# Sidechains

“A sidechain is a blockchain which validates data from other blockchains” [1]. Sidechain is a blockchain that runs parallel to the main blockchain. It extends the functionality of the main chain enabling decentralize transfer of assets and tokens between the two chains. Simply put sidechaining allows coins to be moved between two separate blockchains. Tokens from the main chain can be securely moved to the sidechain and used in these chains. A short summary of how it is done is given below.

**Sidechains and Two-Way Peg**

A sidechain is a separate blockchain which is attached to the parent with the help of a two-way peg which allows assets to between the two chains to be interchangeable at fixed deterministic rate. This two-way peg is implemented with the help of a method known as Simple Payment Verification (SPV) see 4.1. The user who wants to transfer assets from one chain to another first sends their coins on to a special output address where they become locked and un spendable. Once transaction is completed SPV confirms it across both chains and after a special waiting period known as contest period has elapsed the equivalent amount becomes available and spendable in the sidechain and vice versa. In order to add more security beyond SPV it requires the parent to soft fork an update to its core wallet software in order to verify transactions across both chains [2].

## SPV

Simple payment verification is a light weight and quick method for checking the validity of transactions without verifying everything. “Essentially an SPV proof is composed of (a) A list of blockheaders demonstrating proof-of-work (b) a cryptographic proof that an output was created in one of the blocks in the list. This allows some verifiers to check that some amount of work has been committed to existence of an output. Such a proof may be invalidated by another proof demonstrating the existence of a chain with more work which does not include the block which created the output. [4]”

## RootStock RSK SideChain

It is a sidechain to bitcoin. It is Two-way pegged to bitcoin. RSK code aims to be backwards compatible to ethereum i.e. code can be written in solidity or serpent and can be used on ethereum. RSK aims to be a platform for decentralized applications and smart contracts on bitcoin. Miners are incentivized to mine smart contracts on RSK by rewarding them with bitcoins. Bitcoin miners can simultaneously mine both blockchains and rewarded through a process called “merge-mining”. Rootstock has its own version of EVM. RSK developers claim to be able to perform 300 transactions per second but comes at some loss of decentralization. RSK website claims currently over 50% bitcoin miners are implementing RSK plugin. Currently the developers are working on what is called a federated peg which is essentially a system of notaries and a multisignature exit address. The developers aim to go from federated peg to a two-way pegged system based on SPV.

**Federated peg:** is a system consisting of a set of notaries and a multisignature exit address. When you send funds to this exit address you can create an SPV proof on RSK sidechain. This SPV proof allows you to convert the locked bitcoins in the federated address into bitcoins on rootstock sidechain. This is done automatically. However, moving funds from RSK back to bitcoin requires collaboration of federators. Basically a smart contract acts as bridge master and controls all unspent transaction outputs. This contract broadcasts a transaction to federators by using a log message. On receiving this message, federators send signatures to bridge master who combines all these signatures in to a fully signed transaction. This signed transaction is broadcast to RSK blockchain where any user can put this transaction onto bitcoin blockchain. This unlocks bitcoins on the bitcoin blockchain.

**Requirements**

RSK is a nice idea but in my research I came across a few requirements that need to be met on the parent blockchain i.e. bitcoin or litecoin before RSK can be fully realized.

RSK requires SEgwit2.x activation on bitcoin.

Needs new scripting opcodes.

Ability to soft fork some of the scripting language.

## Advantages and Disadvantages of Sidechains

**Advantages**

Several new features can be added to the parent chain without causing a hard fork and risking a network split.

**Disadvanatges**

Sidechains introduce additional complexity on several levels. We need to have miners support the new chain in order to guarantee security and consensus. In addition, we need to devise mechanism for two-way pegged and adequate compensation mechanisms for miners for supporting and mining in the side chain. In the end decision to use side chain might depend on exact use case of the topic and thesis but in any case, it might significantly increase workload and may detract from the app development itself. In many ways side chains in my opinion seem like a good way to test some hard fork upgrades of the main chain in a safe manner rather than having a longer term viability as anything other than secure sandbox testing environment or platform.

# Private chains

Are permissioned block chains, analogy to private intranets? Nodes in private chain are pre-selected only verified and pre-approved participants can become nodes and interact with the blockchain. These chains can have better performance in the short term and have better security, but in the long term it might introduce single points of failures or few points of vulnerabilities. These systems can be easily censored and regulated as essentially, they are not truly decentralized and subject to regulatory and other constraints

# Use cases

## Smart contract for the P2P Parcel Delivery

<https://github.com/manuhalo/iothackffm>

This is a minimum proof of concept implemenation of a decentralized peer to peer parcel delivery system based on smart contracts. The sender specifies the intended recipeint and any additonal constraints e.g maximum delivery time and / or maximum temperature the package can be exposed to during shipment. The “Deliverers” can see all pending delivery requests and can propose themselves as candidates to carry them out by proposing a fee they will charge for making this delivery. The candidate offer of carriage comes with an expiration time so that candidates / deliverers don’t get stuck in a commitment indefinitely. The sender selects a candidate offer by funding the agreed smart contract. Upon shipment pick up both sender and receiver have to confirm that it has been picked up and at this stage the status changes to shipping. If the package is delivered on time and without violating any contract constraints the payment is unlocked and deliverer is allowed to withdraw funds. If any terms of the contract are violated the deliverer just doesn’t get paid for his service.

Future Plans:

Uport Reputation to implement reputation based system such as ebay.

**Suggestions:**

Uber like model for this service should work better. I,e instead of candidate proposing the fee the Dapp should calculate the fee based on contract constrains distance and delivery times requested by the sender. Faster times might require advanced deliverers e.g. people with drones or helicopters.

Deliverer should need to fund the wallet to incase of high value items to cover the costs

## Bluetooth Charging Station

https://github.com/DanielPollithy/bluetooth\_drone

This is an app for charging raspi bluetooth drones using hikey charging stations.

**Assumptions** drone is running the cleint side of the app while charging station is running the server side of the app (server.py)

This is an app for charging drones. The station listens to a bluetooth request from a nearby drone which needs to recharge. The charging station is booked from a website? By drone itself? Or the operator?. (drone\_poller.py). The drone and the charging station have ethereum addresses. The correct charging station is selected based on etehreum address of the drone and the charging station (end\_charging.js). The charging station needs to be booked before it can be used. Makebooking.js run by operator in web client or automatically done by the drone ?. Bluetooth RSSI is used for calculating distance between the station and the drone before establishing connection.

**Open questions:**

Is the exchange of protocol information carried out over bluetooth? i.e. to release funds to charging station?

I did not see any payement logic in the app /code is it not implemneted yet ? or is handled some where else

As far as I could understand the code only handles establishing connection, making boooking and finding correct charging station.

## DroneChainWeb

<https://github.com/gosticks/DroneChainWeb>

has drone Ethereum address and Station Ethereum address as well as GPS cooridnates for dornes and stations (**dataprovider.js**). May be an app for using drones fro making deliveries?

**generateData () and PostNewBooking() APP.vue.** No details about how drone makes deliverires can be found. Just that the coordinates of target and coordinates of station where the drone is initially being charged are entered through this app.

## Peakon

<https://github.com/quiquee/peakon>

Seems to be an app to get reward from breacons placed on certain mountains. The reward is distributed once data which appears to be weather data is read and parsed from the beacon placed on mountains.

# References

[1] <http://avc.com/2014/10/sidechains/>

[2] <https://www.youtube.com/watch?time_continue=66&v=g9_MakNlHDA>

[3] <https://gendal.me/2014/10/26/a-simple-explanation-of-bitcoin-sidechains/>

[4] <https://blockstream.com/sidechains.pdf>

[5] <https://github.com/ethereum/wiki/wiki/White-Paper>