepřímo potvrzeny milníkovou transakcí jsou považovány za validní, tedy že mají jistotu potvrzení 100%. Tyto milníkové transakce nemohou být žádným způsobem falšovány, jelikož jsou speciálně podepsané Koordinátorem, což dokládá jejich zdroj. Z tohoto důvodu máme jistotu validity transakce, která by v tuto chvíli při poměrně malém počtu transakcí byla méně důvěryhodná, případně nedosažitelná.[15]

Existence Koordinátora vzbuzuje velké emoce v rámci veřejnosti, která z tohoto důvodu často Tangle napadá kvůli tomu, že není ve skutečnosti decentralizovaný, jelikož v něm existuje uzel, který určuje jednu pravdu. Ačkoliv je toto pravda a jedná se o uzel, který je vytvořen IOTA Foundation a zároveň jí provozován, uzel nevytváří nové tokeny, ani neumožňuje takzvané dvojí výdaje (typ útoku, kterým se také budeme zabývat v analytické části). Samozřejmě se ale jedná o určité specifikum, které by nemělo v síti být. Cílem IOTA Foundation je vytvořit naprosto decentralizovanou síť a z tohoto důvodu vidí odstranění Koordinátora jako nevyhnutelný krok. Původní plán byl, že k odstranění dojde v létě roku 2018, k čemuž však nedošlo, především z důvodu malé vyzrálosti sítě. Ta je hlavním problémem při snaze o odstranění. Dokud nebude síť dostatečně využívaná, aby mohl být Koordinátor odstraněn, aniž by vznikly potenciální bezpečnostní díry v síti, nemůže k odstranění dojít. Snahou je samozřejmě, aby k tomuto došlo co nejdříve, ale zde se jedná o otázku využití dané technologie, která není přímo v rukou IOTA Foundation a je tedy otázkou budoucnosti, s nemožností přesně určit specifický termín.[14]

2.2.3.8 Ternární logika

IOTA samotná není, jako téměř veškerá zbylá technologie, postavená na principu dvojkové soustavy (binární), ale na soustavě ternární, tedy trojkové. Přesněji na balancované trojkové soustavě, tedy používající oproti normální ternární soustavě s hodnotami 0, 1, 2 hodnoty -1, 0, 1. Důvodem k tomuto je fakt, že trojková soustava by měla být rychlejší než dvojková a v mnoha ohledech by i práce s ní měla být přirozenější, než právě se soustavou binární. Například práce se znaménky, nebo desetinnými čísly je o poznání jednodušší.

Veškerá data jsou tedy zakódována do tritů, ekvivalentu bitů dvojkové soustavy. Trity se slučují do trytů, tedy trojic tritů. Jeden tryte tedy může nabývat 27 různých hodnot. IOTA Foundation jednotlivým hodnotám přiřadila znaky a to velká písmena abecedy a číslo 9.[16] Vzhledem k aktuální absenci hardware, který by podporoval ternární soustavu, jsou data vždy zakódována do těchto znaků a následně přenášena klasickým způsobem, tedy zakódována do dvojkové soustavy. Tvůrci projektu ale věří, že v budoucnu budou standardem ternární mikroprocesory, které umožní práci přímo s ternární soustavou a tedy značně zrychlí a zjednoduší i práci se samotným IOTA protokolem.

Jedním z projektů, který by údajně měl již vyvíjet levný a energeticky velmi nenáročný ternární procesor je projekt Jinn. Tento projekt by měl být veden jedním z hlavním vývojářů IOTA. Spousta informací jsou ale spíše dohady a kromě několika komentářů na fórech, případně dalších komunikačních kanálech IOTA Foundation, je projekt poměrně neprůhledný.

Ternární logika je zároveň jedním z důvodů, proč je IOTA bezpečnější, než většina projektů podobného typu a to z důvodu, že je odolná vůči kvantovému počítání. Ačkoliv je technologie kvantového počítání zatím ve velmi rané a především teoretické fázi, pomalu vznikají jednoduché prototypy. Do budoucna se věří, že kvantové počítače budou fungovat a jako takové znehodnotí velkou část současné kryptografie, jelikož jejich výpočetní kapacita bude umožňovat prolomení velké části klíčů hrubou silou. V případě IOTA jsou klíče tvořeny 81 tryty a z tohoto důvodu jsou i pro kvantové počítače hrubou silou neprolomitelné. [14]

2.2.3.9 IOTA Token

Jelikož je IOTA protokol, jehož hlavní podstatou je technologie decentralizované účetní knihy, je zjevné, že kromě samotných technologií přenosu a distribuce dat je třeba, aby existovalo něco, co bude mít nominální hodnotu, která bude v rámci transakcí převáděna. Tuto hodnotu nese IOTA token. Token obecně je většinou definován jako mince nebo listina, jejíž nominální hodnota, tedy hodnota většinou na minci přímo uvedená, je větší než její vnitřní hodnota.[17] Vnitřní hodnotou je myšlena například cena materiálu ze kterého je daná mince vyrobena. Typickou ukázkou jsou například běžné měny jako euro nebo česká Koruna. Dalším důležitým aspektem je to, že token by měl být většinou spravován jednou institucí a nemělo by tedy být možné jeho hodnotu ovlivňovat například tím, že si ho doma sami vytvoříme, čímž jeho hodnotu devalvujeme vpuštěním velkého objemu na trh.

Poslední dobou se ukazuje, že tokeny jako takové vůbec nemusí být fyzické. Ukázkou jsou například kryptoměny, které vznikají v rámci specifických projektů a většinou jsou postaveny na důležité vlastnosti decentralizace. Toto jim dává určitou výhodu oproti běžným měnám spravovaným centrálními bankami. Banky samy rozhodují, kolik tokenů, v jejich případě většinou peněz, nechají vytisknout a pošlou do oběhu. Tím jsou schopné na přímo ovlivňovat hodnotu dané měny, což je základem monetární politiky. V moderní společnosti je toto čím dál tím více předmětem kritiky, jelikož je poukazováno na fakt, že to odporuje volnému trhu a takováto míra centralizace není vhodná, jelikož hodnota měny neodpovídá realitě, což může vést k ekonomickým problémům. Oproti tomu většina kryptoměn je založena na principu, kdy nikdo nemůže rozhodnout o vydání dalších tokenů do oběhu, ani jejich stažení z něj. Token většinou vzniká na základě prvotního pravidla, které vzniklo společně s tokenem samotným. Ať už jsou v oběhu tokeny všechny, nebo ne, je ve většině případů předem známé, kolik jich v oběhu bude maximálně a jak se

do oběhu dostávají. V případě Bitcoinu například nové tokeny vznikají při takzvané těžbě, tedy ověřování transakcí pomocí proof-of-work, za což jsou těžaři odměňováni částečně poplatky, které platí ti, kteří transakce zakládají a částečně jsou placeni nově vznikajícími tokeny. Obtížnost proof-of-work se ale s narůstajícím počtem tokenů v oběhu zvyšuje a tím pádem se postupně snižuje rychlost vznikání tokenů nových. Zároveň se dopředu ví, kolik tokenů bude a kdy už těžba nebude přinášet nové. Z tohoto důvodu nedrží moc nad cenou tokenu v rukou jedna instituce, ale cena vychází přímo z trhu na základě nabídky a poptávky.

V případě IOTA tokenu byly všechny tokeny do oběhu vydány hned na začátku v rámci genesis transakce. Tento počet se nikdy nebude měnit a tokeny se dají pouze koupit, nebo získat jinou cestou, ale vždy tak, že jsou zaslány v rámci transakce. Nikdy nemůže vzniknout nový IOTA token. V oběhu je 2 779 530 283 277 761 IOTA tokenů, kterým se jednoduše říká IOTA. Jedná se o číslo shodné s $(3^{33}-1)/2$. Jedná se o největší 33-tritové číslo. Tedy převedeno do ternárního kódování, 33 jedniček v řadě. Jelikož se jedná o opravdu velký počet, je díky tomu možné přenášet opravdu malé finanční částky. V době kdy vzniká tato práce, se cena IOTA pohybuje okolo desítek centů amerického dolaru za 1 000 000 IOTA. 1 IOTA se tedy pohybuje okolo 0,0005 Korun českých. Vzhledem k faktu, že 1 IOTA je v tuto chvíli až příliš malá pro běžné užití, bere se za základní jednotku 1 000 000 IOTA, tedy 1 MIOTA. V rámci IOTA názvosloví se používá mezinárodní systém jednotek (Systém SI). Ukázka jednotlivých jednotek a jejich hodnot je v tabulce 2.1. IOTA jako taková je už dále nedělitelná a nejmenší proveditelná transakce je tedy 1 IOTA.

Jednotka	Název	Hodnota
Pi	Peta IOTA	10^{15}
Ti	Terra IOTA	10^{12}
Gi	Giga IOTA	10^{9}
Mi	Mega IOTA	10^{6}
Ki	Kilo IOTA	10^{3}
i	IOTA	1

Tabulka 2.1: Převody jednotek IOTA

2.2.3.10 Seed a adresy

Specifikum IOTA protokolu není pouze ve vytváření a ověřování transakcí, ale i v samotném přístupu a práci s adresami. V minulém odstavci byl popsán IOTA token a fakt, že jedním z hlavních účelů Tangle je převod tokenů mezi účty. Ať už je účet vlastněn fyzickou osobou a ovládán ručně, nebo je součástí IoT zařízení a práce s tokeny probíhá strojově, musí být dodržován specifický princip, který vychází z metodiky, kterou je pracováno s adresami a účty.

V kryptografii známe asynchronní šifry, které jsou založeny na principu soukromého a veřejného klíče. Veřejný klíč můžeme distribuovat a je používán dalšími stranami, které pomocí něj zašifrují zprávy, které nám posílají. Oproti tomu soukromým klíčem, který z důvodu bezpečnosti nesmíme nikomu poskytnout, jsme schopní přijatou zprávu dešifrovat.[18] IOTA je založena na podobném principu s určitou limitací. V rámci IOTA sítě rozlišujeme něco čemu můžeme říkat například účet, nebo peněženka. Ty na sebe mohou mít navázáno více adres.

Hlavním identifikátorem účtu je řetězec nazývaný seed. Jedná se o řetězec dlouhý 81 znaků složený pouze z velkých písmen abecedy a čísla 9. Seed je jediným identifikátorem účtu a pouze pomocí něj se dá s účtem pracovat. Funguje tedy mimo jiné jako doklad vlastnictví daného účtu. Z tohoto důvodu by měl být vhodně uložen a zabezpečen.

Pro generování soukromých klíčů adres k danému účtu je využíváno speciální schéma založeno na WOTS schématu (Winternitz One-Time Signature). Soukromý klíč je generován ze seedu zkombinovaným s indexem adresy. První adresa má index 0, druhá 1 a tak dále. Soukromý klíč pro první adresu tedy získáme tak, že jako vstup hashovací funkce použijeme seed doplněný o číslo 0. V tomto případě se používá hashovací funkce Kerl, což je další hashovací funkce vyvinutá IOTA Foundation. Je založena na hashovací funkci Keccak secure hash, která se v podstatě stala předlohou funkce SHA3-256.[19] Zahashováním dostaneme privátní klíč. Pokud tento klíč znovu zahashujeme stejnou funkcí, dostaneme k němu příslušný veřejný klíč. V případě, že dojde k vytvoření transakce, která odesílá tokeny na jiný účet, je vlastnictví tokenů doloženo podpisem transakce soukromým klíčem. Vzhledem k využití WOTS je podpisem zveřejněna polovina privátního klíče a tím pádem dochází k 50% snížení bezpečnosti adresy. S každým dalším použitím této adresy by její bezpečnost dále klesala. Z tohoto důvodu je tedy při každé odchozí transakci příslušná část tokenů odeslána na adresu příjemce a zbylé tokeny na adrese jsou poslány na novou adresu, která má index o jedna vyšší než aktuální. Tímto je zajištěno to, že tokeny nebudou zůstávat na neúplně bezpečné adrese. Lze tedy na jednu adresu opakovaně přijímat tokeny, ale jakmile jednou z adresy nějaké tokeny odejdou, je třeba adresu opustit.[14]

Důležitým aspektem je také připojení adresy k Tangle. Nově vzniklá adresa nemusí být k Tangle připojena. To samo o sobě nevadí, jelikož v případě, že vznikne transakce, která posílá tokeny na tuto adresu, bude automaticky připojena. Je ale dobré adresy připojovat vždy hned po vzniku z důvodu algoritmu kterým jsou nalézány všechny adresy propojené s určitým seedem. Takto je například zjišťován objem tokenů na účtu aplikacemi, které nabízejí uživatelské rozhraní k účtu. Postupně se hledají všechny adresy spojené se seedem od indexu 0. Ve chvíli co algoritmus dojde k indexu adresy, který neexistuje, tak se zastaví a posčítá objemy na všech nalezených adresách (Objem by měl být pouze na poslední nalezené adrese kvůli bezpečnosti - jisté to ale samozřejmě není). Je teoreticky ale možné, že adresa s indexem x ne-

bude připojená k Tangle, tím pádem v rámci algoritmu nebude nalezena, ale adresa s indexem x+1 bude existovat, bude obsahovat nějaký objem tokenů a k Tangle připojena bude. Jelikož ale adresa s indexem x není připojena, není objevena a algoritmus k x+1 už nedojde. Toto je důležité pro další odstavec zabývající se principem snapshotů.[20]

2.2.3.11 Snapshot

Vzhledem k tomu, že IOTA cílí na to, aby se stala standardem v rámci Internetu věcí, musí počítat s velkým počtem transakcí, které skrz ní budou protékat. Toto klade nejen velké požadavky na rychlost a dostupnost, ale také na paměť. V kapitole o uzlech bylo řečeno, že plné uzly uchovávají celou kopii Tangle. To by znamenalo, že každý plný uzel musí mít uloženy všechny transakce od genesis až po nejnovější. Po určitém čase by z tohoto důvodu byla paměťová náročnost opravdu velká a provozovat plný uzel by bylo tedy velmi nákladné. Je pochopitelné, že to by se moc jednotlivcům nebo institucím dělat nechtělo a síť by tedy nebyla schopná běhu. Řešením tohoto problému je takzvaný snapshot.

Snapshot je postup kterým lze efektivně snížit velikost Tangle. Jedná se v podstatě o promazání veškerých transakcí, které probíhá v určité periodě. Ve chvíli, kdy proběhne snapshot, jsou všechny transakce odstraněny a společně s nimi i adresy, které neobsahují žádné tokeny. Tím dostáváme aktuální stav Tangle, tedy pouze existující adresy, které v tuto chvíli obsahují nějaké tokeny a samozřejmě i objem tokenů, který obsahují.

V tuto chvíli jsou snapshoty centralizovanou operací. V určité chvíli je od IOTA Foundation připraven stav Tangle k novému snapshotu. Ten je poté schválen plnými uzly, které ověří, jestli stav souhlasí a neobsahuje nevalidní hodnoty a pokud je v pořádku a v rámci komunity plných uzlů dojde ke konsenzu, tak je vydán v rámci nové verze IRI (IOTA Reference Implementation), což je implementace aplikace, která slouží jako plný uzel. Její aktualizace není vynucována a každý uzel, respektive jeho provozovatel, se může rozhodnout, zdali snapshot využije a nebo ne. Jedná se čistě o princip, kterým lze snížit paměťovou náročnost, která je na infrastrukturu uzlu kladena.

Vzhledem k tomu, že snapshot je organizován centralizovaně, je zde, stejně jako u koordinátoru, otázka decentralizace IOTA. Stejně jako je cílem do budoucna koordinátor odstranit, tak by v budoucnu neměl být snapshot organizován centralizovaně, ale měly by probíhat takzvané lokální snapshoty, které budou v režii každého uzlu a budou se týkat pouze jeho kopie Tangle. Možnost lokálního snapshotu je aktuálně ve vývoji a měla by být v budoucnu dokončena. Jedním z hlavních problémů této metody je nalézt v rámci uzlu kompromis mezi velikostí Tangle a frekvencí snapshotů. Pokud budou snapshoty časově moc daleko od sebe, bude Tangle zabírat hodně místa. Pokud ale budou moc blízko u sebe, bude uzel riskovat to, že se jeho verze Tangle odchýlí od platné verze. Jak bylo dříve popsáno, platné transakce vychází z konsenzu,

kterého dosahují všechny uzly, respektive jejich většina. To vyžaduje určitý čas a může se stát, že historická transakce, kterou jeden uzel uznal, je komunitou neuznána a stává se tedy neplatnou. Pokud by ale uzel prováděl snapshoty velmi často, mohlo by se stát, že by transakci smazal ještě před zjištěním, že není platná a dostal by se do stavu, kdy nemá validní Tangle, jelikož objemy tokenů na adresách v něm nejsou správné kvůli uznané neplatné transakci.

Další otázkou týkající se snapshotů je to, jestli mizející historie transakcí není problém. Obecně se o problém nejedná, jelikož Tangle je ve své podstatě bezestavový a důležitý je každý jeho okamžitý stav. Jsou ale určité situace, kdy někdo chce, nebo potřebuje zjistit historii transakcí. I v běžných účetních knihách účetní často koukají na historické transakce, aby například zjistili, kdy komu co platili. Kdyby ovšem každý uzel dělal snapshoty, toto by nebylo možné. Z tohoto důvodu je aktuálně ve vývoji prototyp takzvaného permanode, který má v budoucnu sloužit v podstatě jako plný uzel bez snapshotů. Tedy uzel obsahující celý Tangle od počátku, včetně všech vzniklých transakcí. Motivací pro provozování takového uzlu by mělo být právě to, že v určitých případech je třeba se na historii podívat. To bude umožněno právě provozovatelem permanode a vzhledem k faktu, že provoz takového uzlu bude finančně náročnější z důvodu paměťové náročnosti, dává smysl to, že tyto nahlížení do historie budou provozovatelem permanentního uzlu zpoplatněny. [14]

2.2.3.12 Qubic

Ačkoliv se jedná o projekt, který jde ruku v ruce s IOTA protokolem a byl oznámen až tři roky po vzniku projektu IOTA, podle IOTA Foundation vznikl jeho koncept a myšlenka ještě před samotným IOTA protokolem a společně s ním dávají dohromady původní koncept quorum-based (v rámci textu budu používat originální výraz a nesnažit se o vlastní překlad) tokenu. Kvórum (Quorum) je metoda používaná při distribuovaných výpočtech, která pomocí hlasování zajišťuje atomicitu distribuovaných transakcí. Díky kvóru je tedy možné distribuovat výpočty, aniž by docházelo k míchání výpočtů v různých stavech různých uzlů. Qubic je tedy projekt, který do IOTA projektu a tedy do Tangle přináší distribuované výpočty, které jsou základním předpokladem pro využití protokolu v rámci IoT, z důvodu velkých nároků na energetickou efektivnost jednotlivých zařízení a tedy faktu, že většina zařízení bude mít poměrně nízkou výpočetní kapacitu.

Qubic je založen na třech základních technologiích, kterými jsou Oracle machines, outsorcované výpočty a smart kontrakty. **Oracle machine** je turingův stroj doplněný o černou krabici (black box), která umí řešit dva typy problémů. První rozhodovací, tedy určit, jestli element x je v množině A. A druhý funkcionální, tedy říci, co je f(x). Qubic oracle machines využívá jako rozhraní pro získávání dat z venkovního prostředí, oproti prostředí, kde probíhají výpočty. Orákula umožňují přenášet data a zároveň zachovávat poměrně velkou jistotu ohledně konzistence daných dat. V kombinaci s kvó-

rem, které je používáno v rámci outsourcovaných výpočtů, je zajištěna konzistence výpočtů a dat, které do nich vstupují. Orákula by se v podstatě měla sdružovat do skupin po více, kdy všechny provedou určitý výpočet, případně poskytnou určitá data a v rámci skupiny musí dojít k určité shodě, která vychází z kvóra. V tu chvíli se dá výsledek, nebo data považovat za validní. Existují zároveň i mechanismy, jak zajistit konzistenci dat pouze z jednoho, případně dvou orákul. **Outsourcing výpočtů** je možnost, jak zajistit, že i zařízení v rámci IoT budou schopná řešit složitější výpočetní problémy. Tato zařízení mají většinou velké nároky na to, aby byla velmi energeticky nenáročná a z tohoto důvodu nemohou mít moc velkou výpočetní kapacitu. I tak se ale může stát, že k výkonu své práce budou potřebovat spočítat určité složité problémy. V rámci Qubic mohou jednoduše vytvořit problém, který poté ostatní stroje mohou zpracovat. Jedná se o klasický model producent - konzument. V tomto případě producenti vytváří výpočetní problémy, které si konzumenti rozebírají a za úplatu se podílí na jejich řešení. Qubic protokol ve spolupráci s oracle machines potom zajišťuje, že výsledek těchto decentralizovaných výpočtů je do vysoké míry správný. **Smart kontrakty** (do češtiny překládáno jako „Chytrý kontakt“, ačkoliv se většinou stále používá spíše originální výraz) jsou poslední nutností pro úplné fungování Qubic protokolu. Smart kontrakt je forma softwaru, která k vynucení určitého výsledku na základě nějaké události nepotřebuje třetí stranu. Chytrý kontrakt je na začátku vytvořen a zanesen do prostředí, kde existuje a na základě událostí provádí práci, která mu byla na začátku přidělena, nehledě na okolnostech. Každý tedy může ověřit, že určitá událost vždy vyústí ve specifický výsledek v rámci kontraktu a není možnost jeho fungování zvenku ovlivnit. Je tedy velmi vhodný pro tvorbu decentralizovaného softwaru, jelikož běží v rámci decentralizované sítě, je veřejně dostupný, ale zároveň neovlivnitelný.[21]

Při spojení těchto tří technologií dostáváme qubic, který by měl být předdefinovaný quorum-based výpočetní problém. Tento problém je následně v rámci transakce zanesen do Tangle. Transakce obsahuje specifická metadata, která nesou informace o problému. Na jejich základě se potom orákulum může rozhodnout, jestli chce daný problém řešit, nebo ne. Tyto problémy jsou specifikovány pomocí speciálního ternárního programovacího jazyka zvaného Abra. Tento jazyk je v tuto chvíli ve vývoji v rámci IOTA Foundation. Jedná se o funkcionální jazyk, který by měl být údajně velmi podobný Haskellu. Takovýto typ jazyka je vzhledem k podstatě qubic problémů velmi vhodný, jelikož by se v podstatě mělo jednat o event-driven architekturu. Tedy akce by se měly vyvolávat na základě události, kterou by měla být změna dat na vstupu v Tangle. [22]

Když se spojí Qubic protokol s IOTA protokolem, měl by podle IOTA Foundation vzniknout velký technický ekosystém, který by umožňoval efektivně pracovat s daty a transakcemi v rámci Internetu věcí.

2.2.4 Historie IOTA projektu

Historie projektu IOTA je přímo napojená na projekt Jinn a čtyři osoby, kterými jsou David Sønstebø, Dominik Schiener, Sergey Ivancheglo a Serguei Popov. První dva, tedy David Sønstebø a Dominik Schiener jsou aktivnějšími členy týmu z hlediska komunikace s veřejností, ačkoliv hlavními vývojáři byli Sergey Ivancheglo a Serguei Popov. Kromě Serguei Popova mají všichni historii spjatou s distribuovanými účetními knihami a to především s Bitcoinem postaveném na technologii blockchain. **Serguei Popov** učí matematiku na univerzitě v Brazílii a je kromě mnoha jiných akademických prací autorem i whitepaperu projektu IOTA. Jde tedy o autora hlavních myšlenek a principů IOTA, jako je například využití DAG nebo Monte Carlo náhodné procházky. **Sergey Ivancheglo** je autorem projektu NXT, který umožnil vytvářet vlastní tokeny na blockchainu bez nutnosti programování vlastního protokolu. Zároveň vytvořil startup Jinn, který se zabýval stavbou ternárního mikroprocesoru. **Dominik Schiener** a **David Sønstebø** jsou oba postavami, které se točily kolem více projektů spojených s technologiemi týkajícími se kryptosvěta. Dominik Schiener například spoluzaložil burzu kryptoměn. David Sønstebø byl zároveň jedním ze spoluzakladatelů startupu Jinn.

V rámci projektu Jinn byly vytvořeny JINN tokeny na platformě NXT, které byly následně prodávány a měly majiteli garantovat podíly na zisku. Tímto byl financován chod projektu. Po čase si začali zakladatelé uvědomovat, že samotný hardware je krok příliš dopředu a vznikl projekt IOTA, který má být nutností pro úspěšný vznik Jinn hardwaru. V rámci projektu vznikl token IOTA, který byl nabízen v rámci ICO a zisk měl sloužit jako zdroj financí pro projekt. Veřejnost mohla koupit tokeny buď za Bitcoin, případně vyměnit za JINN tokeny. Během tohoto byly prodány všechny vzniklé tokeny. Vybráno bylo přibližně 526 000 dolarů. Na rozdíl od většiny ostatních projektů si zakladatelé a vývojáři nenechali žádné tokeny, jelikož dle jejich slov chtěli od začátku vytvářet projekt, který bude hnán komunitou. Následně začal samotný další vývoj, během kterého byly vybírány dary v podobě tokenů, které měly financovat další vývoj, jelikož zakladatelé začali zjišťovat, že samotných 500 000 dolarů stačit na dlouhodobější vývoj nebude.

3. listopadu 2017 oficiálně vznikla německá nezisková organizace IOTA Foundation, která je v tuto chvíli oficiální institucí, která za projektem stojí.[23] Pár dní předtím bylo první oficiální nizozemské IOTA setkání, kde Dominik Schiener prezentoval aktuální finanční situaci nadace. Ta dle jeho slov v tu chvíli disponovala přibližně 142 Terra IOTA (v té době kolem 61 milionu dolarů). Tyto peníze sloužily jako finance nadace. Zároveň však vlastnila 22 Terra IOTA, které byly investovány do takzvaného IOTA ekosystému. Ten by měl financovat komunitní projekty, které jsou pro projekt zajímavé.

IOTA Foundation se od té doby zaměřuje na rozvoj projektu. Nabírá nové lidi napříč mnoha obory. Především se však samozřejmě zaměřuje na schopné inženýry v oblasti vývoje a matematiky. Zároveň bylo nastíněno několik dalších

milníků, kterých je třeba v rámci projektu dosáhnout a celkově se nadace soustředila na komunikaci s veřejností. Začal být více spravován blog provozovaný nadací, zároveň byla předělána internetová stránka projektu, která byla doplněna o dokumentace, prezentaci technologie a dalšího rozvoje, stejně tak jako o základ roadmapy, která bude probrána v další kapitole. Souběžně s tím se stále technologicky rozvíjí samotné jádro projektu a vznikají knihovny pro širší spektrum programovacích jazyků. Některé jsou přímo vyvíjeny IOTA Foundation, některé vznikají v rámci komunitní snahy a rozvoj projektu.[24]

2.2.5 Roadmapa projektu

Jak se ukázalo na předchozích částech této kapitoly, IOTA je projekt teoreticky velmi rozsáhlý. Jedná se o poměrně složitou technologii založenou na mnoha matematických konstruktech. Ačkoliv samotné základy IOTA protokolu jsou již položeny, byly popsány v IOTA whitepaperu a část projektu už je v provozu, je stále spousta aspektů, které je potřeba vyřešit a případně implementovat, aby projekt mohl dosáhnout cílů, které si vytyčil. Jelikož IOTA Foundation byla v minulosti častým terčem kritiky kvůli nedostatečné průhlednosti, při vývoji nového webu projektu byla jednou z domén na kterou se zaměřila právě roadmapa. Výsledek je však poměrně vágní a naprosto zapomíná jakékoliv časové odhady a popis technologií je zároveň poměrně strohý. Jako lepší roadmapa by mohl sloužit příspěvek na IOTA blogu z března 2017. Na následujících řádcích budou probrány jak nové roadmapy, tak starší z roku 2017, kterou z důvodu časové posloupnosti začneme.

První ucelená roadmapa vyšla v rámci příspěvku na IOTA blogu, jehož autorem byl David Sønstebø. Datum kdy článek vyšel je 31. března 2017. V rámci roadmapy se rozebírá nejprve to, jak důležitá je pro úspěch IOTA projektu její adopce v rámci světa. Za tímto účelem je ve vývoji několik implementací IOTA uzlů postavených na různých jazycích, což má zjednodušit jejich provoz a použití. Java klient je hotový a běží. Ve vývoji byl v té době C++ klient, který byl dle IOTA Foundation velkou prioritou. Tento klient je v tuto chvíli v beta verzi. Ve velmi rané fázi vývoje byly implementace v jazycích Rust a Go. O těchto implementacích a jejich aktuálním stavu se v tuto chvíli prakticky neví.

Další čím se roadmapa zabývá jsou alternativní klienti. Zde prezentuje v té době již hotového a veřejně dostupného lehkého klienta, který umožňuje provozování lehkého uzlu. Druhým alternativním klientem je swarm klient, který je ve fázi raného výzkumu. Tento klient by v budoucnu měl umožňovat provoz jednoho uzlu na více zařízeních, pomocí distribuce hlavní logiky a databáze. To by mělo umožňovat clusteru více zařízení efektivní spolupráci a vytváření transakcí bez větší zátěže jednotlivých zařízení.

V další sekci jsou prezentovány nová specifika Tangle. Jedním je zlepšení síťování. Především umožnění přepínat mezi jednotlivými síťovými protokoly a zároveň jednodušší odhalování jednotlivých uzlů, což by mělo přispět

k výkonnější síti. Tato technologie je ve stavu aktivního výzkumu. Druhou technologií je potom automatické snapshotování Tangle. Jedná se o posun od snapshotu manuálně vytvářeného IOTA Foundation, který byl prezentován v předchozích kapitolách. V tomto případě je už část implementace hotová a v testování. Zbytek je zároveň v aktivním vývoji.

V rámci roadmapy je dále poukazováno na fakt, že IOTA je stavěna jako projekt s důrazem na jeho lehkost. Tedy tak, aby se nejednalo o velký projekt se spoustou funkcionalit a v důsledku toho s průměrnou výkonností, ale aby byl jeden jednoduchý klient a další jednoduché moduly, které se dají skládat a zároveň si udržují svoji jednoduchost a na základě toho svou velkou výkonnost. V rámci těchto modulů je prezentován například modul na **identitu věcí**, který je ve fázi výzkumu. Jedná se o aspekt IoT, kdy z důvodu bezpečnosti by každé zařízení v rámci sítě mělo mít svůj identifikátor a zároveň k němu napojené určité atributy, které nesou informace o daném zařízení. Tyto data budou zanesena přímo v síti a vždy tedy bude možné určit, jestli se dané zařízení chová tak jak má. Dalším modulem je **permanentní uzel**, který byl prezentován už v odstavci o snapshotu. Ten je ve fázi hlubšího testování. Dalšími moduly jsou například **autentizované zprávy**, které umožňují šifrovat datový tok a umožnit k němu přístup pouze určitým osobám. Ten se nacházel v době roadmapy ve stavu testování a nyní už existuje jako veřejně dostupná knihovna. Nebo například modul na **privátní transakce**, které jsou v podstatě to samé, jako autentizované datové zprávy, ale týkající se transakcí. Zároveň se tato část okrajově a velmi zkratkovitě zmiňuje o **orákulech** (oracles), která mají být ve fázi pokročilého vývoje. V rámci roadmap jsou většinou zmiňována velmi okrajově a pouze se základními informacemi, ačkoliv se jedná o poměrně důležitou technologii. Jako taková byla probrána více společně s projektem Qubic, který bude popsán dále v roadmapě.

Dále roadmapa rozebírá klientské knihovny, které umožňují nad IOTA technologií vyvíjet. Zde prezentuje 5 knihoven a to JavaScript, Java, Python, C# a Go. Kromě Go byly v té době všechny vyvinuté a veřejně dostupné v beta verzích, které byly dále rozvíjeny. Aktuálně je v tomto stavu už i knihovna v jazyce Go.

V závěru se tato roadmapa zabývá ekosystémovými tématy, jako je zátěžové testování, simulované využití Tangle, testovací prostředí a výukové materiály. Zde se jedná o dlouhodobější proces a například testovací prostředí už běží. Výukové materiály ve verzi na které odkazuje tato původní roadmapa už neexistují, ale mezitím vznikly nové.[25]

Po této původní roadmapě vznikla druhá v půlce roku 2018. Tato roadmapa byla a stále je součástí webových stránek IOTA projektu. Jedná se o **roadmapu výzkumu a vývoje** a opět je poměrně strohá a bez časových informací. Obsahuje informaci, že má být postupně aktualizována s vývojem produktu. Informace o tom, kdy naposledy byla tato roadmapa aktualizována, je ovšem nedohledatelná. V rámci dokumentu jsou funkcionality rozděleny na

dvě hlavní sekce - na projekty, které jsou ve fázi výzkumu a ty, které jsou ve fázi vývoje.

Ve fázi výzkumu se aktuálně nachází projekt na odstranění koordinátoru, o kterém bylo psáno v jeho vlastní části rešerše. Zde se jedná primárně o simulace a matematické modely. Dalšími projekty jsou detekce a prevence spamu v rámci sítě a automatické objevování uzlů. Jedním z důležitých projektů v rámci výzkumu je studium ekonomických stimulů v rámci Tangle. Jedná se o simulace s využitím teorie her, které jsou prováděny za účelem lepšího pochopení chování v Tangle a na základě toho schopnost lepšího odhadu chování samotné sítě, nároků na ní a dalších věcí s tím spojených. Zároveň vznikají specifikace algoritmu na dosažení konsenzu v rámci Tangle a specifikace kryptografie, která je IOTA protokolem využívaná. Obě tyto specifikace mají za cíl umožnění kontroly těchto metod veřejností a případný audit. V poslední řadě se aktuální výzkum zaměřuje na jedno z hlavních témat spojených s decentralizovanými účetními knihami a to zabezpečení před možnými útoky na síť.

V rámci projektů, které jsou v tuto chvíli ve fázi vývoje, se zaměřím na několik důležitých z hlediska ekosystému. Jedním z nich je takzvaný IOTA hub, který umožňuje napojení burzy na Tangle jednoduchým způsobem, bez nutnosti vlastní složité implementace, čímž značně snižuje čas, za který může být IOTA v rámci burzy obchodovatelná. Ačkoliv toto není hlavním účelem IOTA projektu, značně to pomáhá adopci technologie. Za účelem zlepšení adopce zároveň vznikla peněženka Trinity Wallet, která je stále dále rozvíjena. Jedná se o grafické rozhraní nad lehkým uzlem, které umožňuje jednoduše vytvářet a spravovat transakce. Další zajímavou technologií je takzvaný POW-Box, který by měl umožňovat převést proof of work na externí zařízení, například malým zařízením, které by samy nemusely být schopné v přijatelné době POW splnit. Mimo jiné vývoj ještě probíhá právě na lokálních snapshotech, permauzlech, odstranění koordinátoru, knihovně v jazyce C a nebo například vývoj nové, ještě vhodnější hashovací funkce. V poslední řadě je v této roadmapě zmíněn s pár popisnými větami projekt Qubic. Tento projekt je ovšem pro IOTA projekt velmi důležitý a z tohoto důvodu dostal dokonce svou vlastní webovou stránku a bude poslední částí roadmapy, která bude v rámci rešerše probrána. [26]

Poslední roadmapou je roadmapa projektu **Qubic**, o kterém bylo psáno v jedné z předchozích kapitol. Roadmapa opět nedává prakticky žádné časové odhady a sama přímo uvádí, že se jedná o poměrně dlouhý výhled do budoucnosti a věci se mohou ještě měnit. Zároveň je údajně ještě spousta myšlenek a nápadů, které do roadmapy nejsou vůbec zaneseny. První část se zabývá otázkou nového funkcionálního ternárního jazyka Abra. Zde se mluví převážně o vzniku specifikace jazyka a kompilátorech Abra do ASM86 a Verilog, tedy především do efektivních jazyků, které mohou běžet přímo na hardware. Další sekce se zabývá samotným qubic protokolem, u kterého aktuálně vzniká specifikace a dokument týkající se matematické části qubic. Dále jsou v road-

mapě zaneseny části týkající se distribuovaných výpočtů a orákul. Jedná se například o časové známky, odměny za výpočty, úpravy transakcí, nebo práce s externími daty a kvórem.[27]

Jak je vidět, projekt je ve velmi rozpracované fázi a má před sebou ještě spoustu práce, než dosáhne všeho, co bylo v rámci něj vytyčeno. Trochu problémem může být fakt, že roadmapy nemají časové odhady. Většinou ani neobsahují aktuální stav v rámci dané technologie a celkově je průhlednost projektu poměrně slabá. Ačkoliv do určité míry probíhá komunikace v rámci sociálních sítí, blogu IOTA Foundation a na Discord kanálu IOTA Foundation, stále by bylo v rámci projektu dobré, kdyby komunikace probíhala na pravidelné bázi a bylo možné vidět, co se měsíc po měsíci změnilo.

2.3 Shrnutí kapitoly

V rámci kapitoly byla představena podstata Internetu věcí a základ samotné technologie internetu. V budoucnu samozřejmě není nutné, aby Internet věcí byl založen na současné technologii internetu. Může se jednat o novou síť založenou na naprosto jiném transportním protokolu. Jednou z těchto technologií je IOTA protokol, která je agnostická co se transportního protokolu týče a sama decentralizovanou formou zajišťuje uložení dat, jejich distribuci a především potom funguje jako decentralizovaná účetní kniha a umožňuje tedy převod financí a to se schopností až nanotransakcí. Tato technologie je založena na mnoha matematických a dalších principech a v podstatě je poměrně rozsáhlého charakteru. V rámci rešerše byly představeny čtenáři základní technologie a principy o které se protokol opírá a následně představen budoucí plánovaný vývoj protokolu.

Následující analytická kapitola bude na rešeršní navazovat, avšak z podstaty věci se bude s rešerší stále prolínat. Při řešení efektivity protokolu a jeho vhodnosti pro využití v rámci ekosystému Internetu věcí bude třeba probrat určité domény z rešerše ještě do větší technické hloubky. Rešerše jako taková sloužila hlavně k pochopení samotné podstaty protokolu, jeho rozsahu a jednotlivých problémů, které se v rámci něj řeší.

Kapitola 3

Analytická část

3.1 Dopady a problémy Internetu věcí

Jak lze vyčíst z rešeršní části zabývající se otázkou definice Internetu věcí, jedná se o velmi komplexní systém, který má v nejhlubším důsledku dopad prakticky na celý svět. Jedná se o velmi převratný koncept, který posune veškeré bytí na jinou úroveň. Dopad na každodenní život bude obrovský. Díky propojení prakticky všeho v rámci jedné sítě se služby posunou na lepší úroveň. Dostupnost produktů bude vyšší, jelikož díky lepší práci s daty a vzájemnému propojení systémů, bude jednodušší sledovat pohyb a distribuci zboží, stejně jako půjde lépe sledovat mnoho jiných metrik. Lidé by v důsledku toho měli šetřit čas na denních činnostech, které odpadnou, jako je například nákup potravin. To zvýší kvalitu života, jelikož se lidé budou moci věnovat činnostem na které doteď neměli tolik času. V návaznosti na to vzniknou samozřejmě nové sociální problémy, které nyní neexistují. Nedá se zde popisovat, co vše se v důsledku celosvětového propojení změní, jelikož v podstatě se opravdu změní prakticky vše.

Internet věcí ovšem čelí i spoustě výzev a problémů, které musí, jako každá nově vznikající technologie, překonat, aby mohlo být vše tak pozitivní, jak bylo naznačeno v minulém odstavci. Kromě mnoha technologických problémů a výzev, které byly probrány v rešerši a v další kapitole na ně bude navázáno při analýze protokolu IOTA, jsou zároveň problémy spjaté se samotným ekosystémem a celkovou připraveností světa a lidí na tuto technologii. Samotná propojenost je faktorem, který umožňuje spoustu nejen pozitivních věcí, ale i negativních. Každý útok na síť bude mít o to větší dopad na lidstvo. To může při bezpečnostních chybách potenciálně otevírat bránu nové formě globálního kybernetického terorismu. Další oblastí, která se velmi aktivně v poslední době probírá, jsou etické otázky umělé inteligence. Internet věcí je z velké části na umělé inteligenci a autonomních strojích založen. S tím ale opět přichází problémy v oblasti etiky a bezpečnosti umělé inteligence, které je třeba řešit. Vývoj jde velmi rychle dopředu, avšak tyto otázky se zatím

nestíhají přizpůsobit.

V důsledku je k Internetu věcí třeba přistupovat velmi obezřetně a vše promýšlet velmi do důsledku, jelikož se jedná o tak globální věc, která má tak velké dopady, že je třeba velmi se zamýšlet nad tím, na co je celý svět a lidstvo připraveno a na co ještě není. Vývoj technologie je věc jedna, ovšem vývoj lidské existence a její vyzrálost je věc druhá.

3.2 Vhodnost IOTA protokolu v rámci IoT ekosystému

Jak bylo řečeno již v rámci shrnutí rešerše, analýza bude velmi úzce provázaná s rešerší a spoustu témat bude muset být probráno o něco více do hloubky, než tomu bylo přímo v rámci rešerše. V této kapitole bude zanalyzován IOTA projekt z hlediska jeho vhodnosti pro využití v rámci Internetu věcí a zároveň budou představeny možné případy užití, které tato technologie umožňuje. Samotná analýza spojená s implementací, tedy analýza vývoje na tomto protokolu a analýza nově vznikajícího systému, který je předmětem implementace, bude probrána až v rámci třetí - implementační části. Důvodem pro toto je striktní oddělení implementační analýzy a protokolové.

3.2.1 Vhodnost IOTA vzhledem k požadavkům Internetu věcí

Při analýze, jejímž výsledkem by mělo být zhodnocení, jestli je IOTA jako technologie vhodná pro využití v rámci Internetu věcí, je třeba zaměřit se na jednotlivé požadavky, které klade Internet věcí na technologie, na kterých by měl být postaven. Tyto požadavky byly rozebrány v kapitole 2.1.2.1. Ne všechny tyto požadavky jsou kladeny přímo na IOTA protokol. Ten jako takový je cílený primárně jako hlavní technologie pro strojovou ekonomii, tedy možnost převodu financí mezi jednotlivými zařízeními v síti. Nejedná se tedy například o transportní protokol, který by se staral o fyzický přenos dat. IOTA je v rámci této vrstvy agnostická a samotná data mezi uzly mohou být tedy přenášena jakoukoliv technologií, která bude vzhledem ke své výkonnosti vhodná pro Internet věcí, ať už se jedná o mobilní síť 5G, nebo například novou verzi Bluetooth. Každopádně ale jsou určité nároky na síť, kterou IOTA ovlivnit nemůže, jelikož je sama závislá na další protokolové vrstvě.

V rámci této části analýzy budou probrány jednotlivé domény, v kterých byly požadavky definovány v rámci kapitoly 2.1.2.1.

3.2.1.1 Bezpečnost

Jak už bylo výše popsáno, bezpečnost je prakticky nejdůležitějším požadavkem ze všech. Je jedno, jak rychlá nebo škálovatelná síť je, pokud je napadnutelná a data v ní tedy nejsou bezpečná. Samotné napadnutí sítě a její ovládnutí je

jedna z bezpečnostních hrozeb, neméně zásadní je ale bezpečnost samotných přenášených dat. Velmi často je třeba komunikovat zabezpečenou cestou tak, aby potenciální útočník nemohl získat přístup k datům, která si mezi sebou dvě strany posílají. Zároveň je třeba dbát na samotnou integritu dat. Ať už se jedná o to, že někdo nezaměnil data na cestě, tak například o to, že někdo nevytvořil data, která by nebyla validní. Toto vše dohromady dává velké nároky na bezpečnost využívané technologie a v rámci IOTA protokolu je toto ještě umocněno faktem, že protokol by měl být využíván hlavně k převodu financí, kdy bezpečnost samozřejmě hraje roli znatelně větší.

Když se zaměříme na možné útoky, tak jedním z hlavních útoků, které je třeba v rámci sítě řešit je takzvaný **double-spending útok**. Jedná se o situaci, kdy útočník vytvoří transakci, převádějící tokeny z adresy A na adresu B. Tato transakce je sítí potvrzena a obecně uznána jako provedená. V tu chvíli ale útočník zvrátí výpočetní kapacitou Tangle a donutí síť uznat jako validní stav, ve kterém tato transakce není a naopak tokeny z adresy A jsou posílány na adresu C. V takovém případě může dojít například k podvodu při nákupu. Útočník si od někoho objedná určitý produkt a peníze pošle pomocí Tangle. Prodejce, který je vlastníkem adresy B zjistí, že transakce je s vysokým procentem uznána jako provedena a produkt tedy odešle. Nějaký čas poté ale útočník znevalidní původní Tangle a na adrese B tedy ve výsledku tokeny nebudou. Jsou dva hlavní útoky, kterými lze dosáhnout konsenzu, v rámci kterého je za validní Tangle považován ten útočníkem podvržený. Jedná se o takzvaný Parasite chain, jehož cílem je vytvořit vedle vlastní Tangle a následně donutit nové transakce z hlavního, aby začaly potvrzovat transakce z parazitního. Druhým je takzvaný splitting attack, kdy se původní Tangle rozdělí na dvě části, které se udržují najednou. V rámci parasite chain útoku je hlavním aspektem výpočetní kapacita potřebná pro podvržení Tangle. V dubnu 2018 vyšla vědecká práce, jejímž autorem je Quentin Bramas. Tato práce nese název The Stability and the Security of the Tangle a jejím hlavním tématem byla právě obrana Tangle proti double-spending útoku. V rámci práce bylo matematicky dokázáno, že **pro všechny dosud známé algoritmy pro výběr transakcí k potvrzení musí být hashovací kapacita větší než hashovací kapacita útočníka**.[28] Toto je jedním z důvodů, proč je v této rané fázi třeba koordinátoru. Samotné uzly nemají dostatečnou kapacitu, aby generovaly dostatečně velkou výpočetní sílu, která by pro jednoho útočníka nebyla nepřekonatelná. Jedná se o poměrně velký problém, který bude potřeba vyřešit. Jednou z možností je přijít s lepším algoritmem na výběr transakcí k potvrzení. Jedná se však o velmi složitou možnost, ne-li nemožnou. Druhou možností je ta, ve kterou doufá IOTA Foundation, kterou je adopce projektu a tedy organický nárůst uzlů, který dostatečně navýší výpočetní sílu uzlů s dobrými úmysly. Každopádně samotný útok je velmi těžce proveditelný, protože kromě samotné výpočetní síly je potřeba pokrýt dostatečnou část uzlů, kvůli čemuž musí znát útočník velkou část síťové topologie. V rámci Tangle je síťová topologie mesh(mřížka). Odhalit jednotlivá propojení uzlů

tedy rozhodně není snadné. V každém případě pro dostatečnou bezpečnost před tímto útokem je třeba, aby všechny uzly měly dostatečnou výpočetní kapacitu. V opačném případě je možné útok úspěšně provést. V případě splitting útoku musí útočník rozdělit Tangle na prakticky shodné poloviny a na konec každého umístit transakci, které se navzájem vylučují. Tím zajistí, že distribuce potvrzení bude rovnoměrně rozprostřena mezi dvě nově vzniklé části a útočník bude moci v obou polovinách opakovaně provádět double-spending útoky. Problémem pro útočníka je ovšem to, že rozdělit Tangle na dvě části s téměř shodnou kumulativní vahou je problém s velmi složitým řešením, který se v reálném čase nedá řešit. Jak lze vidět, provést úspěšný útok na Tangle možné je. Problém je však nutná výpočetní kapacita, kterou je útočník nucen disponovat. V aktuální době, kdy je Tangle poměrně malý, není tato kapacita příliš vysoká a síť je tedy náchylná útokům. Těm ale v tuto chvíli zabraňuje koordinátor. Ten by měl být v budoucnu odstraněn, až bude síť dostatečně velká na to, aby byla dostatečně bezpečná i bez něj. I poté bude však možné Tangle napadnout, ovšem s velmi malou pravděpodobností úspěchu.[7]

V rámci **bezpečnosti a konzistence přenášených dat** vyvinula IOTA Foundation modul pro autentizované zprávy zvaný **MAM** (Masked Authenticaded Messaging). Význam tohoto modulu byl ve stručnosti probrán už v rámci rešerše. Tento modul je založen na principu hašového (Merkleova) stromu. Jedná se o kryptografickou strukturu, která umožňuje jak zabezpečení samotné zprávy, tak kontrolu její integrity. Základní myšlenkou je to, že v každém uzlu v rámci stromu je hash zřetězení hodnot jeho dětí.[29] Kořenem stromu je identifikátor komunikačního kanálu, každá zpráva se poté stává kořenem stromu nového. Zpráva je zašifrována jednorázovou tabulkovou šifrou obsahující id kanálu a index klíče použitého k podpisu dané zprávy. Zpráva je následně zahashována a hash je podepsán privátním klíčem jednoho z listů stromu. Zašifrovaná zpráva, podpis a sourozenci listu použitého k podpisu jsou poté nahrány na Tangle. Zde může být zpráva přečtena kýmkoliv, kdo vlastní příslušný veřejný klíč. Za účelem ověření integrity je nejprve validován podpis, kdy je třeba ověřit, že patří jednomu z listů, pokud ne, tak je zpráva nevalidní.[30] MAM je modul založený na velmi bezpečné metodě, především díky využití jednorázové šifrovací tabulky, která umožňuje dokonalé utajení (perfect secrecy). Z tohoto důvodu je velmi vhodná pro využití v rámci Internetu věcí, kde budou proudit velká množství dat a je potřeba mít možnost omezovat přístup k nim. Vzhledem k využití Merkleova stromu je zároveň možné pracovat s různými přístupy zabezpečení, ať už s využitím klíče, nebo pouze omezení viditelnosti. Zároveň je například možné omezit viditelnost zpráv tak, že každý vidí pouze zprávy dále od první zprávy, kterou přeložil.

Jedním s dalších důležitých faktorů v rámci bezpečnosti je odolnost proti kvantovému počítání. Zde jsou často v rámci projektu používány zavádějící slova, jako je například slovo odolnost. Faktem je, že žádná technologie nikdy není plně odolná proti hrubé síle. Vše je pouze otázkou času, kterého je potřeba k vyřešení daného problému. V případě IOTA je odolnost větší než

u běžných technologií, díky ternárnímu systému a hashovacím funkcím, které byly popsány v rešerši. Problémem však je, že v tuto chvíli jsou pouze odhady, jaké mohou být výpočetní kapacity v rámci kvantových počítačů a je tedy těžké hodnotit odolnost proti jejich výpočtům. V každém případě lze říci, že IOTA protokol je na tom se zabezpečením proti kvantovým strojům o mnoho lépe, než většina současné technologie. Neodvážil bych se ale prohlásit, že je proti těmto výpočtům odolná, ačkoliv je tak často prezentována.

Poslední složkou bezpečnosti, která je v rámci Internetu věcí velmi důležitá je integrita dat. Toto by v budoucnu mělo být řešeno orákuly v rámci projektu Qubic, o kterém bylo psáno v rámci rešerše. Orákula by měla sloužit jak k provádění výpočtů, tak jako zdroje dat z vnějšího prostředí. Oba tyto typy dat jsou pro určité strany důležité a potřebují proto, aby byla zajištěna jejich integrita. Tedy aby byla data důvěryhodná. Toho lze docílit právě spojením orákul. Jak již bylo řečeno, qubic je určitý výpočetní modul, který je zodpovědný za určitou práci. Jako takový tedy vytváří úkoly, ať už se jedná o úkol na získání dat, nebo na výpočet. Tyto úkoly poté zpracovávají sdružení více orákul. Vlastník qubicu určí, jaké kvórum potřebuje k tomu, aby data akceptoval jako důvěryhodná. Orákula poté vypočítají daný problém, nebo vrátí zvenčí požadovaná data a qubic zjistí, jestli bylo dosaženo konsenzu, tedy jestli se na jednom specifickém výsledku shodlo více orákul než jaký byl limit daný konsenzem. Pokud ano, tak jsou orákula, která dala dobrý výsledek odměněna tokeny. Vlastník qubicu se tedy rozhoduje, jak důležitá je pro něj integrita dat a dle toho určuje kvórum. V případě, že je pro něj velmi důležitá, může například určit, že se na stejném výsledku musí shodnout alespoň 80% orákul, v opačném případě může říct například pouze 20%. Je třeba však pamatovat, že za vyšší kvórum musí následně platit více orákulům a je tedy pro něj tato jistota dražší. Orákula jsou tedy motivována dávat validní výsledky tím, že v opačném případě nedostanou tokeny. Každé orákulum zároveň vrací výsledek tajně, aby se předešlo tomu, že by jedny mohly výsledky kopírovat od druhého, jenom za účelem dosažení kvóra, bez reálné důvěryhodnosti dat. [31] Toto je velmi důležitou složkou fungování, jelikož celý Internet věcí by měl být založen na výměně dat mezi jednotlivými zařízeními. Pokud nebude možné zajistit důvěryhodnost dat, pak celý ekosystém může zkolabovat. Vzhledem k tomu, že by v ideálním případě na síť měla být připojena prakticky všechna zařízení a většina z nich by se měla autonomně rozhodovat na základě dat z dalších senzorů, jednalo by se opravdu o velký problém, kdyby se datům nedalo věřit. Výpočty založené na kvóru jsou zajímavým a zároveň velmi vhodným způsobem, jak tuto důvěryhodnost zajistit. Samozřejmě se nedá zajistit úplně. Respektive dá, ale dosáhnout kvóra 100% bude velmi obtížné. Často ale není až takto vysoké procento potřeba. Je důležité mít i možnost určit úroveň důvěryhodnosti vzhledem k tomu, o jak důležitá data, nebo o jak důležitý úkol se jedná. Toto přesně qubic projekt, který je součástí, nebo velmi úzkým doplňkem IOTA projektu, umožňuje.

Jak lze z odstavce vyčíst, IOTA a přidružené protokoly jsou z hlediska

bezpečnosti na poměrně dobré úrovni. Stále jsou otázky, které je třeba vyřešit a jedním z důležitých faktů, které mohou vyvolávat určitou skepsi, je to že **síť je bezpečnější s její rostoucí velikostí**. Toto je na jednu stranu **nepříjemnost s kterou je potřeba počítat při využití sítě v době, kdy ještě není její velká adopce**, ale naopak se jedná o **velmi dobrou vlastnost v rámci budoucí vize, kdy má být síť velmi velká**. **Jedním z pozitiv je také využití Merkleova stromu pro zabezpečení přenášených dat a quorum-based výpočtů v kombinaci s orákuly, které zajišťují dobrou integritu dat.** V obou případech se jedná o vhodné modely, které zaručují dostatečnou bezpečnost a integritu. Obecně se tedy dá říci, že úroveň bezpečnosti, kterou IOTA nabízí je na dobré úrovni, ale jelikož se jedná o velmi mladou technologii, musí na ní být dále pracováno. Zároveň je velmi závislá na adopci této technologie a není tedy vhodná pro menší využití.

3.2.1.2 Propojitelnost

Propojitelnost, neboli často používanější výraz interkonektivita v rámci Internetu věcí, je jednou z dalších důležitých otázek a překážek, které je potřeba překlenout při cestě za globálním použitím této technologie. Aktuální stav je takový, že je mnoho technologií, pomocí kterých je vyvíjen hardware a software zařízení. Internet jako takový je potom síť propojující jednotlivé servery. Každý kus dat je tedy uložen na jednom specifickém serveru ke kterému se musí přistupovat určitým způsobem na základě technologie, kterou používá. Toto vytváří v globálním hledisku velký problém, jelikož v budoucnosti, kdy by na Internetu věcí měly být připojeny miliardy zařízení, by mělo každé zařízení problém s přístupem k datům zařízení jiného. Musely by uchovávat informace o jednotlivých způsobech přístupu, což by opět navyšovalo jejich složitost. Data by se složitě sdílela, jelikož by každé zařízení posílalo data na určitý server. Pokud by tedy zařízení A chtělo získat data od zařízení B, muselo by nějakým způsobem od B zjistit, kam data ukládá, následně zjistit, jakou technologii úložiště B používá a na základě toho navázat vhodným způsobem komunikaci. Toto je samozřejmě velmi extrémní představa, která by nikdy fungovat nemohla a možností, jak ji řešit je mnoho. V rámci této práce je důležité, jak vhodně tento problém řeší IOTA.

IOTA Foundation cílí na to, aby se IOTA jako taková stala standardem pro Internet věcí. To samo o sobě říká, že by komunikace měla být jednoduchá, jelikož v rámci standardu by většina zařízení používala tuto technologii. Otázkou však je, jak se IOTA snaží dosáhnout toho, aby se standardem stala.

V první řadě je důležitý fakt, že Tangle je distribuovaná síť. Z tohoto důvodu je sdílení dat velmi jednoduché, protože **v úplné podstatě, by každý uzel měl mít všechna data**. Toto je samozřejmě představa lehce extrémní a ve skutečnosti tomu tak nebude, především z důvodu náročnosti udržovat celou kopii Tangle. Rozhodně ale v rámci distribuce je sdílení dat jednodušší a zařízení se nepotýkají s problémy, že ani neví, kde některá data

jsou. Uzly samotné tvoří **mřížkovou topologii, která umožňuje jednoduchou a efektivní distribuci dat v celé síti**. Jedním z problémů, které je ještě třeba vyřešit je permanentní ukládání dat v Tangle. V rešerši byl popsán proces snapshotu, který z důvodu paměťové efektivity přemaže celou historii Tangle a zanechá pouze aktuální stav. V takovém případě ovšem zmizí i veškeré transakce, které obsahovaly důležitá data. Z tohoto důvodu je aktuálně ve vývoji technologie permanentního uzlu (Permanode) a swarm uzlů. Tyto dvě technologie byly zmíněny v části rešerše zabývající se roadmapou. Permanentní uzel je takový uzel, který uchovává veškerou historii Tangle a s ní tedy uchovává i všechna data. Swarm uzly jsou poté možnost, jak více jednoduchými zařízeními složit jeden uzel, který má větší výpočetní kapacitu. Ty by potom mohly sloužit jako permauzel složený z menších, jednodušších zařízení. Tato technologie je ovšem stále ve vývoji a kdy bude představena se neví. **V tuto chvíli tedy permanentní ukládání dat přímo v rámci Tangle není možné**, což značně limituje právě propojitelnost.

Co však propojitelnosti pomáhá a je jedním z velmi důležitých faktorů je to, že si IOTA Foundation uvědomuje, jak důležité je vytvořit framework, použitelný skrz knihovny více různých jazyků. Toto je jedna z věcí bez kterých by se daná technologie nemohla stát standardem. Na základě toho **aktuálně vzniká, nebo je minimálně zaneseno do roadmapy, že by vzniknout mělo několik knihoven v různých jazycích**, které umožňují komunikaci s Tangle. V rámci rešerše byly tyto knihovny vyjmenovány. Zároveň vedle toho vzniká i mnoho komunitních knihoven v dalších jazycích, kterými se samotná IOTA Foundation nezabývá, jako je například knihovna v jazyce PHP.

Dalším vhodným krokem v rámci propojitelnosti je modul prezentovaný v rámci rešerše a to **modul na identitu věcí**. Ten díky metadatům o jednotlivých zařízeních a jednoznačném identifikátoru každého jednotlivého zařízení umožní lepší kontrolu nad jejich chováním a zároveň i zjednoduší komunikaci zařízení v síti, především proto, že pro ně bude jednodušší se o sobě vůbec dozvědět a získat základní informace o tom, co je které zařízení zač, na základě čehož se budou moci jednoduše rozhodnout, jestli spolu komunikovat chtějí a případně proč, nebo jak.

V rámci otázky propojení má před sebou IOTA ještě dlouho cestu, kterou musí urazit, aby byla vhodná jako standard. Všechny problémy, s propojením spojené, které aktuálně IOTA má, jsou řešeny nějakou z technologií, které jsou zaneseny do roadmapy. Otázkou však opět je, kdy budou vznikající řešení vydány. **Pokud ovšem vše v roadmapě vznikne, dá se IOTA protokol a tedy síť Tangle považovat za velmi vhodnou z hlediska interkonektivity zařízení v Internetu věcí.**

3.2.1.3 Rychlost a škálovatelnost

V budoucnu, kdy by měl Internet věcí obsahovat desítky miliard připojených zařízení, bude na síť kladen velký nárok z hlediska rychlosti. Spousta senzorů

bude generovat data každou sekundu a ty odesílat do sítě. To vše bude muset běžet dostatečně rychle, aby se další zařízení k datům dostala co nejrychleji, jelikož na ně budou potřebovat okamžitě reagovat v rámci svých rozhodovacích procesů. Do toho všeho zároveň budou vznikat transakce, které budou přenášet finance a jelikož by IOTA chtěla být primární technologií pro M2M ekonomii, musí se předpokládat, že bude vznikat mnoho mikrotransakcí, které budou také muset být zaznamenány a potvrzovány téměř okamžitě. Toto vše vytváří velmi náročně požadavky na funkčnost sítě.

Aktuální čas potvrzení transakce je závislý na koordinátoru. Dokud koordinátor nevytvoří nový milník, nemohou být transakce potvrzeny. V tuto chvíli vzniká nový milník každou minutu, což je velice neúnosný čas pro M2M ekonomii. Opět je zde tedy situace, která ukazuje, že **bez odstranění koordinátoru není možné, aby IOTA technologie naplnila své cíle a svůj potenciál**.

Podle IOTA Foundation má IOTA potenciál dosáhnout na miliardy transakcí za sekundu. S tímto by měly pomoci swarm uzly, které budou schopny velkých výpočetních kapacit a zároveň ekonomické rozdělení Tangle na více nezávislých sítí, které nebudou nuceny být neustále propojovány. Jednotlivé uzly tedy nebudou muset zpracovávat všechny transakce na celém Tangle, ale pouze v rámci svého segmentu na kterém pracují. Stejně tak budou existovat lokální subtangle, které nebudou potřebovat být v synchronizaci s hlavním, jelikož jejich využití je například pouze v rámci jedné společnosti.

Toto vše jsou bohužel pouze dohady. Faktem je, že samotná podstata IOTA je myšlenka, že **každá nově vznikající transakce musí potvrdit dvě předešlé**. Toto je také jednou z hlavních myšlenek na které IOTA Foundation staví a společně s faktem, že transakce jsou bezpoplatkové, jsou toto hlavní dvě myšlenky, kterými poukazuje na vhodnost řešení. Jelikož ale IOTA není transportní protokol, je třeba myslet i na to. **Ze samotné teorie je samozřejmě možné říci, že IOTA může škálovat do nekonečna.** Čím více transakcí vzniká, tím rychleji se původní transakce potvrzují. Narážíme ovšem na problém, že **tuto teorii musí podporovat i infrastruktura**. Všechny transakce musí projít uzlem, který má také pouze určitou výpočetní kapacitu. V tuto chvíli v rámci testů bylo naměřeno, že uzly průměrně zvládají maximálně 1000 transakcí za sekundu. Toto chce samozřejmě IOTA Foundation řešit pomocí swarm uzlů. Stále se však jedná o koncept, který nebyl vytvořen a je ve fázi výzkumu a vývoje. Dalším na co IOTA poukazuje je to, že až bude vyvinut ternární procesor Jinn, výpočetní kapacita nepředstavitelně vzroste. Především díky tomu, že Jinn bude umožňovat horizontální škálování díky distribuovaným výpočtům na více těchto procesorech. Stále se však pohybujeme v budoucích otázkách a v rámci technologií, které ještě nevznikly.

Ve výsledku můžeme vidět, že **škálovatelnost a s tím spojená i rychlost transakce je v tuto chvíli ve velmi teoretické rovině**. K dosažení dobré škálovatelnosti je třeba překonat ještě několik poměrně zásadních problémů, ať už se jedná o odstranění koordinátoru, dokončení funkční techno-

logie swarm uzlů, nebo ještě lépe vytvoření ternárního procesoru. **Samotný teorém na kterém IOTA stojí však rozhodně podporuje vidinu miliard transakcí za sekundu**. K tomu ale musí být dostatečně podpořen celým ekosystémem.

3.2.1.4 Energetická nenáročnost

Ve spojitosti s mnoha moderními technologiemi, implementujícími distribuovanou účetní knihu, je možné slyšet, že jsou neúnosně energeticky náročné. Například roční elektrická spotřeba Bitcoinu se pohybuje kolem 70 TWH, což jej staví na podobnou úroveň, jako je roční spotřeba České republiky.[32] Kromě toho, že se jedná o neudržitelnou spotřebu vzhledem k ekologii, jde i o velmi finančně náročný provoz. Je tedy třeba, aby udržení samotné sítě v chodu nebylo silně finančně náročné a neekologické. Zároveň se v rámci sítě bude pohybovat spoustu zařízení, která nebudou neustále připojena ke zdroji energie a nebo se bude jednat o jednoduchá zařízení s nízkým výkonem. Použití sítě tedy nesmí nutit zařízení k tomu, aby obsahovala velice výkoná jádra, jelikož k výkonu samotné funkce by jej zařízení potřebovat nemělo a je zbytečné, aby ho obsahovalo pouze kvůli komunikaci. Stejně tak velký výkon bude velmi energeticky náročný a zařízení by potom nemohla pracovat delší dobu a musela by neustále řešit, jak a kde se dobijí.

IOTA je v rámci možností **energeticky velmi nenáročná**. Při založení transakce je třeba ověřit dvě další transakce, jak bylo popsáno v rámci rešerše. Hlavní energetická náročnost pro koncové zařízení tedy vzniká v rámci ověřování transakcí. Oproti většině ostatních systémů, kde ověření transakce probíhá pomocí miningu, zde nevzniká konkurenční boj, který žene obtížnost ověření a s tím spojenou energetickou náročnost na ověření nahoru. Energetická náročnost by zároveň měla klesat s větším růstem sítě, s kterým se lehce sníží náročnost funkce, kterou se ověřuje transakce, což by mělo vyústit ještě v 1,5 - 3 násobné snížení energetické náročnosti při vytváření transakce. V tuto chvíli je těžké určit přesnou energetickou náročnost na založení jedné transakce, především proto, že jde primárně o poměr mezi výkonností a energetickou náročností samotného procesoru, který je k výpočtům použit. Jedná se ale o malé náročnosti, vzhledem k tomu, na jak malých zařízeních se podařilo založit transakci.

Důkazem energetické nenáročnosti je například to, že **vznikla verze plného IOTA uzlu, který zvládne běžet na Raspberry Pi 3 a za běhu je schopen zpracovávat i 120 transakcí za sekundu**. Jedná se o neoficiální implementaci, která je napsaná v jazyce Go, ale již nějaký čas IOTA Foundation úzce komunikuje s týmem, který stojí za touto implementací. V rámci komunitních projektů vzniklo dokonce zařízení, kde je Raspberry Pi 3, na kterém běží plný uzel, napájeno čistě pomocí solární energie. **Podařilo se dokonce vytvořit transakci na mikrokontroleru STM32F1, který je napájen pouze napětím o výši mezi 2 a 3,6 V.**

Dalším energeticky zajímavým faktem je to, že IOTA používá ternární systém. Dle odhadů, které vznikly na základě výzkumu IOTA Foundation, **ušetří ternární architektura 34% drátů, které je nutno použít v rámci hardware**. To ve výsledku snižuje i energetickou náročnost, která vzniká ztrátami v rámci vedení elektřiny.

Co se energetické náročnosti týče, je v tomto ohledu IOTA poměrně efektivní a pro Internet věcí tedy vhodná. Velkou neznámou v tomto tématu je opět budoucí vývoj a to především potenciální vznik ternárního mikroprocesoru Jinn, který značně sníží energetickou náročnost, především v kombinaci s faktem, že IOTA je postavena na ternárním systému a tudíž pro tento procesor přímo vytvořena. I tak je ale v tuto chvíli protokol energeticky nenáročný, především ve srovnání s většinou dalších distribuovaných účetních knih.

3.2.1.5 M2M ekonomika

V tomto případě už nehovoříme o požadavku samotné technologie Internetu věcí, ale spíše o požadavcích celého ekosystému Internetu věcí. V rámci sítě by měla fungovat velká řada autonomních zařízení, které se budou samy rozhodovat na základě svých, nebo externích dat a budou využívat služeb zařízení jiných, nebo samy jiným zařízením své služby poskytovat. K tomuto bude muset vzniknout možnost převodu financí mezi těmito zařízeními, neboli systém machine to machine ekonomiky. Pro tento požadavek je IOTA, jako distribuovaná účetní kniha, velmi vhodná. Jako taková umožňuje vytvářet transakce a díky nim převádět finance mezi účty. Důležitým faktorem je ale také minimální výše transakce, kterou lze uskutečnit. Dá se předpokládat, že stroje budou chtít platit velmi malé částky za malé úkony, nebo například za velmi specifická data. Vzhledem k tomu, že nejmenší převeditelnou hodnotou v rámci IOTA transakce je 1 IOTA (token), lze vidět, že se jedná o velmi malou částku. V tuto chvíli je 1 IOTA přibližně 0,0005 Koruny české. Jedná se tedy o opravdu velmi malou částku. IOTA je tedy připravena i k potenciálnímu růstu tokenu do budoucna, který by neměl znemožnit formát mikrotransakcí.

3.2.2 Odstranění koordinátoru

Jak bylo již několikrát řečeno, odstranění koordinátoru je stěžejním bodem pro použitelnost protokolu. Z tohoto důvodu také dostal v rámci analytické části vlastní kapitolu. Během vzniku této práce vyšel na portálu Medium článek přímo od IOTA Foundation zabývající se přímo touto problematikou. Ten kromě představení samotného problému a fungování koordinátoru mluví také o výzkumu, který za účelem odstranění probíhá a **nastiňuje některá řešení, která by mohla odstranění umožnit**. Hlavním motivem je vytvoření lepšího bezpečnostního řešení, než pouhého proof-of-work.

První prezentovanou možností je model, kdy uzly samotné budou zodpovědné za hlídání transakcí, na základě čehož se vytvoří **systém, který**

bude hodnotit reputaci uzlu, založenou na tom, které uzly vytváří kolik neplatných transakcí. Čím více jich uzel vytváří, tím méně důvěryhodný bude a v souvislosti s tím bude klesat jeho váha v rámci sítě. Tento model je v tuto chvíli používán například u decentralizovaného sdílení souborů, kde jsou uzly hodnoceny na základě toho, jestli data, která poskytují, jsou nezávadná.

Dalším zkoumaným aspektem je **algoritmus pro výběr potvrzovaných transakcí**. Tento algoritmus byl probrán v rámci rešerše. Jako takový je funkční, avšak jeho úpravou a vylepšením by bylo možné dosáhnout vyšší úrovně zabezpečení, která by mohla dále podpořit odstranění koordinátoru.

Posledním prezentovaným konceptem je model, kde by **důvěryhodné entity**, jako například vládní instituce, nebo velké důvěryhodné korporáty, **provozovaly uzly, které by byly, stejně jako jejich majitelé, důvěryhodné. Tyto uzly by následně vytvářely referenční transakce podobným způsobem jako koordinátor**. Ty by pak měly větší váhu v rámci sítě, jelikož jako takové pochází z důvěryhodného uzlu. Osobně **toto řešení nevidím jako vhodné, především z důvodu nedostatečné decentralizace**. Cílem decentralizované účetní knihy je její decentralizace. Pokud se IOTA Foundation snaží odstranit koordinátor, jedním z jehož problémů je právě fakt, že se jedná o centralizovaný uzel, který má velkou moc a může tedy v rámci sítě určovat pravost dat, pak není řešením to, že tuto moc rozprostřeme mezi více institucí. Stále máme formu centralizace, pouze je více rozprostřená. Pro dosažení decentralizace je třeba, aby všechny uzly v rámci sítě vycházely ze stejných počátečních podmínek a jejich další upřednostňování vycházelo již pouze z veřejně známé nastavené metodiky, optimálně založené na jejich chování.

IOTA Foundation zároveň vydala zdrojové kódy k upravené verzi koordinátoru a vyzývá komunitu k tomu, aby daný kód testovala a hlásila případné chyby. Podle článku by do šesti měsíců měl tento koordinátor být nasazen do hlavní sítě místo aktuálního. Nejedná se však o odstranění koordinátoru, pouze náhrady starého za nový, lépe otestovaný. K nasazení by podle článku mělo dojít, dle optimistického odhadu, do šesti měsíců, tedy do května 2019.[33]

Ačkoliv je vidět, že IOTA Foundation na problematice odstranění koordinátoru pracuje jak na úrovni teoretické, tak na úrovni experimentální, **je stále zřejmé, že samotné odstranění je teoretickou otázkou budoucnosti s nejasným termínem a mírou úspěchu**. Stále probíhá výzkum a testy, ale samotné řešení stále není nalezené, natož implementované a otestované. Tento fakt je stále jedním z hlavním problémů, které jsou s IOTA protokolem spojeny a jsou velkou překážkou pro jeho širší využití v rámci Internetu věcí.

3.2.3 Výsledné zhodnocení použitelnosti

IOTA je zajímavý projekt, bezesporu postavený na silných teoretických a především matematických základech. Klade si velké cíle a jeho vize je velmi

globální a ambiciozní. Samotná IOTA Foundation, která za celým projektem stojí, v sobě sdružuje velmi kvalitní osobnosti z oblasti matematiky, vývoje a distribuovaných systémů. Celý projekt má za hlavní cíl stát se standardem pro Internet věcí, především v oblasti machine to machine ekonomiky a práce s daty v síti. Stát se tedy distribuovanou databází. Zároveň však chce umožňovat distribuované výpočty, identitu věcí a spoustu dalších technologií, které jsou pro Internet věcí velmi vhodné. Není sporu o tom, že kdyby se IOTA Foundation podařilo realizovat všechny myšlenky a teorie, které v rámci projektu vznikly, byl by projekt pro Internet věcí velmi vhodnou, ne-li nejlepší, volbou.

Problém který však pozoruji je to, kolik věcí před sebou ještě projekt má a kolik vývoje a změn musí nastat, aby byl projekt vůbec použitelný. Například koordinátor je velkou překážkou v rámci úspěchu technologie. Pokud nedojde k jeho odstranění, nemůže být vůbec pomyšleno na její využití v globálním měřítku. IOTA Foundation zároveň o odstranění mluví, ale není zatím dostupné oficiální vyjádření k tomu, kdy by k odstranění mohlo dojít, nebo jaké musí Tangle splnit podmínky, aby k odstranění vůbec dojít mohlo. Celkově je projekt protkán otázkami na které zatím chybí odpovědi. V roadmapě projektu úplně chybí časové odhady, zároveň stavy vývoje jednotlivých systémů jsou často velmi strohé a chybí jistota, že jsou založeny na stavu reálném. V minulosti zároveň už několikrát došlo k nedodržení a opakovanému odsouvání předem avizovaných harmonogramů v rámci projektu.

Jedním z dalších problémů, s kterými se již nějaký čas IOTA projekt potýká, je občasné neprofesionální vystupování hlavních členů týmu na sociálních sítích. Matematické jádro týmu se téměř na sociálních sítích nevyjadřuje. Oproti nim jsou potom v rámci projektu postavy jako Dominik Schiener nebo David Sønstebø, kteří jsou velmi aktivní na sociální síti Twitter. Tato aktivita je ovšem občas poměrně neprofesionálního charakteru a jejich vyjádření bývají občas velmi nevybíravá. Tyto problémy kulminovaly okolo doby, kdy se řešil problém bezpečnostní díry v hashovací funkci Curl, kterou IOTA Foundation vyvinula. V poslední době jsou tyto typy příspěvků již velmi ojedinělé, ne-li vyloženě historií, avšak v minulosti se toto vystupování často poměrně neblaze projevilo na důvěře veřejnosti i firem v celý projekt a tým, což se samozřejmě určitou mírou promítá i do samotné adopce této technologie.

Za IOTA protokolem a celým projektem stojí velmi kvalitní teoretický základ, schopní lidé a zároveň aktivní komunita, která sama do projektu investuje čas. Pokud se projektu podaří teorii podpořit opravdovým vývojem, pak věřím, že finální technologie bude pro Internet věcí velmi vhodná. Na cestě k tomuto je ovšem potřeba vyvinout spoustu systémů, provést mnoho testování a dále rozvíjet teoretický základ projektu. V neposlední řadě je úspěch celého projektu a vůbec fungování celé sítě založeno na nutnosti dostatečné adopce ekosystémem, jelikož například bezpečnost a rychlost vychází primárně z počtu uzlů a transakcí, které sítí proudí. To, jestli všechny tyto podmínky pro úspěch projektu budou splněny a z IOTA protokolu se stane standard,

ovšem ukáže pouze čas.

3.3 Aplikace IOTA protokolu a obchodní modely

Díky technologiím, které IOTA projekt přináší, vzniká možnost vytvořit mnoho nových obchodních modelů a vymyslet spoustu různých případů užití, které díky IOTA mohou vzniknout. Mezi hlavní vlastnosti, které jsou důležité pro zamýšlení se nad využitím protokolu, patří hlavně možnost mikrotransakcí. Tedy převodu velmi malých částek a to bez poplatků a téměř bez časové prodlevy. Další velmi důležitou funkcí je identita věcí, tedy to, že každé zařízení v síti by mělo mít svůj unikátní veřejný identifikátor a kromě něj by si mělo nést i určitá další metadata, z kterých půjde zjistit užitečné informace.

Není samozřejmě možné sepsat všechny modely a případy užití. Těch je prakticky neomezené množství. V rámci kapitoly se tedy zaměřím na velmi obecné modely, které jsou založeny na určitém konceptu a případně vždy nějaký nastíním více do hloubky.

3.3.1 Identita věcí

Identita věcí je jedna z velmi zajímavých a užitečných technologií v rámci IOTA a Internetu věcí. Díky tomu, že má každé zařízení své informace včetně identifikátoru, může autonomně fungovat. Díky tomu je možné například nechat vydělávat zařízení, která vlastníme, ale v danou chvíli nevyužíváme.

3.3.1.1 Sdílená vozidla

Ať už je vůz v našem vlastnictví, nebo je provozován určitou institucí, v případě že je nevyužíván, může být objednán a využit jako autonomní taxi. Zákazník vytvoří transakci, která poptává odvoz z bodu A do B, následně může dostat nabídky a jednu z nich si vybrat. Vůz, jehož nabídku zákazník vybral, přijede do bodu A, zákazník nastoupí a vůz zákazníka odveze do bodu B, kde budou tokeny převedeny z účtu zákazníka na účet vozidla. Případně mohou být tokeny převáděny postupně během jízdy, například za každých ujetých 100 metrů. Díky tomu může člověk nechat vydělávat své auto v době, kdy ho sám nevyužívá.

3.3.1.2 Automatické platby za využití služby

Řekněme, že osoba přijede někam s vozem a zaparkuje. Ve většině míst se již musí platit parkovné a tak musí dojít k platebnímu automatu, zadat číslo vozidla, zadat jak dlouho bude parkovat a zaplatit. Případně podobné údaje vyplnit na webu a zaplatit. V tomto moderním případě by jednoduše osoba zaparkovala, vystoupila a odešla. Automobil by sám platil za stání, například po jednotlivých minutách, kdy na místě opravdu stojí. V rámci kontroly by

pak pouze bylo ověřeno, že vůz s daným identifikátorem posílá tokeny po dobu co na místě stál a na základě toho není například pokutován. V rámci tohoto samozřejmě nemusíme mluvit pouze o vozidlech a parkování. Může jít i například o osobu, která jde třeba na veřejné koupaliště. Její zařízení, například mobil má identifikační číslo, na jehož základě se pozná, že je osoba v koupališti, případně při vstupu osoba mobil spáruje s určitým zařízením na koupališti. V čase, kdy na koupališti je, mobil posílá po minutách mikroplatby a při odchodu je zase oproti kontrole, že za čas bylo zaplaceno, osoba puštěna ven.

3.3.1.3 Kontrola v rámci dodavatelského řetězce

V této době v rámci dodavatelského řetězce většinou nemáme moc informací o tom, kde se zboží, které jsme si objednali nachází. V lepším případě máme určité číslo zásilky, pomocí kterého si můžeme v systému zásilkové služby zjistit, jestli je zboží ještě v jiné zemi, nebo bylo-li předáno například lokálnímu dopravci. S pomocí identity věcí a Tangle je ale možné dodavatelský řetězec sledovat s velkým množstvím specifických údajů. Každá zásilka by kromě jasného identifikátoru v rámci sítě obsahovala i senzory a bylo by tedy možné v okamžitém čase zjistit, co se se zásilkou děje, kde přesně se nachází a v jakém je například stavu. Tím by se dalo například kontrolovat, jaká je teplota zásilky v případě, že se jedná o zboží, které je tepelně náchylné. Díky tomu by firmy měly o mnoho lepší kontrolu nad svým zbožím, které jim chodí od dodavatelů a koncový zákazník kontrolu nad tím co si objednal, což by vedlo k mnohem menšímu počtu ztracených nebo rozbitých zásilek a v rámci toho ke snížení počtu nespokojených zákazníků.

3.3.1.4 Integrita informací o produktu

Jedním z velmi palčivých problémů při nákupu produktů z druhé ruky je často nejistota ohledně stavu produktu, záruce a dalších aspektech, které jsou při nákupu produktu důležité. Toto je častý problém například při nákupu ojetého vozidla. Především se zde jedná o problém se stáčením najetých kilometrů. Ačkoliv se tyto problémy snaží řešit legislativa například tím, že se při pravidelných technickým kontrolách vozidla nyní musí fotit a zaznamenávat stav tachometru a ke stáčení by tedy nemělo docházet, nikdo nezajistí to, jestli nedošlo k podvodu přímo při technické kontrole. Systém, který data uchovává je opět centralizovaný a data jsou tedy držena v rámci jedné instituce, čímž jsou opět náchylná k manipulaci s nimi.

Toto by mohlo být řešeno tím, že každý automobil bude informace o sobě ukládat na Tangle a tím pádem budou důvěryhodná a dohledatelná. Konsenzus obecně nepřijme informaci o tom, že vozidlo má nově menší počet najetých kilometrů, než mělo při posledním nahrávání dat, čímž bude zajištěno, že nebude moci docházet ke stačení kilometrů. Možná se někomu podaří hodnoty

upravit na samotném vozidle, ale rozhodně ne v rámci decentralizovaného registru, kde bude možné data ověřit.

Tímto způsobem by samozřejmě nemuselo být řešeno pouze stáčení najetých kilometrů u vozidel, ale i například záruční lhůty u běžných produktů. Záruční lhůta bude, stejně jako počet najetých kilometrů, uložena v Tangle, tím pádem bude neměnná až na výjimky, kterou může být například oprava produktu, která v rámci legislativy záruční lhůtu prodlužuje. Věc samotná potom bude vědět, že došlo k opravě a posune záruční lhůtu. Stejně tak bude zaznamenávat opravy a svůj aktuální stav. Pokud tedy dojde k poruše určité komponenty, věc tuto poruchu zaznamená a uloží. Tím pádem, pokud bude kupující chtít, bude si moci ověřit, jaký je stav produktu, ať už jde o záruční lhůtu, nebo například o to v jakém je stavu. Díky tomu se nestane, že by mu prodejce mohl o stavu lhát a kupujícímu by tím vznikla finanční újma.

3.3.2 Mikrotransakce

Možnost posílat malé částky bez poplatku je převratná funkce, která se ve výsledku bude líbit spíše zákazníkům než poskytovatelům služeb. V určitých případech ale budou benefitovat obě strany.

3.3.2.1 Plaťte za to, co opravdu využíváte

Moderním způsobem využívání placených služeb je forma předplatného. Objednáte si službu a platíte měsíční paušál, nehledě na to, jak moc službu využíváte. Některým zákazníkům, kteří službu využívají velmi často, se takovýto formát vyplatí, ale většinou je faktem to, že zákazník za peníze, které paušálně platí, služby dostatečně nevyužije. Mikrotransakce by umožňovaly službu platit pouze v případě, že je využívána. Například platit každou sekundu nebo minutu, co máme službu puštěnou. Platit za každý přehraný film nebo přečtený článek.

Jednou z možných ukázek tohoto modelu je například platba za energie. V současné době jsou energie placeny formou záloh. Zákazník každý měsíc zaplatí určitý paušál a na konci roku se vyúčtuje, kolik za rok reálně vyčerpal a buď musí doplatit určitou částku, nebo je mu část vrácena. Jedná se o nepříjemný proces už z důvodu, že zákazník musí platit každý měsíc, zároveň nemá často přehled o tom, kolik dané energie spotřebuje, takže neví, jestli by neměl šetřit, nebo si naopak stojí dobře. Toto zjistí až na konci účetního roku, kdy většinou musí ještě doplácet. V rámci tohoto modelu by zákazník provedl mikrotransakci například za každý půllitr vody, který vyčerpal. Tokeny by automaticky odcházely z účtu a zákazník by se nemusel o nic starat. Zároveň by díky historii účtu viděl, kolik přesně za vodu zaplatil v rámci předchozích měsíců a díky tomu měl využití pod kontrolou.

Další možností je například platit pouze za přenesené byty v rámci komunikace. Například při sledování streamovaných filmů. Umožnit nákup pouze

jednotek řádků z určité databáze, na rozdíl od aktuálních modelů, kdy pokud chce zákazník zakoupit databázi, většinou musí koupit celý set o velkém objemu, ačkoliv ve skutečnosti například potřebuje pouze pár řádků.

3.3.3 Hodnota dat

S nárůstem výpočetních kapacit se stále více rozrůstá obor big data. Samotná data jsou proto čím dál hodnotnější. Díky tomu může člověk v budoucnosti šetřit část nákladů za využívání určitých zařízení tím, že zařízení budou odesílat anonymizovaná data, která budou určitým způsobem hodnotná pro zařízení jiná, případně pro společnosti, které se budou zabývat jejich výzkumem.

3.3.3.1 Data o vozidlech

Jedním z moderních problémů je mobilita. Vozidel přibývá a doprava je čím dál horší. Tvoří se kolony a například ve velkých městech se často automobilová doprava stává prakticky nepoužitelnou. Díky autonomním vozidlům, případně fyzicky řízeným vozidlům se senzory zaznamenávajícími rychlost, polohu, stav vozovky a další informace, by se tento problém mohl vyřešit, nebo by se situace mohla alespoň částečně zlepšit. Každý vůz by odesílal každých několik sekund informace ze senzorů. Za tyto data by dostával určitý obnos tokenů, který by majiteli auta částečně hradil náklady na jeho provoz. Data by poté využívala například další vozidla, která by na jejich základě vypočítávala nejvhodnější trasu. Nebo by tyto data mohla například ve velkém zpracovávat jedna instituce, který by následně na jejich základě řídila autonomní vozidla a zároveň poskytovala informace do navigačních systémů vozidlům s fyzickým řidičem, to vše opět za úplatu.

3.3.3.2 Zdravotní data

Stejně jako vozidla odesílaly údaje ze svých senzorů, mohly by například chytré hodinky odesílat údaje o zdravotním a jiném stavu osoby, která je nosí. Tyto data by opět měla mnoho využití. Ať už se jedná o informace o tom, kam lidé kdy chodí, jak cvičí, jaký mají zdravotní stav, nebo jak dlouho spí. Všechna tyto data se dají zpracovávat například pro marketingové účely. Data by samozřejmě byla poskytována anonymizovaně. Za tyto data by opět vlastníci hodinek byli odměňování tokeny.

3.3.4 Decentralizované aplikace

Chytré kontrakty umožňují tvorbu takzvaných decentralizovaných aplikací. Zde se nejedná o vyloženě nový model. V této době již spoustu decentralizovaných aplikací běží, například na technologii Ethereum[34]. Decentralizované aplikace jsou specifické v tom, že nejsou v podstatě nikým vlastněny,

běží veřejně, takže každý může prozkoumat jejich zdrojový kód a ověřit, že jsou opravdu naprogramovány tak, jak říkají, že jsou. Zároveň není možné je ovlivnit zvenčí. Jednoduše reagují na vstup a na jeho základě provádějí to, co mají. Díky tomu jsou vhodné například na uzavírání smluv, nebo na finanční aplikace, jako jsou burzy a nebo například sázkové kanceláře.

Například v případě sázkových kanceláří teoreticky může hrozit, že nedojde k vyplacení výhry, jelikož majitel se tak rozhodne, nebo například zkrachuje. Pomocí decentralizované aplikace by byly vytvořeny kontrakty, které by přijímaly sázku na určitý zápas, data braly z nějakého ověřeného zdroje a na jeho základě pak případně vyplácely výhru sázkařům. Zároveň by například kontrakt mohl být nastavený tak, že na účtu z kterého má být vyplácena výhra musí být alespoň 50% objemu jako je výše sázek na daný kontrakt. Díky tomu je jistota, že kontrakt (je-li dobře naprogramovaný, což lze ověřit prozkoumáním veřejného zdrojového kódu) výhru doopravdy vyplatí a zároveň i v případě, že by provozovatel zkrachoval, tak z účtu, kde byly uzamčené peníze budou hráčům vyplaceny výhry.

3.3.5 Superpočítač

Jednou ze zajímavých využití IOTA projektu je přidružený projekt Qubic, který byl rozebírán v rámci rešerše a následně ještě analýzy. Díky principu orákul bude možné využít aktuálně nevyužívaná zařízení v rámci Tangle, jako výpočetní kapacitu za úplatu. Bude se v podstatě jednat o obří cloud, na jehož výpočetní kapacitu se budou dobrovolně skládat zařízení v síti. Jde o koncept cloud computingu, pouze ve velmi globálním měřítku, čímž je možné dosáhnout o mnoho lepších výpočetních kapacit. Toto dává další možnosti jednotlivým zařízením, které například mohou koupit větší objem dat ze senzorů a potřebovat nad nimi udělat nějaké analýzy, které jsou důležité pro jejich další rozhodování. Jelikož tyto zařízení mají poměrně omezenou výpočetní kapacitu, nechají si za úplatu problém spočítat dalšími zařízeními, které zrovna mají volnou výpočetní kapacitu.

3.4 Shrnutí kapitoly

V rámci analytické kapitoly byl probrán IOTA protokol z hlediska jeho vhodnosti pro využití v rámci Internetu věcí. Výsledek nebyl vyloženě uspokojivý. IOTA je technologie, která má velký potenciál, především díky kvalitnímu teoretickému základu na kterém je založena, vizi a schopným lidem v IOTA Foundation. Stále má ale před sebou velmi dlouhý vývoj, jak po teoretické stránce, tak po implementační. Zároveň k tomu, aby byla použitelná, jako hlavní technologie pro Internet věcí, je nejprve třeba, aby dosáhla určité adopce, jelikož na ní je založena většina jejích vlastností. Roadmapa projektu bohužel nespecifikuje pro jednotlivé vyvíjené technologie žádné časové odhady a tak je těžké určit, s jakou pravděpodobností projekt uspěje.

V druhé části analýzy byly následně prezentovány zajímavé vlastnosti IOTA projektu, díky kterým mohou vzniknout nové obchodní modely a systémy. Například mikrotransakce umožňují obchodní model, kde zákazník platí opravdu pouze za to, co v rámci služby čerpal. Například platí za každý vyčerpaný půllitr vody hned ve chvíli co ho vyčerpal. Těchto nových modelů je spousta a nelze zde všechny probrat, proto jich bylo nastíněno pár a jeden z nich bude implementován ve třetí, implementační části.

Kapitola 4

Implementace

4.1 Úvod k implementaci

V rámci implementační části jsem se rozhodl implementovat pay-per-view (PPV) článkový web. Pay-per-view je koncept, kdy uživatel platí za zhlédnutí či zobrazení. V tomto případě by tedy měl vzniknout článkový web, respektive jeho jádro, kde uživateli bude zpřístupněn článek na základě toho, že za něj zaplatí. Vzhledem k tomu, že platby by měly být často nízkého objemu, je zde vhodné využít k těmto transakcím právě technologii IOTA, jelikož pro platby v řádu korun jsou většinou běžné elektronické platební metody nevhodné. Z tohoto důvodu bude v rámci kapitoly probrán i ekonomický aspekt a navržena možná další řešení, které by mohla fungovat i se standardními platebními metodami.

Implementace je především něčím jako Proof of Concept implementace s využitím technologie IOTA. Jde tedy především o to, zjistit, jestli takovýto koncept může fungovat a jak použitelný v tuto chvíli je. Zároveň je v rámci implementace třeba zanalyzovat samotné možnosti vývoje nad IOTA protokolem a systém tohoto typu je k tomu vhodnou volbou. V poslední řadě je zároveň třeba ověřit funkčnost řešení a doporučit další postup v rámci projektu.

4.2 Analytická část

4.2.1 Aktuální stav

Hlavní myšlenka systému vyvíjeného v rámci implementační části práce není nijak převratná ani nová. Pay-per-view model je používán poměrně často a to především pro sportovní přenosy. Často se například tímto způsobem prodává možnost sledovat galavečery boxu a podobných sportů. V moderní době se tento model začal ovšem dostávat právě i do prostředí článkových webů. V tomto prostředí jsou průkopníky, kteří začali využívat tento model, především tištěné deníky, které část článků replikují na webu. Je to zároveň

způsob, jakým být schopen na zprávy reagovat rychle, což přímo tištěné noviny, vydávané jednou denně, nemohou.

Obchodní model, který většina těchto webů dříve využívala byl založen čistě na pronájmu reklamních ploch. Reklama na webových stránkách začíná být ovšem čím dál tím agresivnější, například tím, že zabírá větší plochu, nebo se často skrývá tak, že se snaží vizuálně působit jako nereklamní obsah webu. Čtenářům toto začíná čím dál tím více vadit a začali se uchylovat k řešením, která jim od reklamy uleví, jako jsou rozšíření do prohlížečů, která jsou do velké míry schopná reklamu blokovat. Nejen z tohoto důvodu si provozovatelé těchto webů začali uvědomovat, že lepší cestou k výnosu bude spíše placený obsah. Ten umožňuje snížit reklamu na stránkách, což zlepší uživatelský zážitek, avšak zároveň nepřijít o zisky díky jinému finančnímu zdroji, kterým je právě placený obsah. Tento obchodní model je v zahraničí již hojně využíván, do České republiky však pomalu přichází až v posledních letech. Placený obsah nyní využívají například Hospodářské Noviny ve své elektronické podobě na portálu ihned.cz. Na portálu je možné zakoupit předplatné digitálního obsahu, které kromě toho, že uživateli zpřístupní placený obsah, zároveň z webu odstraní i reklamu. Předplatné se uzavírá na dobu neurčitou a stojí 349 Korun českých na měsíc.[35] Kromě předplatného je ale možné zakoupit i jednotlivé placené články samostatně. Články stojí standardně 9 Kč a je možné tuto částku zaplatit i kartou. Další kdo využívá placeného obsahu je například deník E15, který za 199 Korun českých měsíčně umožňuje přístup do E15 Premium, které má obsahovat speciální články, investiční poradnu a další výhody. Zároveň má členství částečně omezit reklamu na webu.[36]

Projektů, které jsou založeny na placeném obsahu zatím není příliš mnoho. Postupně na tento model ale přechází čím dál tím více obsahových webů, které si začínají uvědomovat, že se může jednat o krok správným směrem. Nedá se ovšem říci, že všechny obsahové weby jsou vhodným adeptem. Například ale u článkových webů jsou lidé stále zvyklí kupovat tiskoviny a na základě toho jsou poměrně naučení za textový obsah platit a poměrně rádi si tedy za obsah, který je zajímá zaplatí.

Ačkoliv je model placeného obsahu na vzestupu, nedá se to říci i o samotném Pay-per-view modelu. Samozřejmě, že ten také částečně stoupá, jelikož jde trochu ruku v ruce s placeným obsahem, ale spousta provozovatelů webů zavádí pouze předplatné, jako například uváděná E15 a samotnou možnost zaplatit jednorázovou částku, například pouze za jeden specifický článek, neumožňují. Toto může být překážkou pro část potenciálních zákazníků, kteří se na web dostali kvůli jednomu specifickému článku, který by si rádi koupili, avšak zbylé články je nezajímají a nechtějí si tedy kupovat celé předplatné. Z tohoto důvodu často odejdou bez jakéhokoliv nákupu a provozovatel tím ztrácí potenciální zisk. Důvodem proč se články často nedají koupit, může být problém s výší transakce, která by měla být uskutečněna. Hospodářské Noviny za článek požadují 9 Kč. To není rozhodně málo a při srovnání s průměrnou cenou článku, která vychází po vydělení ceny jednoho

výtisku počtem článku v něm obsažených, je jasné, že se jedná o poměrně vysokou částku. I tak ale na prodeji článku Hospodářské Noviny nevydělají tolik, jelikož při takto malé částce zaplatí velké procento na poplatcích za karetní platbu. Například známý poskytovatel platební brány, společnost GoPay s.r.o. požaduje za jednu transakci přes platební bránu 3 Kč a až 2,2% z částky.[37] V takovém případě je tedy prakticky nemožné prodávat článek například za 3 Koruny, jelikož by prodejce z jeho ceny nedostal prakticky nic. Z tohoto důvodu je možnost efektivní elektronické platební metody, která je nejen rychlá, ale zároveň levná, praktickou nezbytností. IOTA protokol je v tomto případě vhodným kandidátem, jelikož transakce samotná je bez poplatku a rychlostí by v budoucna měla karetní převod spíše překonat.

4.2.2 Obecné požadavky na implementovaný systém

Hlavním požadavkem, který je třeba zajistit, je možnost plateb pomocí IOTA protokolu, tedy možnost zaplatit IOTA tokeny, na základě čehož bude uživateli umožněn přístup ke specifickému článku, který si vyžádal. Uživatel by měl mít zároveň svůj profil, ve kterém bude mít jednotlivé články, které si zakoupil, jelikož nemůže být nucen za každé zobrazení platit znovu. Za tímto účelem je nutná možnost registrace a přihlášení. Jednotlivé specifické funkcionality budou probrány v rámci vlastní kapitoly a následně v rámci kapitoly obsahující případy užití.

4.2.3 Možnosti vývoje nad IOTA protokolem

Úvodem této části je důležité poznamenat, že tato práce vznikala v době, kdy nebyl vydán jazyk Abra. Pokud tedy v době čtení již překladač pro tento jazyk byl vydán v úrovni, která umožňuje využití Abra k vývoji, berte, prosím, v potaz, že v době, kdy tato práce vznikala překladač nebyl, stejně tak ani dokumentace k jazyku. Občas v rámci příspěvků na IOTA blogu vyšel článek pojednávající o postupu ve vývoji tohoto jazyka a občas ukázka kódu, avšak dokumentace samotná, ani jazyk dostupné nebyly. Z tohoto důvodu nemohou být probrány více do hloubky a zmíněny budou pouze okrajově.

Abra jako taková umožňuje tvorbu decentralizovaných aplikací. V případě existence by tedy bylo možné vytvořit sadu funkcí, které by byly zaneseny do tangle a exekuovány na různých zařízeních v rámci sítě. Jako taková by aplikace byla většinou sadou listenerů reagující na události v rámci sítě. Jelikož ale Abra není v použitelném stavu, budeme se muset v rámci práce omezit na aplikaci centralizovanou.

Důležitým aspektem při vývoji systému, který by měl komunikovat s Tangle je nutnost provozovat vlastní plný uzel. Uzel je totiž jediná možnost, jak s Tangle komunikovat. Je sice možné použít uzel cizí, avšak tato možnost se většinou nenaskytne, jelikož pokud by se už člověk dostal k ip adrese a portu na kterém uzel běží a komunikuje, bude mít jeho majitel s největší

pravděpodobností uzel nastaven tak, aby nepřijímal komunikaci z vně serveru na kterém se nachází, případně pouze v omezeném módu. Z tohoto důvodu je třeba provozovat vlastní uzel, přes který bude systém komunikovat s Tangle a samotný systém optimálně provozovat na stejném serveru. Případně dostatečně zabezpečit v rámci uzlu to, aby s ním mohl komunikovat jenom námi provozovaný systém. S uzlem je poté možné komunikovat dvěma způsoby. Jedním jsou HTTP volání. Uzel má své vystavené API, s jehož pomocí je možné s Tangle komunikovat. Toto volání je vhodné například na vytváření transakcí, nebo jednorázové dotazování. Druhou možností je využití IOTA implementace Zero Message Queue, kterou může mít uzel vystavenou a další systémy se na ni mohou napojit. Jedná se o implementaci decentralizované fronty zpráv (message queue)[38]. Ta obsahuje informace o událostech, které v uzlu nebo na sítí nastaly a případně i jejich samotná data. Klient se může přihlásit k odběru určitého typu zpráv a díky tomu například efektivně získávat informace o nově vzniklých transakcích, bez nutnosti neustálého opakovaného volání HTTP požadavků na API.

Když existuje uzel, pomocí něhož může systém komunikovat s Tangle, je třeba vytvořit samotného klienta, který bude s uzlem komunikovat. Tento klient může být buď částí samotného systému, nebo samotnou vlastní službou s vlastním rozhraním. IOTA Foundation poskytuje několik knihoven, které implementují právě klienta a díky tomu značně zjednodušují komunikaci s uzlem. Jedním ze zjednodušení je například fakt, že knihovna se sama stará o zakódování hodnot do trytů a naopak zpětné dekódování při přijímání dat. V opačném případě by toto musel vývojář implementovat sám, což by samozřejmě značně prodlužovalo čas vývoje. Aktuální hlavní oficiální implementace klienta jsou v jazycích Javascript, Java, C# a Go. Toto jsou knihovny uvedené v dokumentaci. Na GitHubu IOTA Foundation je však možné nalézt v tuto chvíli i implementaci v jazyce Rust, nebo Python. U většiny těchto knihoven je ovšem na GitHubu uvedeno, že se jedná o beta verzi. V případě Rustu a Pythonu potom přímo doporučeno knihovny nevyužívat. Jediná knihovna, které nenese informaci o beta verzi, ani žádné další varování o jejím využití na vlastní riziko, je implementace v Javascriptu a jako taková tedy působí jako nejlepší možná volba.

Další využitelnou knihovnou pro vývoj je knihovna, která implementuje Masked Authenticated Messaging o kterém bylo pojednáváno dříve v rámci práce. Tato knihovna je pro tuto implementaci nepotřebnou, avšak díky možnosti přenášet zašifrované zprávy je určitě v mnoha systémech využitelnou.

4.2.4 Funkcionality

V rámci této sekce budou pouze nastíněny jednotlivé základní funkcionality, které jsou nezbytnou součástí aplikace a bez nichž by byla pro uživatele prakticky nepoužitelná. Samotné případy užití jsou probrány až v sekci, týkající se přímo této problematiky.

4.2.4.1 Registrace a přihlášení

Aby uživatel mohl články číst, je třeba, aby měl možnost vytvořit si účet a do něj se přihlašovat. Vzhledem k tomu, že uživatel nezískává přístup k ničemu v rámci bezplatného využívání aplikace, není třeba aby v tuto chvíli při registraci vyplňoval více údajů, než je e-mail a heslo. Hlavním účelem registrace je pouze vytvoření účtu, kterému budou následně přiřazovány články na základě plateb. Je pravděpodobné, že v budoucnu budou vyžadovány další údaje, především ve spojitosti s právními aspekty plateb.

4.2.4.2 Výpis článků

Pro to, aby si uživatel mohl vůbec vybrat, který článek si chce zakoupit, je třeba umožnit mu výpis článků s jejich názvy a perexy. Pro jednodušší výběr je vhodné, aby systém umožňoval filtrování článků přes jejich kategorie, názvy, nebo další hodnoty.

4.2.4.3 Zažádání si o článek

Pro možnost čtení článku je třeba umožnit uživateli jeho výběr a určitým způsobem vyžádání platebních údajů. Systém by na základě tohoto měl zaregistrovat požadavek a předat uživateli informace o platbě, po jejímž provedení bude uživateli článek zpřístupněn.

4.2.4.4 Zobrazení historie plateb

V rámci informací o svém účtu by měl mít možnost uživatel vidět, jaké platby kdy prováděl a jaké články na základě jednotlivých plateb dostal.

4.2.4.5 Zobrazení zakoupených článků

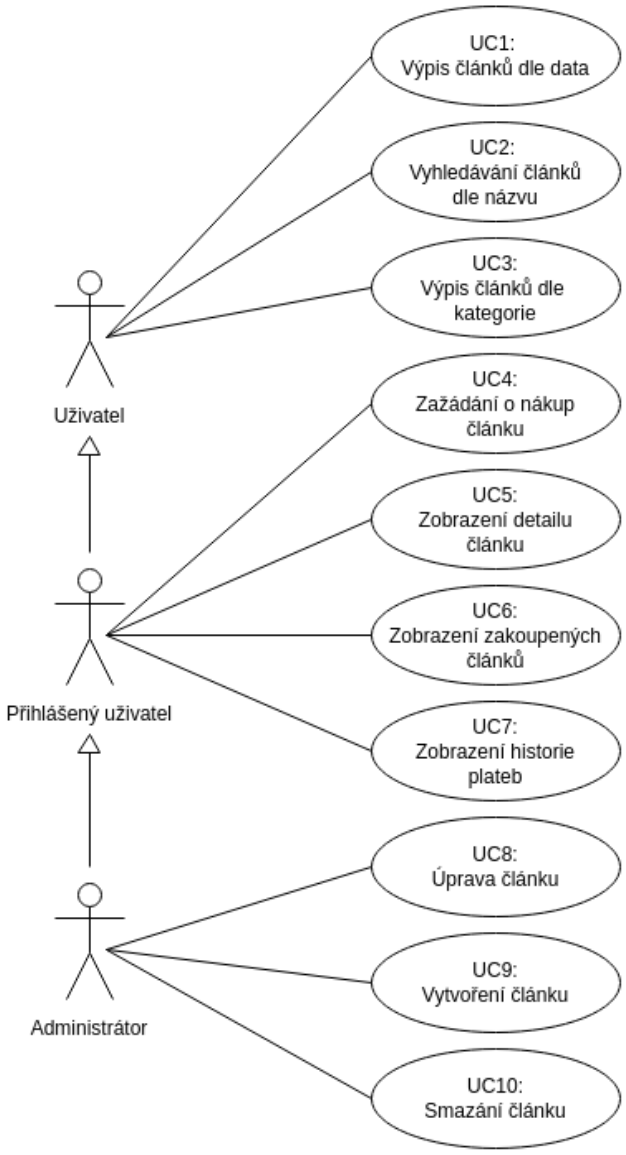
Stejně jako historii plateb by měl mít uživatel možnost procházet své již zakoupené články, aby se v případě potřeby mohl jednoduše vrátit k nějakému, který si už přečetl, nebo například již zakoupil a přečíst ještě nestihl.

4.2.5 Případy užití

Případy užití jsou v rámci softwarového návrhu popisy posloupností akcí ze kterých sestává určitá funkcionalita. Případ užití většinou popisuje interakci mezi uživatelem a systémem. Uživatelem nemusí být vždy fyzická osoba, ale může jím být i organizace, nebo například i jiný software.

Popis případu užití nemá oficiálně definovaný formát. Jedním z často užívaných způsobů je takzvaný diagram případů užití, který spadá do jazyka UML. Tento diagram je vhodný k jednoduchému soupisu všech případů užití a naznačení aktérů, kterých se daný případ užití týká, případně jejich provázání mezi sebou. Vhodnější formou pro popis samotného případu užití

je formátovaný text, který jednotlivé případy užití slovně popisuje. V rámci této kapitoly je vypracován jak diagram případů užití, tak textové popisy jednotlivých případů. Jelikož, jak jsem již řekl, není forma popisu pevně dána, vycházím z doporučené struktury, jak byla popsána v knize Writing Effective Use Cases[39], jejímž autorem je Alistair Cockburn. Tuto knihu na svém blogu doporučuje Martin Fowler, který je obecně uznáván za jednoho z předních specialistů na systémové návrhy a vývoj.[40]



Obrázek 4.1: Diagram případů užití

4.2.5.1 UC1: Výpis článků dle data

Hlavní účastník: Uživatel
Rozsah: Komponenta
Úroveň: Uživatelský cíl
Zúčastněné strany a zájmy:

- Uživatel - Získat seznam článků seřazených dle data
- Navrhovaný systém - Dodat uživateli seznam článků seřazených dle data
- Databáze

Předpoklady:

- Funkční připojení k databázi

Minimální plnění: Uživatel získá nazpět informaci o výsledku.
Úspěšné plnění: Uživatel získá nazpět požadovaná data.
Spouštěč: Vstup na stránku s výpisem článků.
Scénář úspěšného plnění:

1. Uživatel vstoupí na stránku pro výpis článků.
2. Frontendová aplikace provede požadavek na API na získání seznamu článků.
3. Systém z databáze získá seznam článků od nejnovějších a uživateli předá jeho první stránku.

Výjimky:

- **Systémová chyba:** Uživateli je prezentována informace o chybě v systému.

4.2.5.2 UC2: Vyhledávání článku dle názvu

Hlavní účastník: Uživatel
Rozsah: Komponenta
Úroveň: Uživatelský cíl
Zúčastněné strany a zájmy:

- Uživatel - Získat seznam článků odpovídajících určitému názvu
- Navrhovaný systém - Dodat uživateli seznam článků odpovídajících určitému názvu
- Databáze

Předpoklady:

- Funkční připojení k databázi

Minimální plnění: Uživatel získá nazpět informaci o výsledku.
Úspěšné plnění: Uživatel získá nazpět požadovaná data.
Spouštěč: Kliknutí na tlačítko hledat na stránce s výpisem článků.
Scénář úspěšného plnění:

1. Uživatel vstoupí na stránku pro výpis článků.
2. Uživatel vyplní pole pro vyhledávání.
3. Uživatel stiskne tlačítko hledat.
4. Frontendová aplikace odešle požadavek na API na vyhledání článků odpovídajících vyhledávanému názvu.
5. Systém z databáze získá odpovídající články a předá uživateli výpis první stránky.

Alternativní scénář:

- **Vyhledávání neodpovídají žádné články:** Uživateli je vrácen prázdný seznam s informací, že nebyly nalezeny žádné články.

Výjimky:

- **Systémová chyba:** Uživateli je prezentována informace o chybě v systému.

4.2.5.3 UC3: Výpis článků dle kategorie

Hlavní účastník: Uživatel
Rozsah: Komponenta
Úroveň: Uživatelský cíl
Zúčastněné strany a zájmy:

- Uživatel - Získat seznam článků odpovídajících určité kategorii
- Navrhovaný systém - Dodat uživateli seznam článků z určité kategorie
- Databáze

Předpoklady:

- Funkční připojení k databázi

Minimální plnění: Uživatel získá nazpět informaci o výsledku.
Úspěšné plnění: Uživatel získá nazpět požadovaná data.
Spouštěč: Kliknutí na specifickou kategorii v seznamu kategorií na stránce s výpisem článků.
Scénář úspěšného plnění:

1. Uživatel vstoupí na stránku pro výpis článků.
2. Uživatel klikne na jednu z kategorií.
3. Frontendová aplikace odešle požadavek na API na vyhledání článků v dané kategorii.
4. Systém z databáze získá odpovídající články a předá uživateli výpis první stránky.

Výjimky:

- **Systémová chyba:** Uživateli je prezentována informace o chybě v systému.

4.2.5.4 UC4: Zažádání o nákup článku

Hlavní účastník: Uživatel
Rozsah: Komponenta
Úroveň: Uživatelský cíl
Zúčastněné strany a zájmy:

- Uživatel - Získat údaje k platbě pro nákup požadovaného článku
- Navrhovaný systém - Vytvořit platbu čekající na zaplacení a předat údaje o ní uživateli
- Databáze

Předpoklady:

- Funkční připojení k databázi
- Uživatel je přihlášen

Minimální plnění: Uživatel získá nazpět informaci o výsledku.
Úspěšné plnění: Uživatel získá nazpět podklady k provedení platby a platba je uložena v systému.
Spouštěč: Kliknutí na tlačítko Koupit článek na detailu článku
Scénář úspěšného plnění:

1. Uživatel vstoupí na detail článku, který nemá zakoupený.
2. Uživatel klikne na tlačítko Koupit článek.
3. Frontendová aplikace odešle požadavek na API na vytvoření požadavku na transakci.
4. Systém zkontroluje, zda je uživatel přihlášen a zda jsou přijatá data validní.

5. Systém vypočítá cenu článku v závislosti na aktuální hodnotě IOTA tokenu.
6. Systém uloží údaje k platbě, jako platbu čekající na provedení, do databáze.
7. Uživateli jsou systémem nazpět tyto podklady k platbě předány a zobrazeny.

Výjimky:

- **Kupovaný článek neexistuje:** Uživateli je nazpět vrácena informace o tom, že požadovaný článek neexistuje.
- **Uživatel není přihlášen:** Uživatel je s informací, že není přihlášen, přesměrován na přihlašovací stránku.
- **Systémová chyba:** Uživateli je prezentována informace o chybě v systému.
- **Neaktuální hodnota IOTA tokenu v databázi:** Uživateli je předána informace, že není možné v tuto chvíli získat cenu článku.

4.2.5.5 UC5: Zobrazení detailu článku

Hlavní účastník: Uživatel
Rozsah: Komponenta
Úroveň: Uživatelský cíl
Zúčastněné strany a zájmy:

- Uživatel - Získat detail požadovaného článku
- Navrhovaný systém - Dodat uživateli detail požadovaného článku
- Databáze

Předpoklady:

- Funkční připojení k databázi
- Uživatel je přihlášen
- Uživatel má požadovaný článek zakoupený

Minimální plnění: Uživatel získá nazpět informaci o výsledku.
Úspěšné plnění: Uživatel získá nazpět informaci o výsledku a požadovaný článek.
Spouštěč: Přechod na stránku s detailem článku
Scénář úspěšného plnění:

1. Uživatel vstoupí na stránku s detailem článku.
2. Frontendová aplikace odešle požadavek na API na získání daného článku.
3. Systém zkontroluje, zda je uživatel přihlášen a zda článek existuje.
4. Systém zkontroluje, zda má uživatel daný článek zakoupený.
5. Systém z databáze získá článek a předá jej frontendové aplikaci, která jej uživateli zobrazí.

Alternativní scénář:

- **Uživatel nemá daný článek zakoupený:** Uživateli je s informací, že nemá článek zakoupený prezentováno tlačítko Zakoupit článek.

Výjimky:

- **Požadovaný článek neexistuje:** Uživateli je nazpět vrácena informace o tom, že požadovaný článek neexistuje.
- **Uživatel není přihlášen:** Uživatel je s informací, že není přihlášen, přesměrován na přihlašovací stránku.
- **Systémová chyba:** Uživateli je prezentována informace o chybě v systému.

4.2.5.6 UC6: Zobrazení zakoupených článků

Hlavní účastník: Uživatel
Rozsah: Komponenta
Úroveň: Uživatelský cíl
Zúčastněné strany a zájmy:

- Uživatel - Získat seznam článků, které si uživatel zakoupil
- Navrhovaný systém - Dodat uživateli seznam článků, které má zakoupené
- Databáze

Předpoklady:

- Funkční připojení k databázi
- Uživatel je přihlášen

Minimální plnění: Uživatel získá nazpět informaci o výsledku.
Úspěšné plnění: Uživatel získá nazpět informaci o výsledku a seznam článků, které má zakoupené.
Spouštěč: Přechod na stránku se seznamem zakoupených článků.
Scénář úspěšného plnění:

1. Uživatel vstoupí na stránku pro výpis zakoupených článků.
2. Frontendová aplikace odešle požadavek na API pro získání zakoupených článků uživatele.
3. Systém zkontroluje, zda je uživatel přihlášen.
4. Systém z databáze získá články, které má uživatel zakoupené a předá je frontendové aplikaci, která je uživateli zobrazí.

Výjimky:

- **Uživatel není přihlášen:** Uživatel je s informací, že není přihlášen, přesměrován na přihlašovací stránku.
- **Systémová chyba:** Uživateli je prezentována informace o chybě v systému.

4.2.5.7 UC7: Zobrazení historie plateb

Hlavní účastník: Uživatel
Rozsah: Komponenta
Úroveň: Uživatelský cíl
Zúčastněné strany a zájmy:

- Uživatel - Získat seznam plateb, které jsou spojeny s daným uživatelským účtem
- Navrhovaný systém - Dodat uživateli platby, které jsou s jeho účtem spojeny
- Databáze

Předpoklady:

- Funkční připojení k databázi
- Uživatel je přihlášen

Minimální plnění: Uživatel získá nazpět informaci o výsledku.
Úspěšné plnění: Uživatel získá nazpět informaci o výsledku a historii jeho plateb.
Spouštěč: Přechod na stránku s historií plateb
Scénář úspěšného plnění:

1. Uživatel vstoupí na stránku pro výpis historie jeho plateb.
2. Frontendová aplikace odešle požadavek na API pro získání historie plateb uživatele.
3. Systém zkontroluje, zda je uživatel přihlášen.
4. Systém z databáze získá platby, které jsou spojené s uživatelovým účtem a předá je frontendové aplikaci, která je uživateli zobrazí.

Výjimky:

- **Uživatel není přihlášen:** Uživatel je s informací, že není přihlášen, přesměrován na přihlašovací stránku.
- **Systémová chyba:** Uživateli je prezentována informace o chybě v systému.

4.2.5.8 UC8: Úprava článku

Hlavní účastník: Uživatel
Rozsah: Komponenta
Úroveň: Uživatelský cíl
Zúčastněné strany a zájmy:

- Uživatel - Upravit existující článek
- Navrhovaný systém - Zaznamenat úpravu dat do databáze
- Databáze

Předpoklady:

- Funkční připojení k databázi
- Uživatel je přihlášen a má roli administrátora

Minimální plnění: Uživatel získá nazpět informaci o výsledku.
Úspěšné plnění: Uživatel získá nazpět informaci o výsledku a data v databázi jsou změněna.
Spouštěč: Kliknutí na tlačítko uložit na stránce pro úpravu článku
Scénář úspěšného plnění:

1. Uživatel upraví článek na stránce pro úpravu.
2. Uživatel klikne na tlačítko Uložit.
3. Frontendová aplikace pošle na API požadavek na změnu článku s jeho daty.

4. Systém zkontroluje, je-li uživatel přihlášen a má-li roli administrátora
5. Systém zkontroluje validitu dat.
6. Systém spočítá novou cenu článku.
7. Systém článek uloží.
8. Systém na frontendovou aplikaci pošle informaci o úspěšném uložení úprav a ta uživatele přesměruje na detail článku.

Alternativní scénář:

- **Některá z odeslaných hodnot není validní:** Uživatel je upozorněn, která hodnota není validní.

Výjimky:

- **Uživatel není přihlášen:** Uživatel je s informací, že není přihlášen, přesměrován na přihlašovací stránku.
- **Uživatel nemá roli administrátora:** Uživatel je upozorněn, že pro danou funkci nemá dostatečná oprávnění.
- **Upravovaný článek neexistuje:** Uživateli je nazpět vrácena informace o tom, že článek, který se snaží upravit, neexistuje.
- **Systémová chyba:** Uživateli je prezentována informace o chybě v systému.

4.2.5.9 UC9: Vytvoření článku

Hlavní účastník: Uživatel
Rozsah: Komponenta
Úroveň: Uživatelský cíl
Zúčastněné strany a zájmy:

- Uživatel - Vytvořit nový článek
- Navrhovaný systém - Uložit nový článek do databáze
- Databáze

Předpoklady:

- Funkční připojení k databázi
- Uživatel je přihlášen a má roli administrátora

Minimální plnění: Uživatel získá nazpět informaci o výsledku.
Úspěšné plnění: Uživatel získá nazpět informaci o výsledku a data v databázi jsou změněna.
Spouštěč: Kliknutí na tlačítko uložit na stránce pro vytváření článku.
Scénář úspěšného plnění:

1. Uživatel vyplní formulář na stránce pro vytváření článku.
2. Uživatel klikne na tlačítko Uložit.
3. Frontendová aplikace pošle na API požadavek na vytvoření článku s jeho daty.
4. Systém zkontroluje, je-li uživatel přihlášen a má-li roli administrátora
5. Systém zkontroluje validitu dat.
6. Systém spočítá cenu článku.
7. Systém článek uloží.
8. Systém na frontendovou aplikaci pošle informaci o úspěšném vytvoření článku a ta uživatele přesměruje na jeho detail.

Alternativní scénář:

- **Některá z odeslaných hodnot není validní:** Uživatel je upozorněn, která hodnota není validní.

Výjimky:

- **Uživatel není přihlášen:** Uživatel je s informací, že není přihlášen, přesměrován na přihlašovací stránku.
- **Uživatel nemá roli administrátora:** Uživatel je upozorněn, že pro danou funkci nemá dostatečná oprávnění.
- **Systémová chyba:** Uživateli je prezentována informace o chybě v systému.

4.2.5.10 UC10: Smazání článku

Hlavní účastník: Uživatel
Rozsah: Komponenta
Úroveň: Uživatelský cíl
Zúčastněné strany a zájmy:

- Uživatel - Smazat existující článek
- Navrhovaný systém - Smazat článek z databáze

- Databáze

Předpoklady:

- Funkční připojení k databázi
- Uživatel je přihlášen a má roli administrátora

Minimální plnění: Uživatel získá nazpět informaci o výsledku.
Úspěšné plnění: Uživatel získá nazpět informaci o výsledku a článek je z databáze smazán.
Spouštěč: Kliknutí na tlačítko Smazat na detailu článku
Scénář úspěšného plnění:

1. Uživatel klikne na tlačítko Smazat na detailu článku.
2. Frontendová aplikace pošle na API požadavek na smazání článku.
3. Systém zkontroluje, je-li uživatel přihlášen a má-li roli administrátora
4. Systém článek smaže.
5. Systém na frontendovou aplikaci pošle informaci o úspěšném smazání článku a ta uživatele přesměruje na hlavní stránku s informací o smazání.

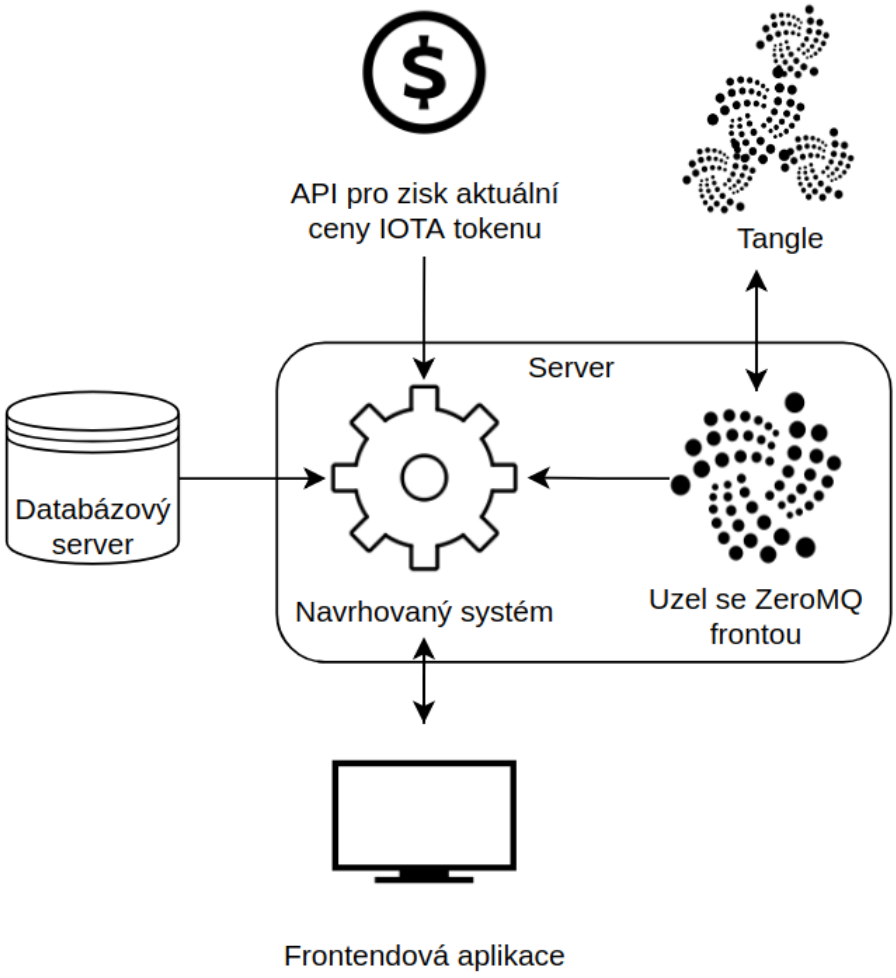
Výjimky:

- **Uživatel není přihlášen:** Uživatel je s informací, že není přihlášen, přesměrován na přihlašovací stránku.
- **Uživatel nemá roli administrátora:** Uživatel je upozorněn, že pro danou funkci nemá dostatečná oprávnění.
- **Upravovaný článek neexistuje:** Uživateli je nazpět vrácena informace o tom, že článek, který se snaží smazat, neexistuje.
- **Systémová chyba:** Uživateli je prezentována informace o chybě v systému.

4.2.6 Technický návrh

4.2.6.1 Systémová infrastruktura

Architektura je ve zjednodušené formě vyobrazena na diagramu na obrázku 4.2.



Obrázek 4.2: Diagram propojení infrastruktury

Z hlediska moderního návrhu bude systém tvořen jako služba, která bude mít veřejně dostupné API. Samotné zobrazení by následně mělo být řešeno formou frontendové aplikace, která získává data právě z tohoto API a zároveň uživateli pomocí API umožňuje systém používat. Serverová služba bude na jedné straně komunikovat s frontendovou aplikací, na druhé straně s uzlem, z nějž bude získávat data ze ZeroMQ a na straně třetí bude komunikovat

pomocí API se službou třetí strany, od které bude získávat aktuální údaje o ceně IOTA tokenu, na jehož základě bude určovat výše transakcí. Uzel bude jedním z uzlů, na kterém bude provozován Tangle. Uzel jako takový bude mít ven vystavenou ZeroMQ frontu, na kterou se připojí systém a bude z ní brát data o příchozích transakcích na adresu, na kterou budou placeny články. Systém bude z důvodu lepšího propojení na stejném serveru jako provozovaný uzel. Do budoucna by bylo dobré dát uzel na samostatný server, ovšem bude potřeba dostatečně dobře vyřešit zabezpečení, aby jediný kdo může s uzlem komunikovat z jiného serveru byl právě navrhovaný systém. Vedle serveru na kterém poběží systém a uzel bude ještě databázový server, na kterém poběží databáze. Ta bude sloužit jako zdroj a úložiště dat pro systém.

4.2.6.2 Problémy spjaté s platbou

Jelikož adopce IOTA protokolu je stále na poměrně malé úrovni, je Tangle nedostatečně využíván a v důsledku toho nejsou v tuto chvíli transakce rozhodně rychlostně srovnatelné s platbami kartou. Z tohoto důvodu je třeba k platbě přistupovat trochu jiným způsobem. Platba jako taková bude muset být založena na asynchronním principu. Běžně se v rámci mikroservis, které fungují na podobném principu k většině asynchronních volání přistupuje tak, že se volání bere jako úspěšné a pouze v případě, že v rámci procesu dojde k chybě, je uživatel o problému informován. Toto bohužel v tomto případě není možné, jelikož nemůžeme uživateli rovnou dát přístup k článku, když nemáme jistotu, že platba proběhla. V tomto případě tedy bude proces probíhat podobným způsobem jako při platbě bankovním převodem. Uživatel dostane podklady k provedení platby. Platba se zaznamená v rámci systému a ten bude sledovat ZeroMQ frontu uzlu a z ní získávat všechny transakce. Ve chvíli, kdy zaznamená transakci se zprávou a výší odpovídající ještě neprovedené platbě (zpráva bude fungovat jako variabilní symbol), transakci zaznamená a začne sledovat, jestli je potvrzená sítí. Jakmile k potvrzení dojde, bude platba brána za provedenou a na základě toho bude uživateli přidělen jeho zakoupený článek.

Dalším problémem, který je třeba v rámci plateb pomocí IOTA tokenu řešit, je volatilita. Kryptoměny jsou známé tím, že jejich cena velmi kolísá a to i v řádu jednotek až desítek procent v rámci hodiny. V extrémních případech i v rámci několika minut. Z tohoto důvodu je třeba často aktualizovat cenu IOTA tokenu v systému. K tomuto bude sloužit API třetí strany a to API aplikace CoinMarketCap. Tato aplikace zaznamenává průměrnou cenu jednotlivých kryptoměn na základě průměru z více trhů. Díky tomuto je vhodná pro využití v rámci navrhovaného systému. Data v API jsou aktualizována jednou za minutu a proto se i v systému budou aktualizovat v této periodě.[41]

S volatilitou je spojen ještě jeden problém, který je nutné řešit. Při bankovním převodu je v pořádku, když uživatel zaplatí například až po dvou dnech. Jelikož ale cena velmi kolísá, mohlo by v případě pozdního zaplacení

dojít k poklesu ceny tokenu a uživatel by tedy ve skutečnosti zaplatit reálnou částku menší. V opačném případě by mohlo dojít k nárůstu ceny a uživatel by naopak platil více. Z tohoto důvodu bude platnost platby omezena pouze na 15 minut. Pokud transakce nebude zaplacena do 15 minut, bude neplatná a uživatel bude muset vygenerovat transakci novou.

V rámci systému je ještě třeba nějakým způsobem určovat cenu článku. Nedává velký smysl mít cenu článku statickou. Měla by se optimálně odvíjet od délky článku. I samotní redaktoři jsou často placeni od počtu normostran. Z tohoto důvodu by největší smysl dávalo naceňovat články podle délky. Není ale nutné se zaměřovat na počet normostran, ale můžeme brát počet slov, případně přímo počet znaků. Normostrana v České republice je dlouhá 1800 znaků. Můžeme tak říci, že chceme například 2 centy za normostranu a odtud můžeme jednoduše zjistit cenu za jeden znak. Cenu článku pak následně dostaneme jednoduše vynásobením počtu znaků cenou za jeden znak.

4.2.6.3 API

Při tvorbě návrhu API je třeba nejprve rozhodnout, která API architektura bude pro tento typ rozhraní nejvhodnější. Prakticky je třeba se rozhodnout mezi třemi architekturami. Těmi jsou REST, RPC a GraphQL.

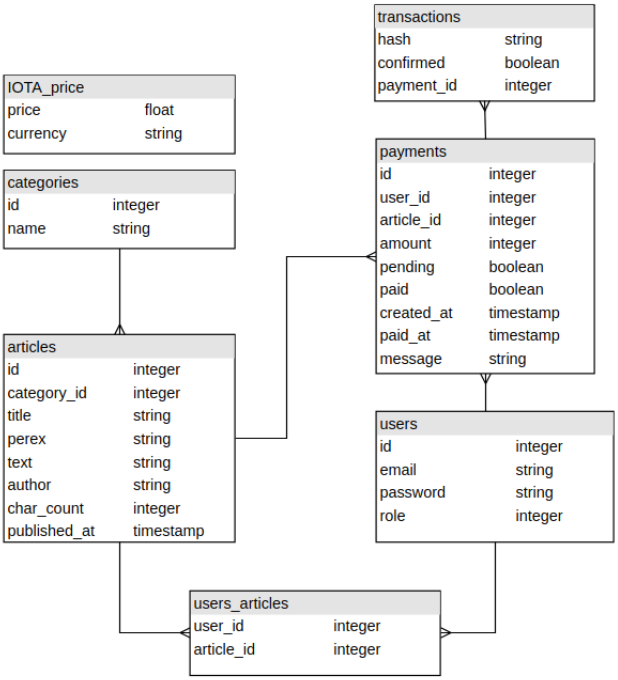
GraphQL je z těchto tří architektur nejnovější a jako takové je stále poměrně ve vývoji a není pevně standardizované. Jednotlivá volání jsou v podstatě přímo dotazy, které kromě filtrování umožňují například v rámci jednoho dotazu získat několik zdrojů. Tato architektura byla vyvinuta společností Facebook v poslední dekádě a jako taková se stala poměrně oblíbenou, především mezi mobilními a frontend vývojáři, tedy těmi, kteří staví frontendové aplikace. Nevýhodou je zatím stále poměrně malá rozšířenost a neucelenost.[42]

RPC je naopak nejstarší formou API. Jedná se o návrh orientovaný na funkce, ne na zdroj jako takový. Tento typ API byl poslední dobou upozaděn v důsledku popularity objektového návrhu, avšak s nástupem moderních návrhových aplikačních vzorů se začíná vracet do módy. Přináší s sebou však řadu problémů. Jelikož se točí úzce kolem samotných funkcí systému, je třeba API často měnit na základě změn v systému. Zároveň v případě tohoto vzoru jsou problémy s orientací v API, jelikož neumožňuje jednoduchou cestou nalézat nové koncové body.[43]

Posledním probíraným vzorem je REST. Tento vzor je prakticky opakem RPC. Jedná se o API zaměřené přímo na jednotlivé zdroje. Z tohoto důvodu je vhodné do aplikací, jejichž hlavní podstatou je manipulace se zdroji (jako v rámci navrhovaného systému - čtení článku, listování články, vytváření článku, editace článku, ...). Tento vzor s sebou přináší jednoduchost orientace v rámci API, díky možnostem odkazů na další zdroje. Problémem je to, že přenášená data jsou často značně rozsáhlejší, než například v případě GraphQL nebo RPC. Z podstaty navrhované aplikace ale toto není problémem a API tedy bude stavěno s využitím vzoru REST.[44]

4.2.7 Databáze

Databázové schéma je v podstatě velmi jednoduché. Hlavními entitami, které nás zajímají jsou uživatelé a články reprezentované tabulkami articles a users. Samotné schéma je vyobrazené na obrázku 4.3.



Obrázek 4.3: Schéma databáze

Články uchovávají základní informace o názvu, perexu, textu, autorovi a kategorii. Zároveň informaci o datu kdy byl článek publikován a v poslední řadě počet znaků, který slouží jako podklad k výpočtu jeho ceny.

Důležitou tabulkou je tabulka plateb nesoucí název payments. Tato tabulka zaznamenává vytvořené platby. Platba nese především informaci o uživateli, který jí inicioval a článku, který je předmětem platby. Společně s tím zaznamenává výši platby, která má být provedena, zprávu pomocí které se identifikuje platba v rámci Tangle a v poslední řadě potom informaci o tom, kdy byla vytvořena a na to navazující informaci, jestli již byla zaplacena, nebo informaci o tom, že selhala z důvodu, že nebyla zaplacena do 15 minut. Tato tabulka má důležitý vztah s tabulkou transactions, která uchovává všechny transakce v Tangle, které jsou s platbou spojené. Jak už bylo řečeno v rešeršní části, je možné, že vytvořená transakce se dostane do části Tangle, která není přijata a je nutné tedy provést takzvaný „reattach“ transakce, tedy vytvoření její nové kopie, která je již v lepší části Tangle a má tedy větší šanci na

to, že bude přijata. Takovýchto kopií může vzniknout několik. Je tedy třeba uchovávat všechny tyto transakce, protože přijatá následně bude sítí pouze jedna a my tedy potřebujeme být schopní ověřit, že sítí přijatá transakce je ta, která nás v rámci platby za článek zajímá.

Poslední hlavní tabulkou je tabulka ursers_articles, která reprezentuje M:N vztah mezi články a uživateli. Do této tabulky jsou přidávány záznamy na základě provedené platby. Pokud uživatel provede platbu, je zde zaznamenáno, že tento uživatel má zakoupený článek, který byl předmětem platby.

Vedle těchto tabulek ještě stojí tabulka IOTA_price, jejímž jediným účelem je zachovávat aktuální cenu IOTA tokenu vzhledem k určité měně. Tato tabulka není vyloženě nutná, jelikož tato informace by se mohla uchovávat přímo v samotné aplikaci, ale v případě potenciálního vícevláknového přístupu je lepší mít hodnotu v databázi, která si sama pohlídá atomický přístup k hodnotě.

4.3 Implementační část

V rámci implementační části budou rozebrány aspekty samotné implementace systému. Bylo již řečeno, že se bude jednat o backendovou část aplikace, která bude implementována jako API. Kapitola jako taková se bude zaobírat výběrem jednotlivých technologií využitých k implementaci, samotnou implementací jako takovou, tedy strukturou kódu a nakonec porovnáním implementace s analýzou a důvody, proč se případně implementace odlišuje od toho, co bylo původně navrženo v rámci analytické části. Poslední částí bude zhodnocení implementace a doporučení pro další postup.

4.3.1 Výběr technologií

Jak bylo již řečeno v rámci rešerše i analýzy, s uzlem, který slouží jako vstupní brána do Tangle, je možné komunikovat jednoduše pomocí HTTP požadavků na jeho API. To ovšem není až tak úplná pravda. Uzel jako takový má vystavené API, ovšem to je třeba volat se specifickými daty, jako například s daty v ternární soustavě. Být nucen toto dělat ručně v rámci kódu přidává zbytečnou složitost a proto je lepší využít jednu z knihoven od IOTA Foundation. Pravdou je, že v rámci doposud probraných funkcionalit systému není třeba volat přímo API. Systém bude pouze číst data ze ZeroMQ fronty. Nebylo by tedy vyloženě nutné volit technologii v závislosti na IOTA Foundation knihovny. Pokud ale budeme brát v potaz fakt že by se systém měl do budoucna rozšiřovat, například o možné vracení plateb a podobné funkcionality, je dobré myslet dopředu a volit jazyk, který umožňuje využití IOTA Foundation knihovny. Zároveň je vhodné využít knihovnu na komunikaci se ZeroMQ frontou. Z tohoto důvodu bude API (část implementovaná v rámci této práce) implementováno v jazyce **Node.js**. Prvním důvodem je to, že Node.js knihovna pro práci s rozhraním uzlu je jediná z oficiálních knihoven

IOTA Foundation, která u sebe nemá informaci o tom, že je v beta verzi, nebo je její využití na vlastní nebezpečí. Dalším důvodem je to, že ZeroMQ má ve stejném jazyce velmi dobrou knihovnu na poslouchání fronty. Třetím důvodem je to, že i bez knihovny je třeba poslouchat frontu, k čemuž nejsou vhodné jazyky, jako je například PHP, které mají problém běžet dlouhodobě, jelikož jsou stavěny na podstatě, že s každým požadavkem se pustí určitý skript a aplikace se následně opět ukončí. Je vhodné, aby aplikace běžela dlouhodobě a nebylo třeba ji při každém požadavku pouštět, jako je třeba právě v případě zmiňovaného PHP. Node.js je tedy z těchto důvodu vhodnou volbou.

V případě databáze jsem zvolil **MongoDB**. Jedná se o objektově orientovanou NoSQL databázi. NoSQL je obecně databáze založena na jiném konceptu než na relačních tabulkách. Z tohoto důvodu je pro tento případ velmi vhodná. Data v této databázi jsou uložena jako jednotlivé dokumenty shlukovány v takzvaných kolekcích. Jelikož se jedná o poměrně jednoduchou formu práce s daty, není tak databáze zbytečně zatěžována a díky tomu umožňuje vysokou výkonnost, dostupnost a zároveň škálovatelnost. Data jsou uchovávána ve formátu JSON a samotná instalace databáze je také velmi jednoduchá. Díky těmto vlastnostem je vhodná především pro systémy pracující na živých datech, potřebující vysokou dostupnost. Stejně tak je velmi vhodná pro menší, stále běžící systémy. Problémem však může být potom například fakt, že MongoDB neumožňuje procedury, nebo prakticky jakoukoliv logiku přímo v databázi. Stejně tak není stavěná na práci s transakcemi. Protože ale moderní architektura se velmi točí okolo mikroservis, NoSQL databáze a především MongoDB získávají na stále větší popularitě. To v kombinaci s vlastnostmi, které jsou vhodné pro systém navrhovaný v rámci této práce a zároveň s přihlédnutím k jednoduchosti, je MongoDB vhodnou volbou.

4.3.2 Instalace plného uzlu

Jak již bylo řečeno, pro provoz aplikace je třeba mít v provozu zároveň svůj plný uzel. Za tímto účelem je nutné jej nejprve nainstalovat a zprovoznit.

Uzel se instaluje poměrně jednoduše. V první řadě je třeba naklonovat projekt z GitHubu. Pomocí příkazu:

```
$ git clone https://github.com/iotaledger/iri
```

Toto vytvoří složku s názvem iri. Do ní je třeba následně vstoupit a projekt zkompilovat pomocí Maven a to příkazem:

```
$ mvn clean compile && mvn package
```

Po kompilaci vznikne složka s názvem target ve které je spustitelný .jar program. Tím je následně spouštěn uzel. Před spuštěním je vhodné ještě vytvořit soubor s názvem iota.ini. V tomto souboru je možné upravovat jednotlivá nastavení. Například na jakém portu uzel komunikuje. Při spuštění

si aplikace sama vezme z tohoto souboru konfiguraci, případně je možné přepínačem nastavit soubor jiný. Jelikož ke správnému fungování systému budeme potřebovat funkční frontu ZeroMQ, je třeba ji přidat do konfigurace v iota.ini a to pomocí následujících řádků:

```
ZMQ_ENABLED = true
ZMQ_PORT = 5556
```

ZMQ_PORT může mít samozřejmě jakýkoliv jiný validní port, 5556 je pouze standardně používaným portem. Jednou z důležitých věcí, které je potřeba nastavit v rámci konfigurace uzlu jsou sousední uzly. Tedy uzly s kterými provozovaný uzel na přímo komunikuje. Bez těchto uzlů není možné, aby provozovaný uzel fungoval. Jednou z možností je zadat tyto uzly na pevno do konfiguračního souboru a to do položky NEIGHBOURS, kam se píší jednotlivé url uzlů, například ve formátu udp:http://somenode.com:14600. Toto je jednou z možností. Je třeba však nějaké uzly najít a zároveň mít jistotu, že fungují dobře, řešit i lokalitu ve které se nachází a podobně. Z tohoto důvodu vznikl projekt Nelson. Jeho autorem je Vitaliy Semko a jedná se o aplikaci na automatické odhalování a propojování uzlů v síti. Funguje na principu epoch, kdy v rámci každé epochy, která trvá několik minut, jsou uzly propojovány a hodnoceny dle jejich výkonnosti, na základě čehož jsou v další epoše propojeny opět jinak. Díky tomu je propojení na optimální úrovni a provozovatel uzlu zároveň nemusí sousední uzly sám kontrolovat a nalézat.

Instalace Nelson je také poměrně jednoduchá. Důležité je mít nainstalovaný Node.js a to ve verzi alespoň 8.9.4. Poté už stačí nainstalovat Nelson pomocí npm:

```
$ npm install -g @semkodev/nelson.cli
```

Puštění uzlu a služby na vyhledávání sousedů je pak již jednoduché puštění dvou aplikací:

```
$ java -jar iri-1.5.5.jar
$ nelson --getNeighbors
```

1.5.5 v rámci iri souboru je číslo aktuální verze. V případě, že není port uveden v rámci iota.ini souboru, měl by být předán pomocí přepínače -p. Pokud není iota uzel provozován na standardních portech, je pak třeba také Nelson nastavit pomocí konfiguračního souboru. Ten se aplikaci předá pomocí –config přepínače a v konfiguračním souboru jsou potom porty uvedeny jako:

```
IRIPort = 14265
TCPPort = 15600
UDPPort = 14600
```

[45][46]

4.3.3 Systémová architektura

Jedním z problémů u knihovny Node.js je systémová architektura. Vzhledem k tomu, že se jedná o poměrně novou technologii, začíná být teprve ve větším měřítku aplikovaná. Z tohoto důvodu se stále vedou rozsáhlé spory o tom, jak je nejlepší stavět architekturu systému programovaného v Node.js. Neexistuje zatím žádný použitelný globální framework, který by toto řešil za vývojáře tím, že by mu vnutil svou architekturu, jak je tomu například u PHP frameworků jako jsou Laravel nebo Symfony. Toto dává vývojáři do značné míry volnost, která může být žádaná, ale zároveň může při nevhodném rozvržení způsobovat problémy. Vzhledem k tomu, že Node.js se většinou používá na stavbu menších backendových servis (v češtině se občas používá slovo servisa, které působí zvláštním dojmem a v dalším textu budu tedy místo servisa nebo service používat český překlad „služba“), není naštěstí třeba dělat přehnaně složité struktury, které jsou žádané u velmi objemných systémů. Z tohoto důvodu také na Node.js neexistují velké moduly, ale většina používaných modulů jsou menšího rozsahu, zaměřující se na určitou specifickou problematiku.

Jedním z hlavních používaných modulů je **Express**. Tento modul se stal v rámci vývoje webových aplikací na Node.js téměř standardem a je možné jej nalézt na většině běžících projektů. Jedná se v podstatě o mikroframework, který značně zjednodušuje základ projektu a to především routing a middleware. Z tohoto důvodu jej budu také v rámci práce využívat. Vzhledem k tomu, že je cílem vytvořit API, jedná se o velmi vhodnou volbu.

Další důležitou částí je datová vrstva. Jednou z možností je vytvořit si vlastní vrstvu, avšak v této době kdy existuje mnoho dostupných open source ORM, bude rozhodně lepší využít nějaké již hotové. V rámci toho jsem se rozhodl využít **Mongoose**. Jedná se o ODM modul na Node.js. Jeho hlavní výhodou je jeho jednoduchost použití. Mongoose je založený na schématech, které reprezentují jednotlivé dokumenty. Díky tomu umožňují například definovat datové typy atributů schémat, nebo validaci dat při manipulaci s nimi. Společně s tím modul umožňuje dotazování do databáze jednoduchou formou. Z těchto důvodů je Mongoose nejpoužívanějším ODM pro MongoDB.

V rámci vývoje je využito ještě několik dalších modulů, které budou případně zmíněny v kapitole o problematice, které se týkají. Poslední důležitou věcí je samotné rozdělení aplikace. Jak bylo již řečeno, je stále vedena diskuze o nejlepší architektuře v rámci Node.js. Někteří lidé nepoužívají controllery, někteří ano a dávají logiku přímo do nich, někdo používá repozitáře, nebo služby, nebo jako služby vnímá pouze napojení na služby externí, například API jiných systémů. V tomto případě jsem se rozhodl pro následující rozložení, které je vidět na schématu 4.4 a bude dále rozebráno více do podrobna.

Složka **bin** obsahuje spustitelné skripty. V tomto případě pouze skript **www**, který vystupuje jako skript, s jehož pomocí se aplikace pouští. Jedná se v podstatě o ekvivalent vstupních skriptů index.php v PHP frameworcích jako je Symfony.

- `bin` ..... obsahuje spustitelné jádro aplikace a další spustitelné skritpy
- `config` ..... konfigurační souboury
- `controllers` ..... základní controllery
- `middlewares` ..... složka všech vlastních middlewarů
  - `validators` ..... specifické middlewary fungující jako validátory http požadavků
- `models` ..... schémata jednotlivých databázových dokumentů
- `node_modules` ..... složka obsahující všechny použité externí moduly
- `public` . složka pro případné soubory, které by měly být veřejně dostupné
- `routers` ..... routery aplikace
- `services` ..... funkce, které jsou využívány v controllerech nebo dalších službách
  - `Article` ..... služby týkající se článků
  - `IOTAPrice` ..... služby týkající se ceny IOTA tokenu
  - `Payment` ..... služby týkající se plateb
  - `Transaction` ..... služby týkající se transakcí
  - `User` ..... služby týkající se uživatelů
- `app.js` ..... inicializační skript aplikace

Obrázek 4.4: Rozložení aplikace

config je složka obsahující konfigurační soubory aplikace. Jelikož se jedná o pouhou ukázkovou aplikaci, je v této složce aktuálně pouze jeden soubor **config.json**. Ten obsahuje veškerou konfiguraci na jednom místě. Jelikož je nastavení poměrně málo, není to problém. V normálním případě by byla konfigurace rozdělena na více souborů, které by nezávisle zachovávaly například přístupové údaje k databázi. Soubor obsahuje objekt s konfiguracemi v JSON formátu. Jediná konfigurace je nyní vývojová nazvaná „dev“. Ta je také staticky použita v inicializačním skriptu, kde se vytváří globální proměnná config obsahující tuto konfiguraci. V budoucnu by se měla konfigurace brát dynamicky na základě prostředí, v kterém se aplikace nachází.

Složka **controllers** je složkou která uchovává všechny controllery aplikace. Controller obsahuje funkce, kterým jsou z routeru předávány HTTP požadavky. Jak již bylo řečeno výše, někdo přímo do controlleru dává i logiku. Dle mého názoru by funkce controlleru logiku obsahovat neměly a měly by pouze vzít požadavek, v případě potřeby provést jeho validaci a následně zavolat určitou službu, která data zpracuje a nazpět vrátí výsledek, který pak controller případně zpracuje a v určitém formátu odešle jako odpověď. Takto jsem také s controllery pracoval a pouze v nich v případě potřeby validoval požadavek a transformoval data, která jsou následně předána jiné službě na zpracování. Z ní jsou poté vrácena data, na jejichž základě controller odešle odpověď.

middlewares složka obsahuje middlewary. Middleware je jedna z hlavním

složek frameworku Express.js. Celý framework je prakticky založen pouze na routerech a middlewarech. Z pohledu frameworku je middleware jakákoliv funkce, která má přístup k požadavku a odpovědi. Vstupují tedy do životního cyklu požadavku a mohou požadavek měnit, ukončit a zároveň volat další middleware v pořadí a předávat mu data. V rámci životního cyklu požadavku jde tedy v podstatě pouze o průchod sérii middlewarů. Z pohledu frameworku je tedy i to, o čem jsem v minulém odstavci mluvil jako o controllerech, pouze middleware. Toto je pravda a důvod proč jsem rozlišil middleware na dvě různé části, je především z tradičních důvodů. Controller je tradičně používaný pojem v rámci webových frameworků a je z něj tedy jasné, jakému účelu slouží. Samotný framework využívá několik middlewarů v základu, ty se starají o klasické úkoly, jako je parsování těla požadavku, zakódovávání odpovědi do JSON formátu a podobné. Složka middlewares tedy poté obsahuje dodatečné middlewary, které jsou používány především ke kontrole dat v požadavcích. Tento middleware je v podsložce **validators**. K nim je ještě middleware, který se stará o kontrolu přístupového tokenu v případě, že se uživatel snaží přistoupit ke zdroji, k němuž je přístup určitým způsobem omezen. Tento middleware tedy zkontroluje, zda-li má uživatel k danému zdroji přístup a v opačném případě ukončí cyklus požadavku se standardní odpovědí o zamítnutí přístupu.

Složka **models** obsahuje definice všech schémat dokumentů v databázi. Mongoose je celý založený právě na schématech, bez kterých není prakticky možné se do databáze dotazovat. Schémata tedy obsahují definice dokumentů, tedy jednotlivé parametry s datovými typy a případně jejich validátory. Určité moduly poté obsahují i akce, které se mají provádět vždy před uložením objektu do databáze, nebo při jiné události.

routers je složka, která obsahuje všechny routery. Ty jako takové berou požadavek, přiřadí mu další middlewary, kterými musí projít a následně ho přepošlou dalšímu middlewaru, kterým je právě funkce v controlleru.

Složka **services** je poslední hlavní složkou a zároveň jednou z nejdůležitějších. Jako taková obsahuje všechny vlastní služby, které jsou v aplikaci využívány Jedná se nejen o funkce, které jsou využívány v controllerech, ale například i o periodicky volané služby, nebo například napojení na ZeroMQ frontu.

V rámci kapitoly byla popsána základní architektura aplikace a jednotlivé části, které dohromady dávají celou její funkčnost. U většiny komponent bylo pouze nastíněno jejich fungování a jako takové bude u většiny více probráno v dalších kapitolách.

4.3.4 Routing

Routing je základem téměř každé webové aplikace. Uživatel odešle požadavek na server, kde uvede URI, kterým identifikuje zdroj ke kterému chce přistoupit. Aplikace následně pomocí odpovědi uživateli předá požadovaná data. Dříve URI většinou odpovídalo přímo cestě na serveru, ke které chtěl uživatel při-

stoupit. Nyní se již dělá jeden vstupní bod, v tomto případě skript www ve složce bin, na který jsou přesměrovány všechny požadavky. Ten většinou provede určité úkony a následně je požadavek předán právě routeru, který rozhodne, kam dále má být požadavek předán.

V případě Express.js je router dalším middlewarem. V případě vyvíjené aplikace je routerů několik a to každý pro jeden zdroj, ke kterému je třeba přistupovat. V rámci aplikace stačí, aby API umožňovalo přístup k uživateli, článku, platbě a kategorii článku. Existují tedy 4 routery pro tyto 4 zdroje. Požadavek následně prochází postupně jedním routerem za druhým v takovém pořadí, jak jsou registrovány v app.js. V každém routeru jsou specifikovány kombinace HTTP metod a části url obsahující cestu, na základě čehož se rozhoduje, jestli požadavek splňuje právě tuto podmínku. Pokud tuto podmínku splňuje, je následně požadavek poslán do funkce specifikované v podmínce.

Ke specifikacím se používá vestavěný router v Express.js a specifikace vypadá například následujícím způsobem:

```
router.get('/user/:id', [authMiddleware.isLoggedIn,
idValidator.isValidId], articleController.getById);
```

Tento kód říká, že pokud je HTTP metoda v požadavku GET a cesta obsažená v URI je ve formátu /user/:id, kde :id je cokoliv (:id je následně součástí požadavku jako id parametr), pak má požadavek jít skrze middleware authMiddleware.isLoggedIn, který kontroluje, jestli je uživatel přihlášen a pokud jím úspěšně projde, je následně poslá dál do middlewaru idValidator.isValidId, který kontroluje validitu id parametru v cestě. Po úspěšném průchodu těmito middlewary je požadavek poslán do articleController.getById funkce.

Aby nebylo v routeru, který je celý pro zdroj uživatele (user), nutné definovat v každé cestě /user, může být v app.js u registrace routeru zadán prefix, který musí cesta obsahovat, aby byl pro požadavek použit a následně v samotném routeru už tento prefix není třeba uvádět. Pokud tedy máme router pro uživatele, kde chceme, aby všechny cesty začínaly /user, můžeme tento router v app.js registrovat následujícím způsobem:

```
app.use('/users', usersRouter);
```

Aplikace ctí REST princip návrhu. Všechny cesty ke zdrojům jsou tedy založeny na stejném formátu. Cesta začíná nejprve identifikací samotného zdroje, tedy například pro uživatele /users. Dále cesta obsahuje případně identifikační parametry nebo vztahy k dalším zdrojům. Všechny další parametry jako jsou například filtrační parametry nebo stránkování, jsou už pouze jako GET parametry. Jedinou výjimkou, kdy je url v jiném formátu, je v rámci uživatelské autentizace, která je registrována pod url /users/auth.

4.3.5 Datové modely

Jak bylo již výše uvedeno, Mongoose je založen na schématech a není bez nich možné se na dokumenty v databázi dotazovat. Datové modely jsou tedy schémata, která reprezentují kolekce dokumentů, které mohou být chápány jako ekvivalent tabulek v relační databázi. Kolekce kopírují schéma na obrázku 4.3. Jediným rozdílem je názvosloví, kdy vzhledem ke zvyklostem v Javascriptu je pro názvy proměnných a dokumentů používán camelCase (jak lze vidět, jednotlivá slova navazují přímo na sebe a každé další slovo začíná velkým písmenem) namísto snake_case (jednotlivá slova jsou rozdělena podtržítkem). Zároveň parametr id se dle konvence MongoDB jmenuje _id.

Schéma je seznam jednotlivých parametrů dokumentu s jejich datovými typy a případně informací o tom, jestli je povinný, nebo o výchozí hodnotě. Tam kde je to vhodné, je zároveň vytvořen validátor na hodnotu parametru, který při ukládání automaticky zkontroluje, jestli vkládaná data odpovídají zadané podmínce a v opačném případě vrátí chybu obsahující zadanou zprávu. Validátor je tvořen následujícím způsobem:

```
text: {
    type: String,
    validate: {
        validator: function (v) {
            return v && v.length > 0;
        },
        message: 'Text must be present and not empty.'
    }
}
```

Některé objekty jsou zároveň rozšiřovány pluginem mongoose-paginate, který umožňuje jednoduché stránkování při výpisu dokumentů.[47]

Jelikož MongoDB je databáze založená na dokumentech, nedají se tvořit vztahy mezi dokumenty stejnou formou jako v klasické relační databázi. Možnosti, jak mezi sebou dvě entity provázat ale existují. První možností je replikovat data jedné entity do druhé. Toto je dobré při čtení, jelikož je ušetřen čas, který je v relační databázi stráven dotahováním dat o druhé entitě. Problémem je ale duplikování dat a tím pádem při změně jedné entity nutnost změnit ji i na všech místech, kde se vyskytuje. Z tohoto důvodu jsem se rozhodl pro druhou možnost a tou je využití datového typu ObjectId, které umožňuje uchovat referenci pomocí identifikátoru. Při výběru dokumentu z databáze je poté možné použít funkci populate(), která automaticky naplní tuto referenci daty z referenčního dokumentu. Například reference na dokument Category, reprezentující kategorii do které článek spadá, je vytvářena takto:

```
category: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Category'
}
```

4.3.6 Formát odpovědí

Jelikož je komunikace v rámci API na poměrně jednoduché úrovni, všechna data jsou zasílána ve formátu JSON. Jedná se o modernější a používanější řešení než je XML a to především z důvodu jeho jednoduchosti. Všechny odpovědi v případě úspěšného volání jsou jednoduše javascriptovým objektem převedeným do JSON, nebo polem objektů převedených do JSON. Odpovědi jako takové neobsahují stavové kódy a další věci ve speciálním formátu, ale využívají se standardní hlavičky. V případě úspěšných volání se využívají dva kódy: 200 pro prakticky všechna volání, pouze s výjimkou vytváření nového zdroje, který vrací standardní stavový kód 201.

V případě neúspěchu je v těle odpovědi JSON objekt obsahující parametr errorMessage s chybovou zprávou. Stavové kódy chyb jsou opět zasílány v hlavičce odpovědi.

4.3.7 Promise

Node.js je poměrně specifický v tom, jak funguje. Na rozdíl od většiny ostatních synchronních jazyků, je Node.js založen na asynchronním vstupu a výstupu. Velká část funkcí běží asynchronně a z tohoto důvodu je třeba s kódem pracovat jinak, než při standardním programování. Hlavními funkcemi, které v Node.js běží asynchronně, jsou funkce používané při práci se souborovým systémem, HTTP požadavcích a práci s databází. Jelikož vytvářená aplikace je z většiny tvořena právě prací s databází a HTTP požadavky, je většina běhu asynchronní. To s sebou přináší na jednu stranu určitou efektivitu a podporu moderní event-driven architektury, neboli architektury řízené událostmi. Na druhou stranu je ale třeba ke kódu přistupovat jiným způsobem, jelikož výstupní hodnoty z asynchronních funkcí nedostáváme okamžitě, ale kód pokračuje dále bez nich.

Jednou z možností je tradiční využití zpětného volání, neboli takzvaného callbacku. Každá asynchronní funkce bere jako poslední parametr funkci, která je zavolána po dokončení běhu. Jedná se o možnost, od které se postupně upouští, především proto, že nastávaly situace, kdy takto bylo nutné dát do sebe například 4 zpětná volání.

Poměrně novým a lepším řešením, představeném v ECMAScriptu 2015, je takzvaná Promise. Jedná se o objekt, který reprezentuje výsledek asynchronního volání. Má 3 stavy, kterými jsou pending (čekající), fulfilled (splněný) a rejected (zamítnutý). Promise jako taková začíná v čekajícím stavu. Ob-

sahuje funkci, která jako vstup bere resolve a reject funkce. S těmi se následně volá výsledek, který lze odchytit. Ukázková Promise vypadá takto:

```
function doSomethingAsync(inputData) {
    return new Promise(function(resolve, reject)){
        someAsyncCall(inputData, function(err, result) {
            if(err){
                reject(err);
            } else{
                resolve(result);
            }
        });
    }
}
```

Na ukázce kódu je vidět funkce doSomethingAsync(), která provádí asynchronní volání funkce someAsyncCall(). Ta má zpětné volání ve kterém se zkontroluje, jestli nedošlo k chybě a pokud ano, vrací Promise volání reject() s chybou. V případě úspěchu vrací resolve() s výsledkem volání. Díky tomuto není třeba při volání doSomethingAsync specifikovat další zpětné volání, ale je možné s voláním pracovat následujícím způsobem:

```
doSomethingAsync('data')
    .then((result) => {
        doSomethingWithResult(result);
    })
    .catch((error) => {
        console.log(error);
    })
```

Funkce then, volaná na Promise, získává data, která byla předána do resolve() funkce. Catch naopak chytá případnou chybu v reject(). Toto samo o sobě není nic speciálního. Velkou výhodou je ale možnost řetězit volání. Tedy následujícím způsobem řetězit .then, které vracejí Promise a následně odchytit chybu pouze jedním .catch:

```
doSomethingAsync('data')
.then((result) => {
    return someFuncThatReturnsPromise(result);
})
.then((result) => {
    return someOtherPromise(result);
})
.then((result) => {
    doSomethingWithTheResult(result);
```

```
})
.catch((error) => {
    console.log(error);
})
```

Díky tomuto je zápis velice kompaktní. Pokud by stejná logika byla zapsána s použitím zpětných volání, jednalo by se o čtyři zpětná volání v sobě, kde v každém volání by se musela zpracovávat případná chyba zvlášť. Kromě velké nečitelnosti by tedy byl kód o mnoho delší a opakovaný.

Z tohoto důvodu je na většině míst využívána právě Promise. Téměř všechny funkce ve službách, které potřebují vracet data, Promise využívají. V controllerech je pak zpracována a na základě toho jsou odeslána data v HTTP odpovědi.

4.3.8 Autentizace a autorizace

Ne všechny zdroje a metody s nimi spojené jsou veřejně dostupné. Za tímto účelem je třeba v aplikaci řešit autentizaci, která je procesem ověření, že uživatel je tím za koho se vydává a autorizaci, která na základě toho, kdo je uživatelem, umožňuje určité úkony.

Pro tyto potřeby je v rámci aplikace používána knihovna jsonwebtoken. Jedná se o velmi hojně využívanou knihovnu, která jednoduchou formou generuje autentizační token, s jehož pomocí po přihlášení uživatel prokazuje svoji identitu. Po přihlášení je zavolána knihovní funkce sign(), které je jako argument předán objekt dat, v tomto případě objekt obsahující id přihlášeného uživatele, jako další argumenty je pak předán neměnný tajný řetězec a objekt s nastavením, jako je například doba expirace. Tato funkce následně vrátí token, který je předán v odpovědi uživateli. Tento token uživatel následně zasílá v hlavičce požadavků jako x-access-token. Pro ověření je poté zavolána verify() funkce, které je předán token a opět stejný tajný řetězec. Pokud je token validní, je navrácen objekt s daty, z čehož je získáno id přihlášeného uživatele, u kterého je následně pomocí databáze možné ověřit, jakou má roli, což ovlivňuje k jakým zdrojům má přístup. Druhou možností je dát tuto roli přímo do dat, které drží token. V takovém případě by ale při případné změně role byla změna zaznamenána až při expiraci starého a vygenerování nového tokenu.

4.3.9 Kontrola plateb s pomocí IOTA modulu a ZeroMQ

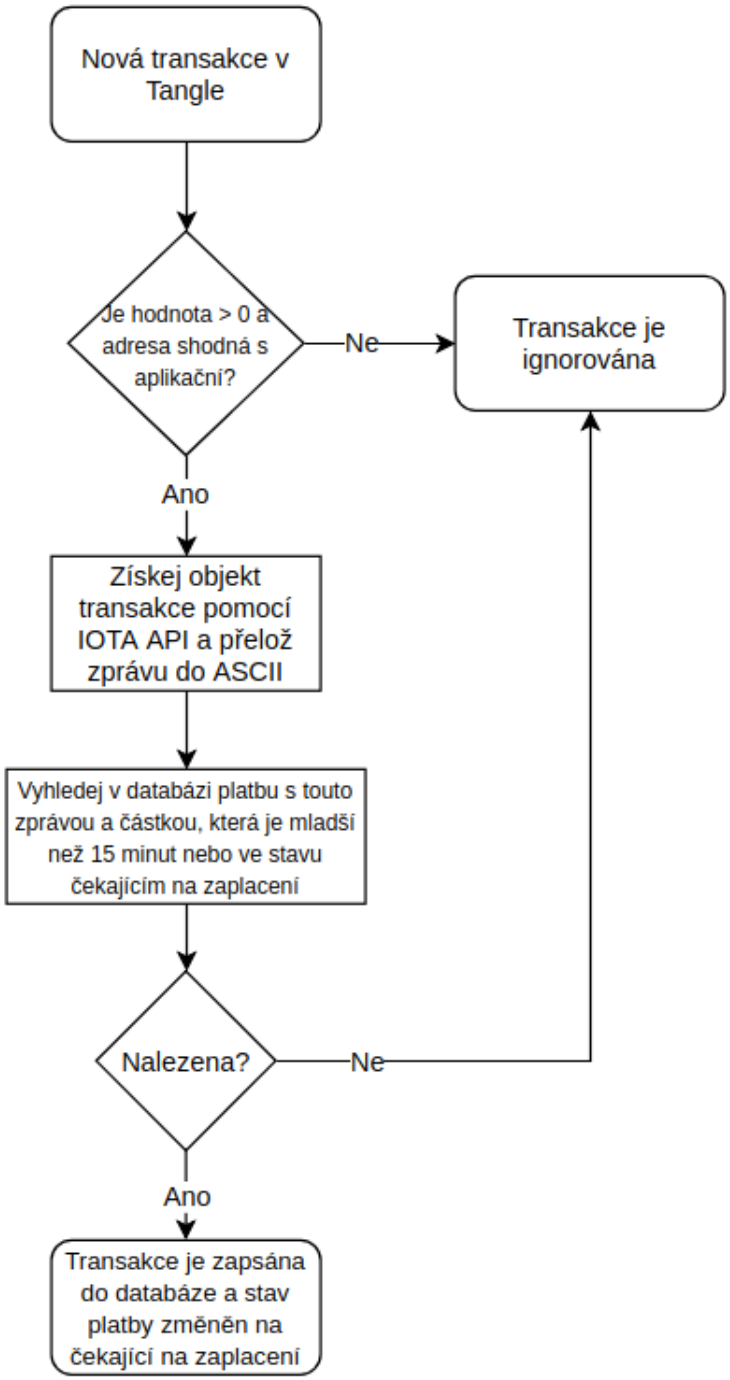
Vzhledem ke stále poměrně vysoké volatilitě IOTA tokenu je třeba cenu článku vždy zmrazit uživateli pouze na omezenou dobu. Doba na kterou se cena zákazníkovi garantuje bývá většinou 15 minut, takovouto dobu například používá společnost Alza při platbě pomocí Bitcoinu a Litecoinu. Jelikož ale, jak bylo popsáno v rešeršní části, se může stát to, že při založení transakce uživatelem se tato transakce dostane do části Tangle, která následně není

přijata a tím pádem i transakce zůstává nepřijata, je třeba s touto možností pracovat.

V případě, že transakce zůstane delší dobu nepotvrzená, je třeba provést takzvaný reattachment. Jedná se v podstatě o vytvoření kopie původní transakce. Tato kopie má většinu hodnot stejnou jako originální transakce. Jednou z důležitých hodnot, která je ale jiná, je hash transakce, tedy její unikátní identifikátor v rámci sítě. Jedná se totiž sice o kopii transakce, ale zároveň o transakci nově vytvořenou, která jako taková musí mít svůj identifikátor. Jelikož ale ze všech takto zkopírovaných transakcí se na konci pouze jedna dostane do stavu, kdy je sítí potvrzena, je třeba sledovat všechny.

Algoritmus kterým se kontrolují platby je následující: Systém zaznamenává všechny nové transakce v síti. Tyto transakce jako takové mohou, ale nemusí být potvrzené. Tuto informaci systém zatím nemá. Jakmile zaznamená transakci, zkontroluje, jestli je na adresu, na kterou mají být platby zasílány a jestli se jedná o příchozí transakci. Pokud ano, pak se z transakce získá zpráva, která slouží jako variabilní symbol. Podle této zprávy a částky transakce se v databázi plateb najde založená platba. Pokud je tato platba nalezena a je mladší než 15 minut, je posunuta do stavu, který říká, že uživatel již transakci vytvořil. Díky tomu již přestává platit limit na zaplacení do 15 minut, jelikož máme jistotu, že transakci již uživatel udělal. Tento algoritmus je vyobrazen na diagramu 4.5. Stále ale není jisté, zda je transakce potvrzena sítí. Jelikož je ale možné, že transakce bude muset být „reattachována“, může být do databáze uloženo více transakcí pro jednu platbu. Periodický proces poté každou minutu bere všechny transakce, které jsou v nepotvrzeném stavu a kontroluje, jestli již nejsou ověřeny. Pokud je nějaká tato transakce ověřena, je označena jako potvrzená a společně s ní jsou jako potvrzené označeny všechny transakce v databázi, které spadají pod stejnou platbu. Nakonec je platba označena jako zaplacená a uživateli je přiřazen zakoupený článek. Jelikož se může stát, že některé transakce nebudou potvrzeny nikdy, jelikož se například uživatel pokouší o takzvaný double-spending útok, který byl zmíněn v rešeršní části, je třeba tyto transakce promazávat, aby zbytečně nezabíraly místo a nezpomalovaly proces kontroly potvrzení transakcí. Toto promazání probíhá v testovací aplikaci jednou za 12 hodin, ale v běžném provozu by byla perioda nižší. Při tomto promazávání se zároveň všechny transakce, které jsou starší než 15 minut a ještě nejsou ve stavu čekajícím na ověření, označí jako neúspěšné.

Pro získávání nově vytvořených transakcí v síti je využívána decentralizovaná fronta ZeroMQ, která již byla zmiňována. Tato fronta obsahuje zprávy, které jsou do ní zasílány plným uzlem. Zprávy jsou rozděleny do jednotlivých témat. Například nově vytvořené transakce mají téma tx, nově potvrzené transakce téma sn a existuje ještě dalších přibližně 20 témat, které se mimo jiné týkají i informací o samotném uzlu. V rámci aplikace je třeba získávat informace o nově ověřených transakcích a aplikace je tedy přihlášena k odběru tématu tx. K připojení a naslouchání frontě je využívána oficiální ZeroMQ knihovna.



Obrázek 4.5: Algoritmus zpracování nové transakce v Tangle

Přihlášení k odběru zpráv z fronty je poté naprogramováno následujícím způsobem:

```
const zmq = require('zeromq');
let sock = zmq.socket('sub', {});

sock.connect(config.ZMQUrl);
sock.subscribe('tx');

sock.on('message', function (msg) {
    doSomethingWithMessage(msg);
});
```

V ukázce lze vidět inicializaci socketu, který se přihlašuje k odběru zpráv pomocí zmq.socket('sub', {}). Takto vytvořený socket je následně přihlášen k ZeroMQ pomocí tcp url v config.ZMQUrl a pomocí .subscribe() funkce je následně řečeno, že se socket přihlašuje k odběru témat typu tx. Následně už je pouze na socketu registrována funkce, která se spouští na základě každé události typu message. Každá tato událost následně vyvolá zaregistrovanou funkci, která s přijatou zprávou provede funkci doSomethingWithMessage(). Takto přijaté zprávy jsou textovým řetězcem s jednotlivými hodnotami oddělenými mezerou. Pokud je zpráva dle mezer rozdělena, je výsledné pole v následujícím formátu:

```
['tx',
'LATWBSVCBGWJQE9APKBKOYXANQTUGSUHLGKG9GYYKBQVRQLJVGFHFOWKSEXCJPK
DORFUYWNBM9QTMF999',
'999999999999999999999999999999999999999999999999999999999999999
999999999999999999',
'0',
'99999999999999999999999999',
'1546620546',
'1',
'1',
'EGRHWWFSKNRHQZRFBVNTHAHXSCXHGDXFKOUKQALNEYCNAWCKKRJUYYYZIAGRXIO
REHKZJXGOKPZFZ9WSC',
'CHOTVZZHEWGYM9INVOFHRVXRONSZURDGNQX9W9KSOCQA9XKNFZG9U9BKY9BOZSL
OXBZHCZVHCGSFUE999',
'CHOTVZZHEWGYM9INVOFHRVXRONSZURDGNQX9W9KSOCQA9XKNFZG9U9BKY9BOZSL
OXBZHCZVHCGSFUE999',
'1546620546676']
```

Na první pozici je téma do kterého zpráva patří, v tomto případě lze vidět, že se jedná o téma tx a tedy o nově založenou transakci. Na druhé pozici je hash transakce, tedy jednoznačný identifikátor v rámci sítě. Následuje adresa,

které se transakce týká. Další hodnotou je počet tokenů v transakci. Na ní navazuje tag transakce. Tag je další možností identifikátoru, který je volitelný uživatelem. Většina peněženek však neumožňuje tento tag zadat. Po tagu je časová známka, která říká, kdy byl vytvořen bundle, ve kterém se transakce nachází. Další dvě čísla říkají index této transakce v rámci bundle a celkový počet transakcí v bundle. Dalšími jsou hash samotného bundle, hash poslední transakce v bundle, hash první transakce v bundle a na konci je časová známka kdy byla transakce připojena k tangle.

Zpráva tedy obsahuje sadu důležitých informací o transakci, neobsahuje ovšem tu, která je pro nás jednou z těch nejdůležitějších a tou je zpráva připojená k transakci. Transakce je tedy nejdřív zvalidována, kde z adresy zjistíme, jestli se shoduje s adresou, na kterou aplikace přijímá platby a zda hodnota transakce je kladné nenulové číslo. To říká, že se jedná o transakci reprezentující příjem tokenů na naši adresu a tedy transakci, která nás zajímá. Pokud touto validací transakce projde, je předána službě checkPotentialTransaction, která tuto transakci vezme a ověří, jestli je možné ji spárovat s nějakou existující platbou v databázi.

K tomuto ověření je využíván modul @iota/core. Jedná se o knihovnu, která umožňuje jednodušším způsobem používat API plného uzlu. Jelikož přijatá zpráva neobsahuje zprávu transakce, je třeba získat z tangle samotný objekt transakce. K tomuto je využita knihovní funkce getTransactionObject, která bere pole transakčních hashů a nazpět vrací Promise, která do resolve funkce dává pole objektů nalezených transakcí. Tyto objekty již obsahují parametr message, který odpovídá zprávě transakce. Problémem ovšem je, že tato zpráva je zakódovaná v ternární soustavě a v databázi je uložena ve standardním ASCII formátu. Proto je využívána další knihovna @iota/converter, obsahující různé konverzní funkce vhodné pro práci s IOTA protokolem. Jednou z těchto funkcí je trytesToAscii(), která přeloží text z ternárního formátu do ASCII. Takto přeložená zpráva je následně vyhledána v databázi a pokud je nalezena platba s touto zprávou, je ověřeno, jestli obsahuje stejnou částku jako přišla v transakci a je mladší než 15 minut. Pokud ano, je platba označena jako čekající na potvrzení a transakce je uložena do databáze jako nepotvrzená transakce. V případě, že je platba již ve stavu čekajícím na zplacení, není kontrolováno to, jestli je mladší než 15 minut, jelikož tato transakce není první, která na tuto platbu byla vytvořena, ale jedná se o reattachovanou kopii, takže v databázi je už jiná transakce, která v síti byla včas. Tato nová transakce je pak uložena stejně jako předchozí.

V tuto chvíli jsou tedy v databázi platby ve stavu čekajícím na potvrzení a transakce, které čekají na potvrzení sítí. Je tedy třeba postupně ověřovat, zda nepotvrzené transakce nejsou již sítí potvrzeny. K tomu slouží služba checkTransactionConfirmed, která se spouští jednou za minutu. Ta z databáze vezme všechny transakce, které jsou v nepotvrzeném stavu a jejich hashe předá do knihovní funkce getLatestInclusion z @iota/core. Tato funkce bere pole hashů transakcí a nazpět vrací pole booleanů, které jsou ve stejném pořadí,

jako hashe, které do funkce vstoupily a říkají, jestli je transakce s daným hashem v posledním milestone braná jako potvrzená. Postupně se tedy projde tento výstup a pokud je transakce označena jako zaplacená, zavolá se funkce setTransactionAsConfirmed(), která v databázi tuto transakci označí jako potvrzenou. Stejně tak jsou jako potvrzené označeny všechny transakce, které mají stejnou vazbu na platbu, jako hlavní potvrzovaná transakce. Společně s tím je označena platba jako zaplacená a uživateli je přiřazen článek, který tato platba obsahuje.

Jelikož je vždy nutné ověřit potvrzenost všech ještě nepotvrzených transakcí, mohlo by se stát, že by tento seznam narostl do velkých rozměrů například v situaci, kdy by byly vytvářeny nevalidní transakce, které by nikdy potvrzeny nebyly. Z tohoto důvodu je vytvořena služba failOldPayments, která v periodě bere platby, které jsou starší než 24 hodin, nastaví je jako neúspěšné a zároveň z databáze smaže všechny transakce, které na tuto platbu byly vytvořeny.

4.3.10 Zhodnocení použitelnosti Tangle a IOTA knihoven k aplikačnímu vývoji

Při vývoji systému jsem postupně narazil na několik problémů. Prvním je samotná podstata, jak funguje Tangle a transakce na něm, které vyžadují specifický přístup při vývoji, který byl prezentován v kapitole o zpracování transakcí. Je třeba při vývoji počítat s volatilitou tokenu a s tím, že transakce, které jsou v síti vůbec nemusí být v budoucnu potvrzené. Zároveň je třeba řešit to, že jedna identická transakce může být ve skutečnosti v síti několikrát, avšak pouze jedna z kopií bude v budoucnu potvrzena. Toto vše klade specifické nároky na návrh systému a jednotlivých algoritmů.

Další otázkou, která stále není úplně jasná, je náročnost na infrastrukturu. Při spuštění aplikace na živé síti bylo třeba zpracovávat přibližně 300 nových transakcí na síti za sekundu. Pravda je, že tyto transakce jsou většinou profiltrovány tím, že se týkají jiných adres, než aplikační, ovšem v budoucnu se počítá s tím, že by sítí protékaly miliony transakcí za sekundu a je otázkou, jak budou systémy takový tok schopné sledovat. Při 300 transakcích za sekundu neměla aplikace se zpracováním problém. Běžné dotazy na výpis článků, trvaly pouze kolem 150 ms.

Předchozí odstavec a předchozí kapitola mluví o zpracovávání všech nových transakcí na síti. Toto ovšem není teoreticky třeba. Dle dokumentace by ZeroMQ měla být schopna prezentovat transakce týkající se pouze jedné adresy a to tak, že se jako téma ke kterému se socket přihlašuje, uvede právě tato adresa. Toto ovšem v praxi nefunguje. Dokumentace je velmi strohá a neuvádí nic více než pouze informaci o tom, že takováto možnost volby tématu zpráv existuje. Stejně tak je internet velmi strohý na informace o vývoji nad IOTA protokolem. Ve snaze tento problém vyřešit jsem se dotazoval na IOTA Stack Exchange a zároveň i na oficiálním IOTA Discord serveru, ovšem obě moje

otázky zůstaly bez jediné odpovědi. Při prohledávání zdrojových kódů a snaze najít část, kde se generují zprávy pro jednotlivé adresy, jsem žádný takový kód nenašel a je tedy otázkou, zda vůbec taková funkcionalita, která je v dokumentaci zmíněna, existuje. Toto je velmi nepříjemné, protože kdyby bylo možné poslouchat pouze určitou adresu, pak by se z nutnosti zpracovávat 300 transakcí za sekundu mohlo stát například pouze několik jednotek transakcí za minutu.

Kromě nedostatečné, až téměř neexistující dokumentace IOTA knihoven, je problémem i jejich občasné divné chování, na které jsem při vývoji narazil. Jednou ukázkou jsou například neočekávané způsoby volání funkcí. Například funkce getTransactionObject() by dle očekávání brala identifikátor transakce a vracela její objekt. Ovšem tato transakce bere pouze pole identifikátorů a stejně tak vrací pole objektů. Toto je dle zvoleného názvosloví poněkud zvláštní. Jedná se ovšem pouze o marginální chybu. Problém, který vidím jako zásadnější je ve funkci trytesToAscii(), která byla zmiňována dříve. Jedná se o funkci, která překládá ternárně zakódovaný text do ASCII formátu. Jelikož na jeden ASCII znak se překládají 2 tryty, funkce kontroluje, jestli má vkládaný řetězec sudý počet znaků. Pokud ne, funkce vrací chybu. Problémem však je, že všechny zprávy v transakcích mají vždy 2187 znaků, nehledě na to, jak jsou dlouhé. Pokud jsou krátké, jsou jednoduše doplněny znakem 9, opakovaným tolikrát, aby celá délka zprávy byla 2187 znaků. Oficiální funkce od IOTA foundation tedy není kompatibilní s formátem zprávy, který byl nadefinován stejnou institucí. Jedinou možností jak zprávu přeložit je tedy vždy odstranit poslední znak, čímž se počet znaků upraví na sudé číslo.

Celkově je však pravdou, že řešení je funkční. Ačkoliv vývoj jako takový není jednoduchý a často ani příjemný, jelikož se během něj vývojář často potýká s nevědomostí a nedostatkem veřejně dostupných informací, výsledné řešení funguje a svůj účel splňuje. Dá se tedy říci, že řešení je použitelné. Stále je třeba brát v potaz fakt, že se jedná o velmi mladou technologii, která není dostatečně adoptovaná, nemá dostatečnou podporu a s tím jsou tedy spojeny všechny nedostatky dokumentace, počtu zdrojů a dalších věcí, které ztěžují vývoj. Je zároveň vidět, že IOTA foundation a komunita, která se okolo technologie točí, aktivně pracují na dalším vývoji, opravách chyb a zevrubném testování. Dá se tedy počítat, že jednoduchost vývoje se bude zlepšovat. Vznikají zároveň portály, které edukují veřejnost a vývojáře

V konečném zhodnocení tedy technologie splňuje požadavky, které na ní byly při vývoji kladeny. Je možné systém tohoto typu zprovoznit a provozovat, ačkoliv samotný vývoj je složitější než při využití tradičních technologií.

4.4 Zhodnocení a ověření řešení

V rámci implementace vzniklo systémové rozhraní, které obsahuje aplikační logiku umožňující napojení frontendové aplikace pokrývající všechny případy

užití obsažené v analytické části implementace. Vytvořený systém má drobné odlišnosti od samotné implementace, například v názvosloví databáze, které vychází z volby technologií. Větší odlišností je forma komunikace mezi uzlem a aplikací, která byla původně zamýšlena pouze jednosměrně a to tak, že aplikace bude odebírat data ze ZeroMQ fronty. Tato představa vznikla z nedostatku dokumentace týkající se fronty a při implementaci následně vyšlo najevo, že fronta neobsahuje všechna potřebná data a je tedy třeba i aktivně komunikovat s API uzlu.

Jelikož bylo vytvářeno pouze API, probíhalo testování především na úrovni funkčnosti všech vytvořených koncových bodů. Byly tedy prováděny především manuální integrační testy. Jelikož se jednalo především o proof-of-concept aplikaci, bylo důležité především otestovat technologii, která byla předmětem ověřovaného konceptu. V tomto případě tedy hlavně IOTA technologii. Zde bylo testováno několik domén. První byla samotná funkčnost. V rámci testování bylo ověřeno, že technologie funguje, tedy že poslané transakce jsou na síti zaznamenány a chovají se tak, jak mají. Systém je následně bezchybně zpracuje a celkový funkční proces je tak v pořádku. Další doménou, která byla třeba ověřit, byla rychlost. Zde byla aplikace testována na hlavní síti, kterou v době testů proudilo přibližně 250 transakcí za sekundu. Vzhledem k nefunkčnosti sledování transakcí pouze na specifické adrese tedy systém musel každou sekundu prověřit 250 transakcí. Toto kladlo určité nároky na výpočetní kapacitu, které se mohly nepříjemně podepsat na výkonu celého systému. Jelikož je ale velmi rychle transakce profiltrována, není její zpracování problémem a aplikace tedy stroj přílišně nezatěžuje. Test byl prováděn pomocí Apache Benchmark, kdy byl spuštěn požadavek na výpis článků a to s 350 konkurenčními připojeními a celkovým počtem 20 000 požadavků. Jelikož Node.js běží na jednom vláknu, bylo při této zátěži vytěžováno jedno vlákno prakticky na 100%. I přesto ale pouze na této jedné instanci byla stále průměrná doba odpovědi 600 ms, tedy přijatelná na poměrně velkou zátěž. Poslední doménou byla dostupnost. Jelikož k běhu systému je kromě samotné běžící instance třeba ještě běžící instance plného uzlu a napojení na ZeroMQ frontu, je mnoho služeb, které musí být zároveň funkční, aby systém správně fungoval. Z tohoto důvodu bylo testováno, jestli systém může běžet delší dobu v kuse bez problémů. Takto byl systém ponechán 3 dny, během kterých nedošlo k pádu, ani k omezení funkčnosti.

Samotné řešení celkově naplnilo předpokládaná očekávání, která s přihlédnutím k aktuálnímu nevyspělému stavu sítě, byla poměrně skromná. Hlavní podstata, kterou je možnost mikroplateb, je plně funkční. Problémem je však fakt, že aktuálně je platební model spíše na úrovni o mnoho rychlejšího bankovního převodu, než na úrovni platby kartou. Toto je spojeno s aktuálním stavem sítě a pokud se Tangle posune tak, jak plánuje, pak by platby měly být otázkou několika málo sekund, i méně. Už nyní je však možné platit opravdu malé částky, což je vzhledem k aktuálně dostupným možnostem, které prakticky neumožňují online zaplatit menší částku, než přibližně 3 Koruny české,

velmi dobrou funkcí. S drobnými úpravami by zároveň systém mohl nabízet kromě článků i média. Aktuální model, kdy je cena článku vypočítávána na základě počtu znaků, je modelem, který umožňuje s odhady zisku pracovat na úrovni, kdy máme přímý kontrast mezi cenou za normostranu, kterou zaplatíme autorovi článku a cenou za normostranu, kterou zaplatí čtenář. V rámci toho můžeme díky odhadům čtenosti odhadovat ziskovost jednoho specifického článku. Pro tuto aplikaci, která slouží primárně jako proof-of-koncept, se jedná o dostačující model. Je však pravděpodobné, že kdyby se měl systém začít používat v praxi, bylo by třeba přejít od tohoto modelu, který pracuje čistě s počtem znaků na model, který by umožňoval jednotlivé články naceňovat individuálně, kvůli tomu, že ne každá normostrana má ve výsledku pro čtenáře stejnou hodnotu. Hlavním za co lidé platí, je obsah a ne počet stran. Z ekonomického hlediska je tedy systém velmi dobrým z pohledu toho, jak malé částky umožňuje přijímat bez ztráty velkých procent na poplatcích. Jsou zde ovšem prostory pro zlepšení systému, především v oblasti práce s cenou článku.

4.5 Shrnutí kapitoly

V rámci implementační části byl navržen systém umožňující nákup jednotlivých článků z článkového webu s použitím IOTA tokenů a sítě pro platbu. Jeho hlavním účelem bylo ověření, zda lze na IOTA technologii postavit funkční aplikaci. Ze samotného systému bylo implementováno systémové rozhraní a aplikační logika, která pokrývá hlavní případy užití navrhovaného systému. Při implementaci byly využity moderní technologie v čele s Node.js a knihovnami pro komunikaci s IOTA sítí. Zároveň bylo nutné nainstalovat a zprovoznit plný IOTA uzel, který je pro komunikaci s Tangle využíván.

Implementace samotná byla poměrně náročná, především z důvodů nedostatečné dokumentace a podpory oficiálních IOTA knihoven. Vyskytly se části, které nefungovaly, nebo fungovaly jinak, než by se mohlo na první pohled zdát. Samotná technologie však funguje, což bylo v rámci testování prokázáno. Systém je poměrně rychlý i pouze na jedné instanci a platby pomocí tokenů je možné úspěšně provádět. Dá se tedy říci, že bylo ověřeno, že technologie jako taková funguje, ačkoliv její použití je poměrně složité a musí urazit ještě dlouho cestu, aby se přiblížila jednoduchosti tradičních technologií. Samozřejmě je třeba brát v potaz, že do budoucna by možná určitá složitost navíc oproti standardu měla být vyvážena nadstandardními vlastnostmi, jako jsou vyšší bezpečnost, rychlost nebo dostupnost.

Kapitola 5

Doporučení dalšího postupu

5.1 Implementovaný systém

Ačkoliv aplikace nebyla zamýšlena jako produkční verze, ale jako proof-of-concept nové technologie, je možné ji dále využívat jako základ pro její rozšíření. V rámci tohoto jsou věci, které by bylo dobré při dalším postupu udělat.

Samotná architektura aplikace by mohla být lépe strukturovaná. Pokud by se měl systém dále rozrůstat, mohlo by se stát, že se narazí na limity navržené architektury. Ta byla plánována především pro aplikaci podobného rozsahu, kterého je teď.

V rámci efektivity by bylo dobré rozdělit systém na dvě nezávislé aplikace. První, která se bude starat o práci s články, uživateli a dalšími zdroji a bude sloužit především jako předkladač dat. Druhá by potom měla být tím, co se stará o komunikaci s Tangle a kontroluje a zaznamenává transakce. Důvodem pro toto je především efektivita. Jelikož je třeba zpracovávat velké množství transakcí najednou a Node.js je jednovláknový, mohlo by se stát, že by tyto zpracování transakcí zpomalovaly čas odpovědi na uživatelovo volání. Rozdělení by zároveň ulehčilo škálování, jelikož by se každá aplikace dala škálovat zvlášť.

Systém je aktuálně pouze API a je tedy pro běžného uživatele prakticky nepoužitelný. Měla by tedy případně vzniknout frontendová aplikace, která umožní uživateli systém využívat pomocí grafického rozhraní. Za tímto účelem by bylo dobré vytvořit i dokumentaci k API, které je sice v tuto chvíli velmi jednoduché, ale v budoucnu by se mohlo rozrůstat.

Poslední menší částí je mailingová služba, která není v tuto chvíli využívána a uživatel tedy není informován o stavu platby. Jedná se o funkci naprosto marginální pro ověření konceptu technologie IOTA, avšak z uživatelského hlediska naopak o velmi důležitou. Pokud by tedy měla být aplikace v praxi používána, je třeba odesílání e-mailových notifikací dodělat.

Posledním, co by se v případě dalšího rozvoje systému mělo změnit je model pro výpočet ceny článku. Jak již bylo řečeno ve zhodnocení implementace,

model je dostačující pro tuto testovací aplikaci, především proto, že dobře simuluje výpočet ceny, který by byl využíván například při odběru energií, tedy velmi malá částka za každou jednu přenesenou jednotku. V praxi by ale pro takovýto článkový web bylo lepší přejít na model, který umožňuje cenu článku přímo určit, případně pro každý článek nastavit jinou cenu za normostranu, jelikož různé články mají díky obsahu různou hodnotu pro čtenáře, v důsledku čehož jsou ochotni zaplatit více či méně.

5.2 IOTA jako projekt

V rámci samotného IOTA projektu osobně vidím několik nedostatků, které mohou zpomalovat postupný rozvoj technologie a ekosystému. Věřím, že všechny tyto problémy si IOTA Foundation uvědomuje a že se jimi aktivně zabývá. Je ale dobré je zde v naprostém závěru práce připomenout.

Prvním poměrně zásadním problémem, který vnímám v rámci velké části projektu je jeho nedostatečná průhlednost. Ačkoliv je většina zdrojových kódů open-source, vnímám mezery v komunikaci směrem k veřejnosti. Asi nejvíce je toto cítit v komunikaci týkající se aktuálního stavu a další vývoje projektu. Roadmapy jsou velmi strohé, neobsahují aktuální stav komponent a už vůbec neobsahují časové odhady, kdy by mohla být daná část dokončena. Chápu, že IOTA Foundation je velmi opatrná, co se všech časových odhadů týče, kvůli předchozí špatné zkušenosti, kdy prezentovali veřejnosti pevný čas, kdy bude vydána nová webová stránka IOTA projektu a tento čas se následně značně nedodržel, což vedlo k poměrně velké kritice celého projektu. V takovém případě by ale mělo docházet k lepším časových odhadům, nebo neudávání tak přesných časů, ale alespoň nějakého obecného odhadu. Vhodné by bylo dělat například měsíční zprávu o vývoji projektu na blog. Pravdou je, že existuje newsletter, který vychází každý měsíc, avšak informací o samotném postupu projektu obsahuje většinou velmi málo.

Další důležitou věcí jsou dokumentace, výukové portály a celkově podpora vývoje mimo IOTA Foundation. Vzhledem k tomu, jak důležitá je pro projekt adopce technologie, je zásadní to, aby se do vývoje zapojovala i samotná komunita a začalo vznikat více komunitních projektů. Občas je vidět nějaký základ konceptu, avšak většinou nepříliš podporovaný. Samotná IOTA Foundation stále propaguje svůj finanční fond z kterého chce podporovat vznikající projekty, avšak zatím jsem se nedočetl o žádném, který by dotaci dostal. Zároveň je vidět, že organizace se zaměřuje primárně na vyhledávání spřáteleným společností především z oblasti velkých technologických firem. Tyto firmy však budou s největší pravděpodobností využívat svou privátní kopii tangle a nebudou tedy přispívat přímo k fungování hlavní sítě. Jedná se tedy o krok, který může projekt částečně zviditelnit, případně mu dodat lepší důvěryhodnost, avšak je třeba zároveň podporovat růst hlavní sítě a to především tím, že je třeba zajistit, aby byla síť využívána aplikacemi a lidmi.

Proto je třeba zlepšit možnost vývoje nad protokolem a zároveň dále zlepšovat použitelnost sítě běžným uživatelem.

Závěr

Práce měla několik hlavních cílů, postupně rozdělených mezi rešeršní, analytickou a implementační část.

V rešeršní části je čtenáři představen koncept Internetu věcí a především prezentována jeho definice, z které tato práce dále vychází. Zároveň jsou rozebrány požadavky, které IoT klade na technologie, které by v rámci něj měly být využity. Společně s tím je čtenáři představen projekt IOTA a vysvětlen koncept na kterém je založen, společně se všemi hlavními teoriemi a algoritmy, díky kterým funguje.

Analytická část nejprve probírá fungování IOTA technologie v rámci jednotlivých požadavků Internetu věcí, které vychází z rešerše. Na základě této analýzy je zhodnocena celková použitelnost IOTA projektu. V další části jsou prezentovány některé nové obchodní modely a aplikace, které díky IOTA mohou vzniknout.

Třetí, implementační část se celá zabývá vývojem pay-per-view systému pro článkové weby, který umožňuje platby pomocí protokolu IOTA a vznikl jako proof-of-concept analyzované technologie. Systém je nejprve zanalyzován a navržen a následně je implementováno API v technologii Node.js s použitím oficiálních knihoven projektu IOTA. V rámci kapitoly jsou analyzovány i samotné možnosti a problémy vývoje nad IOTA protokolem. Na konci je vzniklá implementace otestována, zhodnocena a s tím zhodnoceny i samotné možnosti vývoje s využitím IOTA technologie.

V poslední části je pak doporučen další postup v případě potenciálního budoucího rozvoje implementovaného systému. Kromě toho jsou zároveň prezentovány určité problémové domény IOTA projektu a doporučení, co by v rámci projektu mělo proběhnout, případně na co by se projekt měl více zaměřit.

V rámci práce jsem naplnil cíle, které v ní byly vytyčeny. Nejpřínosnějším cílem osobně vidím představení a analýzu nové, moderní a zatím poměrně neznámé technologie, která si klade za cíl stát se standardem v rámci Internetu věcí. Implementací jsem zároveň ověřil, že již teď je technologie použitelná a je možné na ní postavit funkční aplikaci, ačkoliv je stále mnoho problémů, které

je třeba v rámci projektu vyřešit.

Literatura

[1] Internet Society: Brief History of the Internet—Internet Society. [online], [cit. 2018-09-05]. Dostupné z: https://www.internetsociety.org/wp-content/uploads/2017/09/ISOC-History-of-the-Internet_1997.pdf

[2] Stanford: How Does the Internet Work? 2002, [online], [cit. 2018-09-09]. Dostupné z: https://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm

[3] Minerva, R.; Biru, A.; Rotondi, D.: Towards a definition of the Internet of Things (IoT). [online], [cit. 2018-09-21]. Dostupné z: https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf

[4] IEEE: Internet of Things Requirements and Protocols — IEEE Standards University. [online], [cit. 2018-09-21]. Dostupné z: https://www.standardsuniversity.org/e-magazine/march-2016/internet-of-things-requirements-and-protocols/

[5] Walport, M.: Distributed Ledger Technology: beyond block chain. [online], [cit. 2018-09-22]. Dostupné z: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf

[6] Weisstein, E.: Directed Graph. From MathWorld–A Wolfram Web Resource. [online], [cit. 2018-09-30]. Dostupné z: http://mathworld.wolfram.com/DirectedGraph.html

[7] Serguei, P.: The Tangle, Version 1.4.3. [online], [cit. 2018-09-30]. Dostupné z: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf

[8] IOTA Support: IOTA Support - Tutorial - Wallet Knowledge Base. [online], [cit. 2018-09-30]. Dostupné z: https://iotasupport.com/walletknowledgebase.shtml

[9] Massachusetts Institute of Technology: Random Walks. [online], [cit. 2018-10-14]. Dostupné z: https://www.mit.edu/~kardar/teaching/projects/chemotaxis(AndreaSchmidt)/random.htm

[10] Gal, A.: The Tangle: an illustrated introduction - IOTA. [online], [cit. 2018-10-14]. Dostupné z: https://blog.iota.org/the-tangle-an-illustrated-introduction-f359b8b2ec80

[11] Investopedia: Proof of Work Definition—Investopedia. [online], [cit. 2018-10-16]. Dostupné z: https://www.investopedia.com/terms/p/proof-work.asp

[12] Back, A.: Hashcash FAQ. [online], [cit. 2018-10-16]. Dostupné z: http://www.hashcash.org/faq/

[13] Narula, N.: tangled-curl/vuln-iota.md at master . mit-dci/tangled-curl . GitHub. [online], [cit. 2018-10-16]. Dostupné z: https://github.com/mit-dci/tangled-curl/blob/master/vuln-iota.md

[14] IOTA Foundation: FAQs — IOTA. [online], [cit. 2018-10-16]. Dostupné z: https://www.iota.org/get-started/faqs

[15] Gal, A.: The Tangle: an illustrated introduction - IOTA. [online], [cit. 2018-10-28]. Dostupné z: https://blog.iota.org/the-tangle-an-illustrated-introduction-79f537b0a455

[16] IOTA Foundation: A note on Trinary — IOTA Guide. [online], [cit. 2018-10-28]. Dostupné z: https://domschiener.gitbooks.io/iota-guide/content/chapter1/a-note-on-trinary.html

[17] Cambridge University Press: TOKEN MONEY — meaning in the Cambridge English Dictionary. [online], [cit. 2018-11-02]. Dostupné z: https://dictionary.cambridge.org/dictionary/english/token-money

[18] Kehrer, P.: Asymmetric algorithms - Cryptography 2.4.dev1 documentation. [online], [cit. 2018-11-04]. Dostupné z: https://cryptography.io/en/latest/hazmat/primitives/asymmetric/

[19] Zerynth: Keccak secure hash - Zerynth Docs documentation. [online], [cit. 2018-11-04]. Dostupné z: https://docs.zerynth.com/latest/official/core.zerynth.stdlib/docs/official_core.zerynth.stdlib_crypto_hash_keccak.html

[20] IOTA Support: IOTA Support - How adresses are used in IOTA. [online], [cit. 2018-11-04]. Dostupné z: https://iotasupport.com/how-addresses-are-used-in-IOTA.shtml

[21] IOTA Foundation: Qubic: Quorum-based Computations - Powered by IOTA. [online], [cit. 2018-11-08]. Dostupné z: https://qubic.iota.org/intro

[22] IOTA Foundation: Qubic: Quorum-based Computations - Powered by IOTA. [online], [cit. 2018-11-08]. Dostupné z: https://qubic.iota.org/qubics

[23] Sønstebø, D.: IOTA Foundation - IOTA. [online], [cit. 2018-11-06]. Dostupné z: https://blog.iota.org/iota-foundation-fb61937c9a7e

[24] Konfidio: IOTA Report: Decoding the Tangle Part (2/4) – Konfid.io Blockchain Venture Studio – Medium. [online], [cit. 2018-11-06]. Dostupné z: https://medium.com/konfid-io-blockchain-reports/iota-report-decoding-the-tangle-part-2-the-known-and-unknown-history-of-the-iota-project-3db9f5f3cfa3

[25] Sønstebø, D.: IOTA Development Roadmap - IOTA. [online], [cit. 2018-11-08]. Dostupné z: https://blog.iota.org/iota-development-roadmap-74741f37ed01

[26] IOTA Foundation: Research & Development Roadmap — IOTA. [online], [cit. 2018-11-08]. Dostupné z: https://www.iota.org/research/roadmap

[27] IOTA Foundation: Qubic: Quorum-based Computations - Powered by IOTA. [online], [cit. 2018-11-08]. Dostupné z: https://qubic.iota.org/roadmap

[28] Bramas, Q.: The Stability and the Security of the Tangle. [online], [cit. 2018-11-09]. Dostupné z: https://hal.archives-ouvertes.fr/hal-01716111v2/document

[29] Merkle, R. C.: A Digital Signature Based on a Conventional Encryption Function. *CRYPTO '87 A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, 8 1987: s. 369–378.

[30] Handy, P.: Introducing Masked Authenticated Messaging - IOTA. [online], [cit. 2018-11-09]. Dostupné z: https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e

[31] IOTA Foundation: Qubic: Quorum-based Computations - Powered by IOTA. [online], [cit. 2018-11-09]. Dostupné z: https://qubic.iota.org/oracles

[32] Digiconomist: Bitcoin Energy Consumption Index - Digiconomist. [online], [cit. 2018-11-10]. Dostupné z: https://digiconomist.net/bitcoin-energy-consumption

[33] IOTA Foundation: Coordinator. Part 3: Approaches to Coordicide – IOTA. [online], [cit. 2018-11-20]. Dostupné z: https://blog.iota.org/coordinator-part-3-approaches-to-coordicide-583fb82382bc

[34] Ethereum Foundation: Ethereum Project. [online], [cit. 2018-01-03]. Dostupné z: https://ethereum.org/

[35] Hospodářské noviny: Varianty předplatného — E-shop předplatného vydavatelství Economia a.s. [online], [cit. 2018-11-21]. Dostupné z: https://predplatne.ihned.cz/hn-plus

[36] E15: E15 Premium — E15.cz. [online], [cit. 2018-11-21]. Dostupné z: https://www.e15.cz/e15-premium

[37] GoPay: Kolik stojí platební brána — GoPay. [online], [cit. 2018-11-21]. Dostupné z: https://www.gopay.com/cs/cenik.html

[38] ZeroMQ: Learn the Basics - zeromq. [online], [cit. 2018-11-21]. Dostupné z: http://zeromq.org/intro:read-the-manual

[39] Cockburn, A.: *Writing Effective Use Cases*. Addison-Wesley Professional, 2000, ISBN 9780201702255.

[40] Fowler, M.: UseCases. [online], [cit. 2018-11-23]. Dostupné z: https://martinfowler.com/bliki/UseCase.html

[41] CoinMarketCap: CoinMarketCap API Documentation. [online], [cit. 2018-11-24]. Dostupné z: https://coinmarketcap.com/api/documentation/v1/

[42] GraphQL Foundation: GraphQL — A query language for your API. [online], [cit. 2018-11-25]. Dostupné z: https://graphql.org/

[43] Birrell, A. D.; Nelson, B. J.: Implementing remote procedure calls. *ACM Transactions on Computer Systems (TOCS)*, ročník 2, 2 1984: s. 39–59.

[44] Fielding, R. T.: *Architectural Styles and the Design of Network-based Software Architectures*. Dizertační práce, University of California, Irvine, 2000.

[45] IOTA Foundation: GitHub - iotaledger/iri: IOTA Reference Implementation. 2018, [online], [cit. 2018-11-26]. Dostupné z: https://github.com/iotaledger/iri

[46] Semko, V.: SemkoDev / nelson.cli · GitLab. 2018, [online], [cit. 2018-11-26]. Dostupné z: https://gitlab.com/semkodev/nelson.cli

[47] mongoose-paginate: mongoose-paginate-npm. 2018, [online], [cit. 2018-01-02]. Dostupné z: https://www.npmjs.com/package/mongoose-paginate

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

ICO Initial Coin Offering

IEEE Institute of Electrical and Electronics Engineers

IoT Internet of Things

JSON JavaScript Object Notation

M2M Machine to Machine

MAM Masked Authentication Messaging

MIT Massachusetts Institute of Technology

POW Proof-of-Work

REST Representational state transfer

SMTP Simple Mail Transfer Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

UCLA University of California, Los Angeles

WOTS Winternitz One-Time Signature

XML Extensible markup language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src.zip` ..... zdrojové kódy implementace
- `install.txt` ..... návod na zprovoznění aplikace ze zdrojových souborů
- `thesis` ..... složka obsahující pdf práce a zdrojové soubory k práci
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `src` ..... složka obsahující zdrojové soubory práce
    - `thesis.tex` ..... práce ve formátu LaTeX