Paying your Internet, one byte at a time

Sandra Siby
Supervisor: Christian Decker

December 5, 2013

Abstract

Acknowledgements

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 2

Background

2.1 Bitcoin Overview

Bitcoin is a decentralized digital currency system based on peer-to-peer technology. It was introduced in 2008 by Satoshi [1]. All functions, such as creation of new coins ('bitcoins'), processing and verification of transactions, are performed collectively by the network instead of a central authority. This reduces transaction costs, the need to trust a single entity and unpredictable inflation.

Important terms of the Bitcoin protocol are explained in the following sections [2][3][4]:

Wallet

Bitcoin uses public key cryptography to authenticate transaction information. Every user has a set of public and private keys. The public key is disclosed to others and is used to derive an address to which bitcoins can be sent. The private key is kept secret since it is required by the owner to send bitcoins. A wallet is a file that contains this set of keypairs and information about the transactions that were carried out using the keys and the current balance. There are currently different implementations of the Bitcoin wallet.

Transaction

A transaction represents a transfer of bitcoins from one address to another. If A wants to send bitcoins to B, it creates a transaction record, signs this record with her private key and broadcasts it to the network. A transaction record doesn't solely contain information about how many coins need to

be transferred to B. Instead, it can be considered to be a collection of *inputs* and *outputs*. Inputs are hashes of previous transactions that provided bitcoins to A. They are used as a proof of ownership i.e. A does indeed own the bitcoins that she will be spending in this transaction. In the output section, A puts B's address and the amount to be transferred to B, a transaction fee (see next section) and, if the input amount is higher than amount to B, the rest of the bitcoins as an output back to A (change).

While a transaction verifies A's ownership of bitcoins, it does not prevent A from using the same bitcoins in another transaction, a problem called 'double-spending'. In order to ensure that A spends the bitcoins only once, the transaction record is added to a public ledger known as a *blockchain*.

Block chain

For a transaction to be considered valid, it has to be committed to a public ledger that contains all previous transactions. This public ledger is called a blockchain - it is a chain of entities called blocks, where each block contains a set of new transactions not present in the previous block.

A new block of transactions is added to the block chain by nodes called miners. When A's transaction is broadcast to the network, it is received by miners. These miners collect transactions that took place at the same time. Each miner performs the following operations: A Merkle tree of the transactions is created, its root is calculated and added to the block header. A hash of the last block in the current blockchain is also added to the block header, along with a timestamp. This hash helps in proving the chronology of the block chain. Finally, since several miners are working on creating a block but only one block can be added to the block chain, miners compete to get their block accepted into the chain by searching for a value. This value, when added to the block and hashed, should give a predetermined target value (a large number starting with several zeroes). Since obtaining the input of a hash function given its result is computationally infeasible, a miner has no choice but to keep on trying different nonces and hashing the block till the target value is obtained. This value is known as proof-of-work - it implies that the miner put effort into finding the value but is easy to verify by others.

The first miner that finds the value broadcasts the new block to the network. The other peers verify the value and the block is accepted and added to the block chain. Transactions that were not part of this new block will be validated in future rounds.

In the rare case of two miners finding the answers within a few seconds from each other and introducing new blocks into the network, there is a

disagreement over which is the accepted block. This leads to forks in the block chain as different nodes take different blocks and build upon them. However, there is bound to be a time when one of the forks grows longer than the others. At this point, this branch is taken a part of the block chain and all the other forks are discarded (transactions in the discarded branches have to be validated again).

The target value is determined in such a way that the solution can be found in 10 minutes on average. Note that the predetermined target value's difficulty is increased with time to account for advances in computing power. As time goes by, it becomes more and more difficult to find the nonce that hashes to he target value.

Coin creation and transaction fees

Since mining is a resource intensive task, it has to be incentivized. Hence, every time a miner is successful in creating a block that is accepted into the block chain, he/she gets a reward of brand new bitcoins. The first transaction in the block is always this reward and is called a coinbase transaction. This is how new coins are introduced into the network. The number of new coins per block started off at 50 initially but halves every 210,000 blocks. This regulates the supply off new coins. In addition to this, miners get a transaction fee. The total number of bitcoins is capped at 21 million. Once this ceiling is reached, miners will earn coins only via transacion fees.

(Picture???? Maybeee TBD)

2.2 Micropayment Channel

A micropayment is a transaction involving a very small sum of money. Traditional modes, such as credit cards, aren't efficient when it comes to such transactions because the transaction fee becomes comparable to the amount being transferred.

Bitcoin, with its lack of a third party and relatively low transaction fees, has high potential to be used in micropayments. However, there are still a few issues that crop up. First, the overhead of a transaction might be higher than what the transaction was worth. Second, if several transactions are sent very fast, they might be down-prioritised or not relayed due to anti-flooding algorithms in the network. Finally, there is a minimum value a single transaction can send, determined by the number of bytes required. For our case of the access point, we want to pay as we go, i.e. send multiple small pay-

ments during our surfing session every time we want to prolong the session. Fortunately, there is a micropayment channel implementation in bitcoinj (a Java implementation of the Bitcoin protocol) that enables repeated micropayments between parties without encountering the issues mentioned earlier.

Consider a client C and a server S. The micropayment protocol allows C to make repeated payments to S. The client first creates a multi signature transaction that requires C's and S's signature. This transaction locks in a certain amount of money. However, before signing this transaction and allowing it to be broadcast, the client creates a transaction called a refund transaction and gives it to the server to sign. This transaction refunds the entire amount to the client but is time-locked. This means that the value would be refunded only after a certain period of time has elapsed. This refund transaction is to protect the client from losing money if the server goes down or does not provide the service it is supposed to. Once the server has signed this transaction, the client sends the mutlisig transaction with its signature. The server signs it and broadcasts it to the network. The channel can now be considered to be open between C and S.

C then creates a third transaction which has two outputs, one to C's address and one to S's. Initially, the amount is completely allocated to C's address (like the refund transaction but without a timelock). C signs and sends this to S. Whenever C wishes to make a payment to S, it adjusts its copy of the transaction by reducing the amount allocated to it and increasing S's output. It lets the server know about these changes. S adjusts its copy of this transaction to reflect the changes. Finally, when the client is done, it notifies the server. S signs the final copy of the transaction, broadcasts it to the network and closes the channel. If the server leaves before the client closes the channel, the client gets the money refunded on expiration of the timelock. The micropayment protocol is shown in Figure 2.1. (NOTE SS: IMPROVE PIC FONTSIZE + WORD ORDERING)

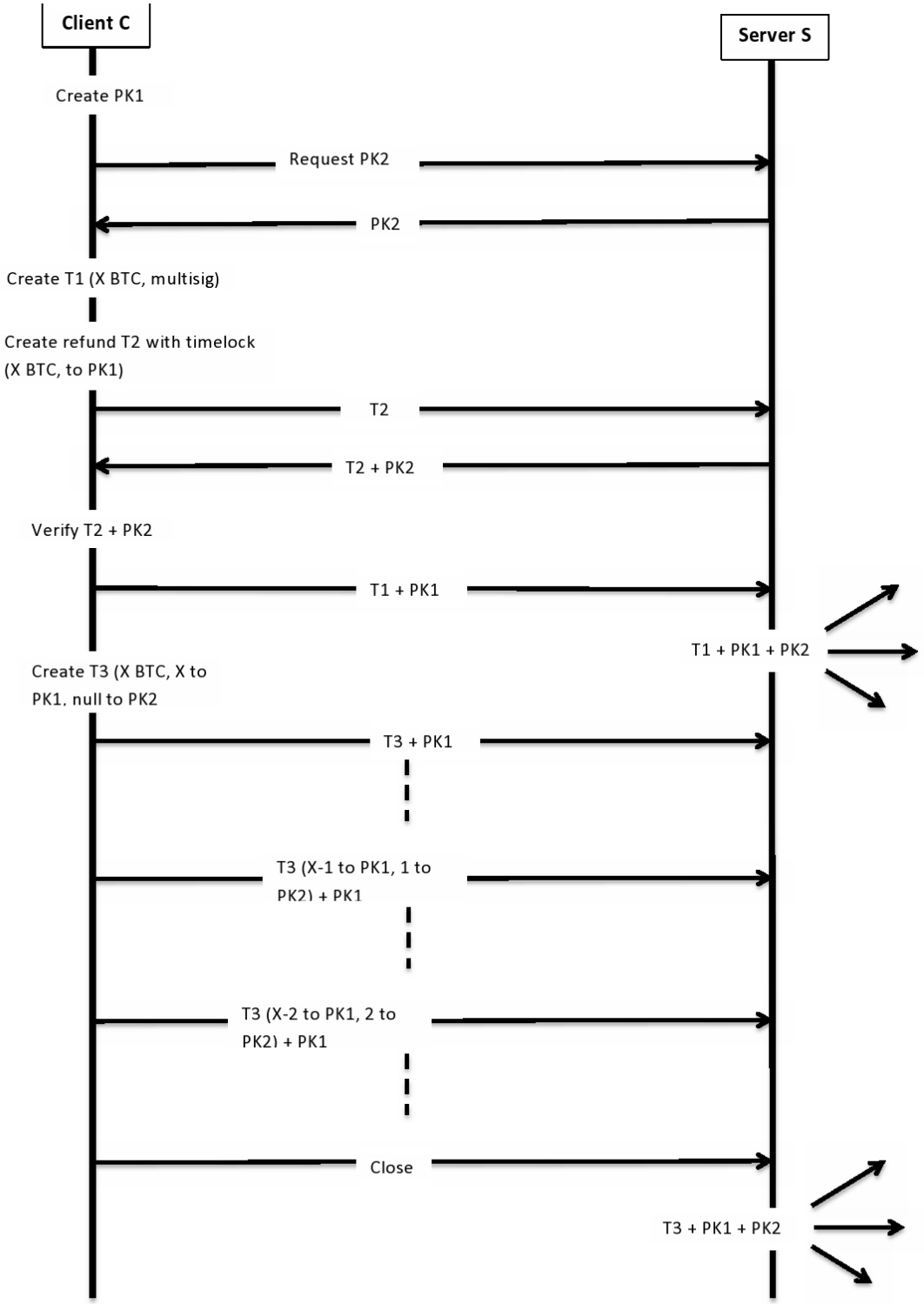

Figure 2.1: Micropayment protocol

Chapter 3

Implementation

Our use case consists of an access point that provides Internet access to untrusted clients in exchange for payment in bitcoins. The communication between a client and an access point can be considered to contain the following steps:

Discovery - The access point announces its availability to clients.

Negotiation - The access point and the client come to an agreement where the client confirms that it will pay for the services of the access point.

Channel Establishment - The client and access point set up a micropayment channel for the client to make payments on the go. The access point provides Internet access when the channel is established.

Closure - When the client decides to stop using the access point's service, it closes the micropayment channel. On closure, the access point stops providing access to the client.

For our proof-of-concept, we set up a Raspberry Pi as a Wi-Fi access point. The Raspberry Pi is a low cost, compact, single-board computer that can be used for a variety of applications. We chose the Raspberry Pi because of its cost and ease of use. (Note to Chris: Better way of saying this? Not so happy with it, sounds lame. Also should I say that trying to modify a vendor access point might be out of scope?). The communication steps are described in the following sections. (Note SS: Add Pic - Rpi, figure showing AP/client).

3.1 Discovery

We set up the Pi as an access point that broadcasts a Wi-Fi service to clients and then redirects the traffic to an Ethernet cable. The Pi was connected to the router via Ethernet cable and the Wi-Fi dongle was inserted. The

Figure 3.1: Discovery

setup is shown in Figure X (Note SS: Take pic tomorrow). We installed the ISC DHCP server on the Pi. DHCP (Dynamic Host Configuration Protocol) is a protocol used to assign IP addresses and configuration information to devices so that they can communicate on the network. The ISC DHCP server accepts clients' requests to connect to it and replies to them. We edited the DHCP configuration file on the Pi to include details about the subnet, name servers, broadcast address, lease time and IP address range. This means that any client that wants to connect to the access point will belong to the subnet specified and be assigned an IP address in the specified range. We also set up the DHCP server to run on the wireless interface wlan0. Next, we assigned a constant IP address to the wlan0 interface. (Finally, we configured the access point to be password protected - Note to Chris: I shouldn't be doing this step right? Untrusted clients do not know the password!) Finally, we set up Network Address Translation between the wireless (wlan0) and Ethernet (eth0) interfaces so that clients connect to the Wi-Fi and their data gets tunneled through the Ethernet IP.

On running the host access point server, we were able to see the new wireless network in the list of available networks (Figure 3.1).

3.2 Negotiation

3.3 Channel Establishment

Once the client clicks on the accept link, a Bitcoin client application is run. The access point has a Bitcoin server application running, that listens for new connection requests. The client connects to the server and starts the micro-payment channel establishment process. The client first creates a transaction that locks in a certain amount of money. This transaction is a multi-signature transaction that requires both the client's and the server's signatures. The multi-signature transaction can be thought of as a shared account between the client and the server. However, before broadcasting this transaction, the client creates a refund transaction. This transaction is linked to the output multi-signature transaction and refunds the entire amount to the client. However, it is time-locked, i.e., it refunds the amount only when a specified time period has elapsed. The refund transaction protects the client from losing money in case of the server not providing the service it advertised. The client sends the refund transaction to the server for it to sign. Once the transaction has been signed, the client signs its multi-signature transaction and sends it to the server. The server signs it and broadcasts it to the network. The channel is now considered to be open.

When the channel is opened, the server has to modify the firewall rules to allow access to the client. It takes the client's MAC address and adds it to the firewall rules so that the client no longer gets redirected to the portal page. It also removes previous connection tracking information about the client. If this tracking information is not removed, the client might still get redirected to the portal. Once the server has modified the rules, the client is able to access the Internet.

The client then creates a transaction that is connected to the output multi-signature transaction it created in the beginning. This transaction has two outputs, one to the client's address and the other one to the server's address. Initially, the entire amount is allocated to the client. It signs this and sends it to the server. Every time the client wants to make a payment, it adjusts its copy of this transaction - it reduces the amount allocated to it and increases the amount allocated to the server. When the client informs the server about these changes, the server adjusts its copy. This process continues, with the server's amount increasing and the client's amount decreasing till the client decides to stop the payment.

3.4 Closure

When the client decides to stop using the services of the access point, it closes the channel. When the server detects a channel closure, it signs the final copy of the payment transaction and broadcasts this to the network. It also modifies the firewall rules to no longer give access to the client. It removes the client's connection track information so that the client is once again redirected to the portal page.

If the server goes down before the channel is closed by the client, the refund transaction comes into play. The client is refunded the amount 24 hours later.

If the client has not closed the channel before the channel has expired, there is a risk of the client getting the refund even though the server has been providing Internet access. To avoid this, the server has to close the channel before the expiry time and a new channel has to be created. (Note to CD: Combine 3.3 and 3.4 to one section - micropayment channel?)

Chapter 4

Evaluation

Chapter 5

Conclusion

Bibliography

[1] Satoshi paper

[2] Decker paper

[3] Khan video

[4] Bitcoin website