Accountant pattern: Lightweight solution for embeded micropayments

Jaro Šatkevič, me@jaro.lt

June 18, 2019

With rise of attempts to build decentralised platforms on internet with embeded payment mechanisms into them, the need of high throughput micropayments solution using crypto currencies is big as never before. Blockchain payments must be verified and stored by every node in the network, meaning that the node with the least resources limits the overall throughput of the system as a whole. Popular off-chain protocols, such as *Lightning Network*, *Raiden* or various *Plasma* implementaitons, may provide solution micropayments using cryptocurrencies, however they have own problems while using for high throughput micropayments in decentralised systems.

In this paper is provided construction of embedable, lightweight micropayment protocol called *"Accountant pattern"* which is essentially solution combining techniques of payment hubs and digital cheque based uni-directional channels. Such construct may be used in decentralised VPN, CDN or Video streaming platforms and provides hundreds of thousands transactions per second throughput while in same time is relatively easy to implement.

Contents

1 Introduction

Idea of building decentralised systems is not new but the main addition to the previous ideas about decentralised protocols is possibility to embed payment mechanisms into them.

In decentralised systems, there are quite a few situations when two parties that don't have trust relationship between each other may want to perform mutual transactions. This is especially true in decentralised VPN network case.

Because all network participants can be anonymous there could be situations when service *consumer* will not be willing to pay up-front whole amount and service *provider* will not be willing to deliver service without prepayment. In such situations service could be split into chunks provided in exchange for micropayments, so each party could potentially risk only tiny amount of funds. In such situation there may be a need of transacting between parties couple times a minute while sending tiny amounts (value of less than 1 cent).

Thanks to cryptocurrencies now it's possible to embed decentralised payment solution in trustless and permissionless way. Because of technical limitations some additional work have to be done however.

The backbone of cryptocurrencies is a technology called *blockchain*. This technology requires each transaction to be stored on the ledger and replicate it over thousands of the network participants. This impose a fundamental limit of amount of transactions can be processed. Let's take two most popular blockchains Bitcoin and Ethereum. Their blockchains can provide throughput of 3 to 25 transactions per second which causes high transaction fees and makes on-chain transactions too expensive for micropayments.

To overcome blockchain scalability problems several approaches have been proposed:

- ***Speedup blockchain*** itself by changing some fundamental components.
- Use ***blockchain per project*** (sidechains, Cosmos and Polkadot networks).
- ***Plasma*** and other kinds of child-chain solutions.
- Payment and ***state channels*** based solutions.

Most of proposed solutions are still in early stage of development and are not widely adopted. Each of them have their own pros and cons and there is no clear winner at the moment. This is one of reasons why most of modern decentralised protocols are still in early betas. Some have already working core solutions, however payments are usually the missing part.

The goal of this work is to research simplest possible but still cheap and powerfull solution for high throughput micro payments using utility tokens and easy embedable into decentralised VPN, CDN or streaming services.

Following tasks have to be accomplished to fulfill the goal of this work:

1. Analyse needs of decentralised VPN, CDN or streaming platform and collect requirements for payment solution which could be used there;

2. Do research of several most promissing 2nd layer solutions such as state channels, micropayment channel Networks, Plasma and Sidechains. Look over possibility to use them as base for high throughput micropayments in decentralised platforms, describe their advantages, limitations and side effects.

3. Provide lightweight, easily embedable solution for high throughput micropayments.

2 Platfrom review and requirements for payment solution

Let's take Mysterium – decentralised VPN platform as example where proposed micropayments solution can be used.

Mysterium Network is a network of nodes providing security and privacy to Mysterium end users via dVPN application. Combining powerful encryption, reputation mechanisms and layered protection protocols team ambition is to build an infinitely scalable P2P architecture. The main difference of dVPN comparing to other VPN service providers in the market is their architecture with decentralisation at the core. There should be no sigle failing part in the network. Any government, company or person should not be able to censure or stop this service. Mysterium team itself should be not able to censure or stop work of network, after final solution will be released into production.

At the moment product itself is in beta testing and is missing scalable payments solution to go into public.

2.1 Components of the network

- **Provider nodes:** service written in golang which can be run by any user of network to provide their bandwidth for other users (called consumers) using OpenVPN in exchange to earning MYST tokens.

- **Consumer app:** software written in golang and JavaScript which can be run by any user in the network to connect to *provider nodes* via OpenVPN and use their bandwidth.

- **Discovery service:** decentralised service which helps providers to advertise themself in network so consumers would find them and use their bandwidth.

- **Myst token:** internal crypto currency used in the network and main and the only method of peyments. Myst token is released as ERC20 token on Ethereum blockchain.

- **Identity:** each user of the network (both, consumers and providers) have to register own identity, which is similar to ethereum address (20 bytes of keccak256 hash of public key derived from private key using eliptic curve cryptography). It is needed to uniquely identify network actors and to be able securly establish encripted connection between them. Identity have to be registered on-chain using smart-contract deployed into Ethereum network.

2.2 Initially proposed payment solution

Initially Mysterium team was planning to use mechanism which remotely resembles the way cheques works. A blockchain account holder can write a cryptographic cheque to another account (the beneficiary) as a form of payment. A cheque has the issuer's address, beneficiary's address, sum of tokens promised, sequence and signatures. Amount written on the cheque can be updated and only the last version of cheque is valid. Later beneficiary can settle promissed amount on blockchain.

Here is snippet from smart contract code which should settle promised value on-chain.

```
function settlePromise(address issuer, address beneficiary,
                       uint256 seq, uint256 amount,
                       bytes issuerSignature,
                       bytes beneficiarySignature) public {
    bytes32 promiseHash = keccak256(beneficiary, seq , amount);

    address recoveredIssuer = ecrecover(promiseHash, issuerSignature);
```

```
        require(recoveredIssuer == issuer);

        address recoveredBeneficiary = ecrecover(promiseHash, beneficiarySignature);
        require(recoveredBeneficiary == beneficiary);

        require(seq > clearedPromises[issuer][beneficiary]);
        clearedPromises[sender][receiver] = seq;

        require(token.balanceOf(issuer) >= amount);
        token.transferFrom(issuer, beneficiary, amount);

        emit PromiseSettled(issuer, beneficiary, seq, amount);
    }
```

This solution is much better than doing on-chain transactions. Cheques with promised amounts can be send from consumer to provider even each second and be send in peer-to-peer manner. Only *consumer* and *provider* will be aware of their existance and only once in a while *provider* will have to settle transactions into blockchain. This helps to reduce on-chain transactions amount a lot and still having quite secure way of providing service for untrusted party which can disapear at any moment.

2.2.1 Problems of digital cheques

There are couple of critical problems however, and they have to be fixed. First of all there may be situation when there are not enought funds on issuer's balance to cover promised value in the blockchain so settling transaction will be rejected. This creates *double spending* possibility, when after issuing cheque and until cheque is settled into blockchain, consumer still can try issue cheque of same funds to another party or even himself. In case of VPN service this is not that big problem because bandwidth is *"perishable product"* and service providers can afford having part of it unpaid. Howewer such risk of double spending possibility pushes to more often on-chain settlements (each time when promised amount is big enough so risk of loosing money is not acceptable anymore).

Second problem is that most of consumers will use providers services only once and only for limited time (e.g. one hour of listening music), so providers will get a lot of small value cheques (e.g. 0.10 EUR value in tokens). On Ethereum to settle one such cheque can cost from 0.01 till 1 EUR depending on network busyness. This means that either big fees will be paid or cheques will be never settled.

2.3 Requirements for "good" payments solution

Because of high level of privacy and anonymity in decentralised networks, actors of the network may don't trust each other and there is no trusted intermediary available which could resolve conflicts or act as custodian service. In such situations *Consumer* will be not willing to pay up-front whole amount and service *Provider* will be not willing deliver service without prepayment. In such situation service could be split into microservices and be provided in exchange for nanopayments, so each party could potentially risk only tiny amount of funds.

This leads into using *pay-as-you-go* payment model where consumers are paying constantly, right after receiving agreed part of service. There may be a need of transacting between parties couple times a minute while sending tiny amounts.

1. **requirement: consumer to provider payments**
   Usually there are two actors in the network, *Consumers* and *Providers* of the service. All payments are done by *Consumers* and received by *Providers* and never vice versa.

   There may be more complicated services where there can be more than two parties. E.g. Video streamer (person who is creating content), nodes which are streaming video and video viewers. In this case we still can separate all of them into *Consumers* and *Providers*, just some actors (e.g. nodes in this case) will have both of these roles, depending on situation. In each pair (*streamer-node* and *node-viewer*) payments will be still done only in one direction.

   Thanks to this observation the protocol can be simplified and take use of *payment promises* or *uni-directional* micropayment channels.

2. **requirement: high throughput and scalability**
   It is extremely important that protocol would support frequent payments of tiny amount (e.g. each 10 seconds by all participating parties of value less than 1 cent). Which means that:

   - payment solution should be able to process as many transactions per second as there are active sessions established between service providers and consumers;
   - value of transaction can be set to parts of cent (in given crypto currency);
   - transactions should be marked as final in very short period of time (ideally should have *instant finality* property);

- there have to be fast answer about transaction status with minimal networking errors and retries amount;
- transaction fee have to be very small (parts of penny) or expressed as percent of transaction value;
- there should be minimal presence on-chain, it have to be possible to aggregate payments of many sessions and from different consumers and settle them on-chain at once.

3. **requirement: utility tokens and stable coins support**
   Bitcoin, Ethers or other popular crypto currencies are volatile and may be not acceptable or too risky in some cases. Also there are many decentralised apps who have issued their own utility tokens (*MYST* token in case if Mysterium Network). This means that there is requirement for payment protocol to support transactions using *ERC20* tokens issued on Ethereum blockchain.

4. **requirement: secure**
   Digital services such as VPN could be named as *"perishable"* because they actually are selling trafic which if not used, is "gone" forever. This means that some level of risk of unpaid use of service can be acceptable. However it's should be decision of each service provider what level of risk, in exchange to increased usability and performance or lower on-chain settling fees, he can take. So in general case, double spending attempt have to be imidiatelly identified and such transactions should be rejected.

   Anonymity and permissionless nature of decentralised systems creates some cases where additional level of protection against bad acting service providers and consumers is needed. For example in decentralised VPN case, pure performing service providers may try to "clean" their raiting by simply creating new identity in the system. Bad acting consumers may organize DDoS attack and for providers it will be hard to ban them, because they may continuously create new identities. To prevent such behaviour there could be used either some kind of identity registration with staking and punishment system.

   - network identity have to be registered in given smart-contract, to do so there should be paid registration fee or staked given amount of tokens;
   - there should be not possible to pay with same coin twice (avoid double spending);

- system should be secure against different kind of attacks (e.g. DDos);

5. **requirement: decentralised**
Such important component as payments should have high level of liveness. There should be no central party which would be easy to shutdown or censor. This means that needed payment protocol should maintain at least some level of decentralisation and should be not operated by single party.

6. **requirement: low implementation complexity**
There are couple of potential ways of embeding payment solution into decentralised platform. It have to be either already very popular and scalable payment network which works on many platforms, has easy and stable APIs and is already used by many people in many other platforms. Or it have to be solution with low level of complexity so it would be easy to implement, cheap to operate and resource efficient to use it.

   (a) **Easy implementation.** Because such protocol can be used in various solutions and be slightly modificated for needs of particular system, it have to be possible reimplement all it's main parts in any popular programming language in a matters of weeks. As little of complicated schemes or components should be used.

   (b) **Cheap to operate.** To make solution easier to decentralise it should be relatively cheap run and operate its nodes. No big initial finantial investment, complicated installation and advanced hardware should be required.

   (c) **Efficient usage.** Because there is requirement to make stable and fast payment solution, less communication messages have to be send via network between parties, less intermediaries involved, better for the protocol.

7. **requirement: good user experience**
Not only technical parameters are important for successful systems. If it is hard for user to make payment, only minority of them will stay to use services. There are various angles of usability which ideally should be solved.

   (a) Consumers should be able to deposite funds using any popular crypto wallet or directly from exchanges.

(b) Users should have possibility to own just one asset (e.g. MYST token) and not be required to own any additional utility token or coin used to pay for payment network. This is especially important while using tokens on Ethereum where for each transaction have to be paid some amount of ethers.

(c) Any cryptographic proofs should be able to be transfered not only by signing party, but also by any third party. In the same time they have maintain same level of security. This allows to use cloud trustless services to send, validate or protect transactions.

(d) Because service providers are earning income on the system, they can be asked for some kind of stake or platform fee. Consumers however have to be able to avoid such requirements.

3 Overview of potential solutions

Blockchain is a replicated state machine which orders transactions on it's global state. Transactions are verified and replayed by each participant, called a *node*, in the network. This limists the throughput of the network as a whole to the lowest throughput of any of it's nodes. Increasing the load beyond that throughput may result in nodes unable to handle the load being pushed out of the network. This impose a fundamental limit of amount of transactions which can be processed.

Let's take two most popular cryptocurrencies Bitcoin and Ethereum. Their blockchains can provide throughput of 3 to 15 transactions per second which in busy period causes high transaction fees and makes on-chain transactions too expensive for micropayments.

One of solutions would be to increase block size which could allow increase throughput. There are blockchains with block limits of 128Mb (100 times higher that bitcoin's) so theoretically they can process up to couple of hundreds transactions per second. Such level of scalability work for one-time type of payments (like buying cup of coffee) but for high throughput nano payments we need solution which could process millions of transactions per second. Unfortunatelly there are phisical network, validation time (CPU) and disk space limits so blocks can't be ultimately big.

Finally there is one more problem. Participants in the network by default agree that the chain with the highest *difficulty* or more blocks put into it, is "true ledger". If some other branch mined in parallel will get longer chain, a new ledger is accepted as the true ledger. Some transactions accepted into older ledger can be not added into new one. This means that the longer waiting period before accepting payment, the bigger guarantee the transaction written in blockchain is irreversible.

For frequent transactions, the faster *finality* there is, the better. *On-chain* transactions however don't have reasonably fast finality.

To overcome *on-chain* limitations, such as scalability, high fees and long finality time, several *off-chain* protocols (or so-called *Layer 2* solutions) have been proposed. Among these techniques, one important differentiator is whether the relocated operations introduce additional consensus assumptions (like in sidechains and interoperable blockchain networks such as Polkadot or Cosmos), or allow users to restore their state to the original blockchain (like in state channels or Plasma).

3.1 Blockchain per project

The most radical solution would be to move platform specific utility token into own blockchain. There are couple of frameworks to build own blockchains with plugable consensus and application layer. Most promissing solutions are *Tendermint* developed by Cosmos Network, *Substrate* developed by Parity Technologies as part of Polkadot Network, using software of Ethereum but with different consensus algorithm and *Hyperledger Fabric* by Linux Foundation.

Nevertheless that these solutions are relatively hard to customise and launch, using them would additionally require building community of full nodes and stake pools which would guaranty network's security. Another unwanted side effect would be need of moving token into new blockchain. This is socialy complicated and hard to implements project.

Alternatively there could be launched a *sidechain* - a blockchain ledger that runs in parallel to a primary blockchain. Assets from the main blockchain can be linked to and from the sidechain. This allows the sidechain to operate independently of the primary blockchain by introducing own consensus mechanism, faster speed of transactions and features needed to some dedicated platform.

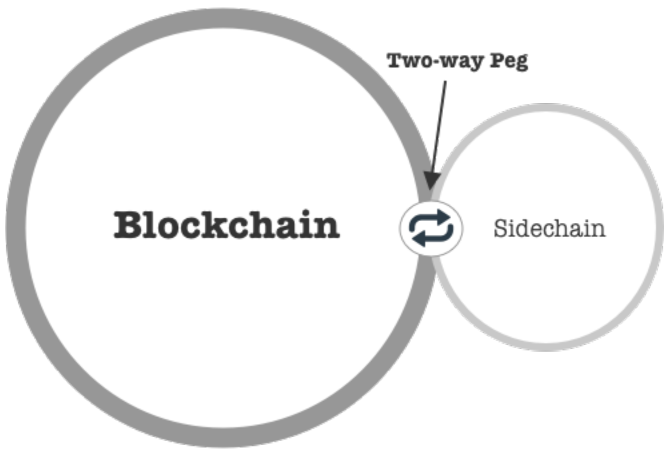

Figure 1: Two-way pegged sidechain

Thanks to possibility to use own, less decentralised but more scalable consensus algorithms, *sidechains* can provide much higher throughput than in primary blockchain. However while introducing consensus assumptions which in situation of fail will permanently compromise long-term guarantees (such as persistence of asset ownership). If state is "moved" to a sidechain and that chain's consensus mechanism fails, owners or beneficiaries of that state may lose everything delegated there, even when the primary blockchain remains secure.

Slightly different approach is taken by interoperable multi-chain networks such as *Polkadot* (see figure 2) which allows new designs of blockchains (called parachains) to communicate and pool their security while still allowing them to have the entirely arbitrary state-transition functions. This helps to bootstrap new chain much faster while having same security guaranties as whole Polkadot network.

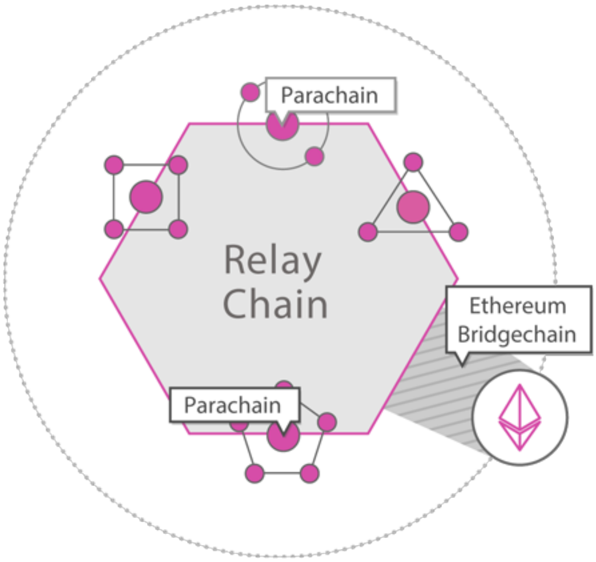

Figure 2: Polkadot network components

Despitethe the fact that *sidechains* can provided significant increase of transaction throughput and in case of *multi-chains* also relatively easy achievable high level of security guarantees, general issues of blockchains still apply. Because of blockchain nature and physical networking and storage limitations, they still can't provide throughput of hundreds of thousands transactions per second and *instant finality*, which may be needed for *micropayments* in succesfull decentralised VPN or streeming platform.

3.2 Plasma

Plasma [4] is a proposed framework for scaling Ethereum capacity by using hierarchical sidechains. Plasma type of sidechains (also called child chains) allow to do a majority of transactions outside of the "root chain" (e.g. Ethereum). Only deposits and withdrawals, the points of entry and exit, are handled on the root chain smart contract.

Similary to blockchains Plasma chain stores all it's transactions packed into blocks while using *UTXO* for balance accounting. To make sure that transactions are final, Plasma operator is doing something called a "state commitment". It is a cryptographic way to store a compressed version of the state of child-chain inside of root chain. Typically all the state is stored in *merkle trees* and only *merkle root* of each block's state is persisted into root chain.

Usually Plasma chains are run by single operators, having distributed nodes using some kind of Byzantine Fault Tolerant (BFT) consensus (e.g. Proof of Stake) is possible though.

Differently to sidechains and multi-chains, Plasma makes possible for users to leave at any time. Such action is usually refered as "exiting". This allows users safely withdrawal their funds out of Plasma even if it was shut-down by operator.

While it offers significant speed (up to 1000 tx/s) and latency improvements over Ethereum itself, Plasma cannot offer the near-zero latency and near-free transaction fees. It also requires significat storage to store it's ledger (esoecially with big amounts of transactions).

Another downside of Plasma chais is that it is complex and difficult to implement solution. Especially hard is to run it with distributed nodes which have BFT type of consensus. When Plasma chain is operated with single operator, there is risk that he will create 'fake' blocks, which may end up with mass exits out of plasma into root chain and stuck of whole network.

One more downside is that plasma is quite expensive to operate. Each couple of blocks it have to commit it's state to root chain, which means, that plasma operator would have to pay 1080 USD (as gas fee) per day.

$$3\,tx/minute * 1\ 440\,minutes/day = 4320\,tx/day$$

$$1\,tx = 25\,cents$$

$$4320\,tx/day * 0.25USD/tx = 1080\,USD/day$$

This isn't big problem when there is already significant load in the network, however in the beginning covering such costs can be problematic.

And finaly, regardless of that Plasma transaction throughput is significantly higher than Ethereum's, it is still not enought for distributed VPN network needs. As alternative, there could be possible to build some kind of payments channels on top of Plasma (see 3). In this way parties could do pear-to-pear transactions while having active service and close channels right after closing connection.

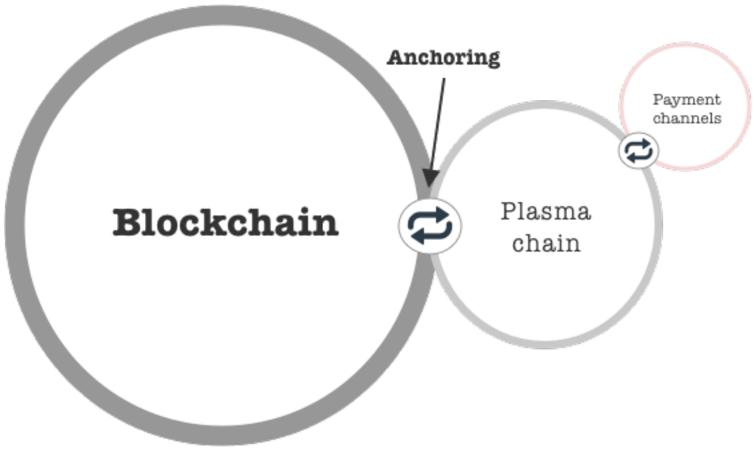

Figure 3: Plasma + payment channels

In situation when there are more that one transaction per second throughput, Plasma transactions can be relativelly cheap (less that 1 cent per transaction) and can have fast including into block (from 1 to 10 seconds, depending on implementation). This allow opening and closing channels when needed and avoding long waiting times.

Unfortunatelly this solution is even more complicated that using just plasma and has similar cost and decentralisation issues mentioned above.

3.3 Payment and state channels based solutions

A *micropayment channel* is class of techniques designed to allow parties to exchange digital value without commiting all of the transactions to the blockchain. In a typical payment channel, only two transactions are added to the blockchain but an unlimited or nearly unlimted number of payments can be made between participants.

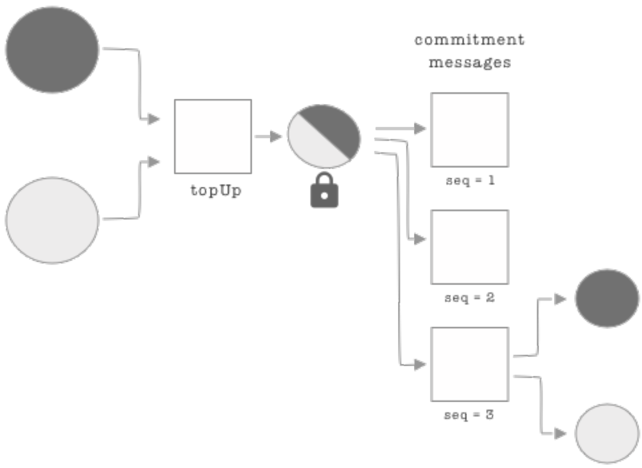

Figure 4: Sequense based payment channel

To open payment channel parties have to lock some funds into multisignature smart contract (see figure 4). This allows parties to updated channels balances off-chains while beeing sure with hight probability that funds will be not double spent or stolen.

There are various payment channels techniques such as sequence based channels, duplex channels, time-locked channels, etc. Any of them would improve security and would help to avoid double spending problems comparing to initially proposed digital cheques solution (see section 2.2). Also that would allow providers to wait longer until settling on-chain what could reduce settling costs because some consumers may return to use service via same provider again.

3.3.1 State channels

State channels are the general form of *payment channels*, applying the same idea to any kind of state-altering operation normally performed on a blockchain. Moving these interactions off of the chain without requiring any additional trust can lead to significant improvements in cost and speed. State channels will be a critical part of scaling blockchain technologies to support higher levels of use.

The basic components of a state channel are very similar to that in payment channels.

1. Part of the blockchain state is locked via multisignature type of smart contract, so that a specific set of participants must completely agree with each other to update it.

2. Participants update the state among themselves by constructing and signing transactions that could be submitted to the blockchain, but instead are merely held onto for now. Each new update ”trumps” previous updates.

3. Finally, participants submit the state back to the blockchain, which closes the state channel and unlocks the state again (usually in a different configuration than it started with).

Because tokens on Ethereum blockchain are represented in form of state in smart contracts, state channels are the way how to move token interacion into *layer 2*.

3.3.2 Downsides and benefits of channels

The downside of using payment channels is that parties have to lock funds into multisigature smart contracts which in comparison to *digical cheques* means additional on-chain transaction for opening channel and needs of having locked bigger amounts of funds. There is not possible to reuse same funds in another channel before closing previous one. Another downside is time to service. Opening new channel will take some time (depending on type of blockchain and network load this make take from one minute till up to couple of hours) which makes bad user experience.

There are also couple of additional benefits of payment channels:

- **Privacy** – each transaction is known only to participating parties, which allows them to have privacy in their individual transactions such that the public only knows the result which has been shared on-chain.

- **Instant finality** – parties can sign and exchange messages instantaneously without having to wait for confirmation from the blockchain. This is leading to a much better user experience.

- **Lower cost** – digital value is exchanged with the only a few on-chain transactions being made for creating the channel and for closing the channel.

3.3.3 Micropayments networks

Payments channels is very promising technique for micropayments. However opening new channel requires on-chain transaction which is both slow and expensive and can't be done in setups when there are a lot of service providers. In this situation payment channels based solution is reasonable only when participants do not need to be connected to everyone else (see figure 5).

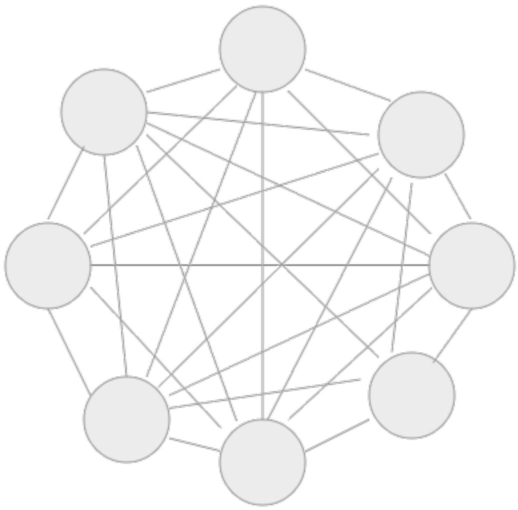

Figure 5: Everyone makes channel with anyone else

Many existing research on payment channels have focused on exploring the design space of how to structure payments through intermediaries [1, 2, 3]. Let's suppose that Alice has a payment channel with Ingrid, and Ingrid has one with Bob. If Alice pays Ingrid off-chain and then Ingrid pays Bob the same amount, this is equivalent to Alice paying to Bob off-chain, without requiring a new Alice-Bob payment channel to be set up.

In situation when parties don't have channels with single intermediary, a micropayment channel network can be used together with a routing algorithm to send funds between any two parties in the network (see figure 6).

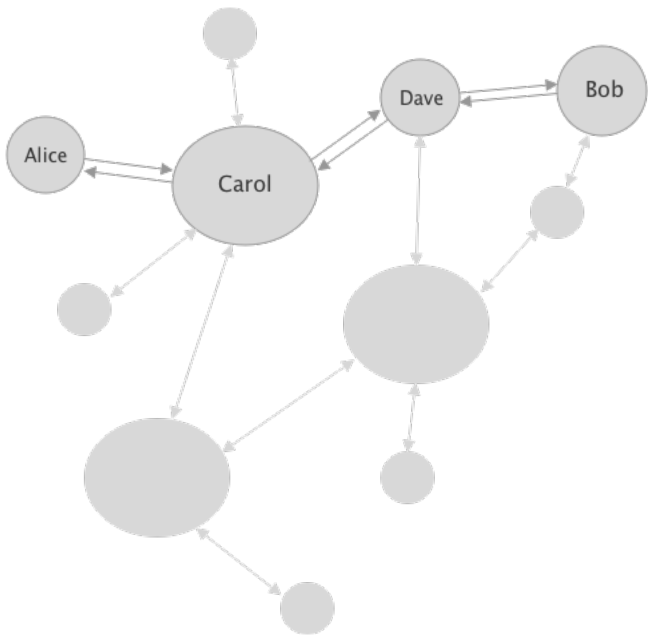

Figure 6: Micropayment channels network

As micropayment channel network can keep most transactions off-chain, blockchain based currencies may scale to magnitudes larger user and transaction volumes than any currently exsting centralised solution. Also, micropayment channel networks allow for fast transactions thanks to *instant finality* property of payment channels. Transaction is final as soon as it is signed and sent to another party, the blockchain latency does not matter.

3.3.4 Hashed Timelocked Contracts and Virtual Channels

There are various ways to create micropayments networks trustlessly (i.e. to ensure that Alice pays Ingrid if Ingrid pays Bob). The most popular is Hashed Timelock Contract (HTLC) based approach [3]. In it's canonical way an equal amount of funds from both payment channels are locked up in a way that they can only be spent if a certain hash is revealed before a certain deadline (thus, locked "by hash" and "by time").

This technique can allow payments to be securely routed across multiple payment channels and is used in Bitcoin's Lightning Network and Ethereum's Raiden Network.

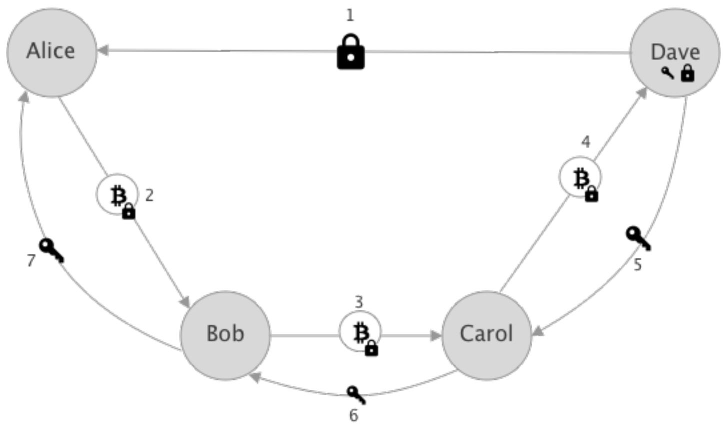

Figure 7: HTLC based payment - Alice pays Dave

In figure 7 we see atomic value exchange over three micropayment channels (Alice to Bob, Bob to Carol, Carol to Dave). If Alice will pay to Dave via Bob and Carol, she needs to ensure that Bob and Carol cannot run away with her money. To do so, Dave generates pre-image R, which is random number (shown as key on scheme), and then shared with Alice hash H (shown as lock on scheme) of this pre-image. In step 2 Alice generates an HTLC with Bob that says *"I will pay you X coins if you show me the pre-image R. If you don't show R during period Δt, I will take back my coins."*. In step 3 Bob generates similar HTLC with Carol, but sets period of $\Delta t - 1$. In 4 step Carol do same with Dave, but with even shorter period of $\Delta t - 2$. Then in step 5 Dave will show R to Carol, in next step Carol shows R to Bob and finally Bob shows it to Alice. At this point payment of X amount from Alice to Dave is finalsed.

HTLCs can be established in any chain of any length consisting of different payment channels. As an incentive for intermediate hops to forward transactionsa small fee can be charged for using the service of the channel. Fee payments are also justified as the balance on a channel gets shifted, which is only beneficial for balancing a lopsided channel. After a successful transaction with HTLCs, channel parties do not need to broadcast their contract and can just replace their HTLC with a new commitment transaction without an HTLC. HTLCs can be combined with timelocks or revocable transactions changing the output of the HTLC accordingly.

In Ethereum blockchain, which supports advanced smart contracts, routing payments could be done in an alternative form via a technique known as *Virtual channels*, introduced by *Perun* paper [2] and simultaneously similar technique was proposed by *Counterfactual* protocol (named as Meta channels), which use an intermediary that serves as a *"virtual payments hub"*. Anyone with a payment channel connected to the hub could establish virtual channels between each other (see figure 8). Unlike routing payments via HTLCs, Hub does not need to be involved in every payment between Alice and Bob. This property reduces latency and costs, while increases privacy.

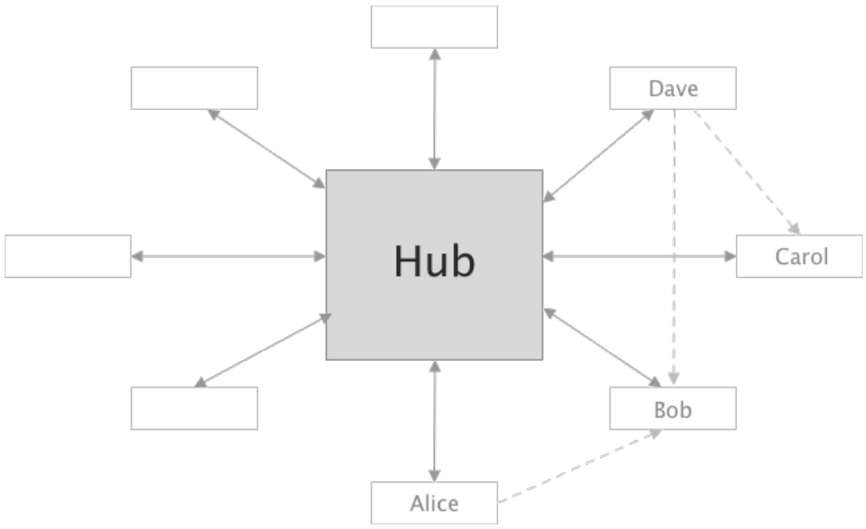

Figure 8: Virtual Payment Channel Hub

To open virtual channel, Alice and Bob essentially need to lock-up a set number of coins from their payment channel with Hub. The amount of locked coins will become the value of the virtual channel between Alice and Bob. Notice, that the Hub remains financially neutral by simply mirroring balances.

This technique could be repplied to increase the length of the channel (see figure 9). Because a virtual channel is an instance of an off-chain contract, increasing the length not require another transaction on-chain.

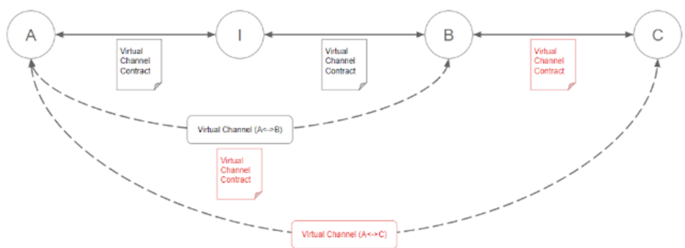

Figure 9: Bob becomes the Hub for channel between Alice and Carol

Virtual channels also represent a different business model from channel routing. HTLCs have a *"pay-per-payment"* fee model where there is need to incentivize each intermediary to route each payment. Virtual channels, on the other hand, have a *"rent-a-path"* fee model. In this model, an intermediary acts as a virtual payment hub that has direct channels with multiple parties. If Ingrid is the intermediary, then Alice and Bob pay Ingrid to keep the channel open for a certain period of time. Such a model might have better economics for high-volume micropayments.

While Perun and Counterfactual have introduced novel ways of routing payments across multiple intermediaries, HTLC-based routing is currently the primary implementation on blockchain mainnets, is better tested and much easier to implement.

3.3.5 Known problems of micropayment networks

Micropayment channels networks looks really promising but unfortunatelly they have couple of negative sides.

Downside of using HTLC based micropayment networks is complicated routing. *Lightning Network* is great for time to time low value payments, however it is much harder to use for frequent transactions into same receiver because of need of constantly checking if all parties in selected route are still alive and good to be used (e.g. all channels has enough funds to send payment forward).

Complicated routing issue could be solved by using payment hub model, when one of a fewnintermediaries have a lot of channels with end users. Downside of this solution is that this requires the Hub to lock a lot of funds in channels with all actors in the network. Another downside is that this solution is centralised and if hub's operator will be required to stop activities or will decide to shutdown servers by any reason, the whole network's payments activities will be stopped.

Another critique of payment channels network is that both parites (paying and receiving) have to be on-line during payment. This issue however is not valid for high frequency nano payments in decentralised networks, because we're sure, that payer (service consumer) and payee (service provider) are on-line, otherwhise service wouldn't be provided and there would be no need for payment.

Finally, micropayment channel networks are really usefull when there are a lot of people already using them. And when there are huge support for all popular programming languages and platforms. Unfortunatelly, them most popular solution (Lightning Network) don't support tokens. On Ethereum there are a few promissing projects (e.g. Raiden, Connext, Perun or Counterfactual),

3.4 Comparison and conclusion

4 The protocol

In this section idea of *"Accountant pattern"* method will be described. It is essentially solution combining *Lightweight payment Hubs* and *Payment promises* based uni-directional channels.

References

[1] Jeff Coleman, Liam Horne, and Li Xuanji. *Counterfactual: Generalized state channels*. 2018.

[2] Stefan Dziembowski et al. "Perun: Virtual payment hubs over cryptocurrencies". In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 327–344.

[3] J. Poon and T. Dryja. "The bitcoin lightning network: Scalable off-chain instant payments". In: (2016).

[4] Joseph Poon and Vitalik Buterin. "Plasma: Scalable autonomous smart contracts". In: *White paper* (2017), pp. 1–47.