Nov 6th, 2015 Jeff Coleman (/author/jeff/)

State Channels

This post is the second in a series which aims to clarify uncommon terms that I use. Like those that follow, this term has proven useful in my own understanding of blockchains and blockchain-related technologies. I hope that by clearly explaining it here, the term can become useful for others as well.

State channels are a very broad and simple way to think about blockchain interactions which *could* occur on the blockchain, but instead get conducted *off* of the blockchain, without significantly increasing the risk of any participant. The most well known example of this strategy is the idea of payment channels (https://bitcoin.org/en/developer-guide#micropayment-channel) in Bitcoin, which allow for instant fee-less payments to be sent directly between two parties. **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 simple:

1. **Part of the blockchain state is locked** via multisignature (http://bitcoin.stackexchange.com/questions/3718/what-are-multi-signature-transactions) or some sort of smart contract, so that a specific set of participants must completely agree with each other to update it.
2. **Participants update the state amongst 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).

That's it! If the "state" being updated between participants was a digital currency balance, then we would have a payment channel. Steps 1 and 3, which open and close the channel, involve blockchain operations. But **in step 2 an unlimited number of updates can be rapidly made without the need to involve the blockchain at all**--and this is where the power of state channels comes into play, because only steps 1 and 3 need to be published to the network, pay fees, or wait for confirmations (https://en.bitcoin.it/wiki/Confirmation). In fact, with careful planning and design, state channels can remain open almost indefinitely, and be used as part of larger hub and spoke systems (https://www.mail-archive.com/bitcoin-development@lists.sourceforge.net/msg06576.html) to power an entire economy or ecosystem.

Despite my simple description here, state/payment channels have generally been perceived as quite complicated (https://www.youtube.com/watch?v=XdbsfRYskMk). There are several reasons for this, and one of them is that **there are some important subtleties hidden in my phrasing** of the three steps. Let's take a closer look at what these simple phrases imply, starting with:

> *could* be submitted to the blockchain

In order for state channels to work, participants have to be assured that they *could* publish the current state of the channel to the blockchain at any time. This results in some important limitations, such as the fact that **someone has to stay online** to protect each individual party's interests until the channel is closed.

Imagine that when we initiated a payment channel I started with 100 bitcoins and you started with 10. If we first sign an update that transfers 10 of those bitcoins to me, and then *later* sign an update that transfers 50 back to you, the later update is obviously more beneficial to you than the earlier one is. If you were to unexpectedly lose internet access (http://www.slashgear.com/three-arrested-for-trying-to-cut-undersea-internet-cable-27275579/), and I were to pretend the second update never happened, I might be able to publish the first update to the blockchain and effectively *steal 50 bitcoins from you*! What you need is somebody to stay online with a copy of that later transaction so that they can "trump" the earlier one and make sure your bitcoins are protected. **It doesn't have to be you**--you could send a copy to many random servers who agree via smart contract to publish it only if needed (for a small fee of course). But however you do it, you need to be assured that the latest signed update to the state is available to trump all others. Which leads us to our next subtle phrase:

> Each new update "trumps" previous updates

To make this part of the state channel work, **the locking and unlocking mechanisms have to be properly designed** so that old state updates submitted to the blockchain have a chance to be corrected by the newer state updates which replaced them. **The simplest way is to have any unlocking attempt start a timer**, during which any *newer* update can replace the old update (restarting the timer as well). When the timer completes, the channel is closed and the state adjusted to reflect the last update received. The length of the timer would be chosen for each state channel, balancing the inconvenience of a long channel closing time with the increased safety it would provide against internet connection or blockchain problems (https://bitcoin.org/en/alert/2015-07-04-spv-mining#list-of-forks). Alternatively, you could structure the channel with a financial penalty so that anyone publishing an inaccurate update to the blockchain will lose more than they could gain by pretending later transactions didn't happen.

But the mechanism ends up not mattering very much, because (going back to the previous point) the game theory of this situation puts a twist on things. **As long as this mechanism is theoretically sound, it will probably never have to be used**. Actually going through the timer/penalty process may introduce extra fees, delays, or other inconveniences; given that *forcing* someone into the mechanism can't give you any advantage anyways, **parties to a state channel will probably just close the channel out by mutually agreeing** on a final channel state. This final close-out operation needs to be fundamentally different from the normal "intermediate" updates (since it will bypass the "trumping" mechanism above), so **participants will only sign a final close-out transaction once for each portion of the state locked within a particular channel**.

The details of these "subtleties" aren't especially important. What it all ultimately breaks down to is that **participants open the channel by setting up a "judge"** smart contract, **sign promises to each other** which the judge can enforce and adjudicate if necessary, and then **close the channel by agreeing** amongst themselves so that the judge's adjudication isn't needed. As long as the "judge" mechanism can be assumed to be reliable, these promises can be counted as instant transfers, with the judge only appealed to in exceptional circumstances, such as when one party disappears.

Of course, these details are only *part* of the reason people think that state/payment channels are complicated. A much bigger one is that **Bitcoin payment channels *are* complicated**. Building a "judge" mechanism in Bitcoin with even reasonably useful properties (http://www.tik.ee.ethz.ch/file/716b955c130e6c703fac336ea17b1670/duplex-micropayment-channels.pdf) is surprisingly intricate. But once you have a clear concept of state channels in general, you can see that **this only comes from trying to implement the idea in a constrained context**. Basic smart contract features like a timer mechanism and allowing two different paths to be taken depending on the signed message submitted are just plain harder to do in Bitcoin. Some of these features are being gradually added or built. Personally I expect to see working Bitcoin payment channels within the next 6 months. But by seeing that **payment channels are only a special subcase of the broader "state channel" idea**, we realise that this is a much broader technique, and that **state channels can apply to any smart contract** which deals with frequent updates between a defined set of participants. You can anticipate seeing this approach in many (if not most) distributed applications going forward.

That's all I have to say about state channels for now! I hope this deeper clarification of the term proves useful to others. My main purpose in publishing these terminology explanations is to more easily refer to these ideas in the future. As I'm sure the reader can imagine, there are a wide number of applications for state channels. Once I have laid down enough vocabulary, I hope to describe some of the more interesting ones in more detail.

terminology (/tag/terminology/) **blockchain (/tag/blockchain/)**