Ripple Inter Server Protocol built on a single-hop synchronization mechanism

Johan Nygren, @BipedalJoe

Ripple is a person-to-person money system that was invented by Ryan Fugger in 2003/2004. Ripple as a money system is unique in that the bookkeeping only needs consensus at the person-to-person level, and thus it is possible to design and build a Ripple Inter Server Protocol where the computation also only needs consensus at the person-to-person level (thus a person could run their own isolated server and share state only with their trusted relationships, if they wanted to). For this to be possible, consensus at the person-to-person level must be guaranteed. The issue with person-to-person decision making over the internet is the “two general problem”, that it is impossible to be certain about if an acknowledgement was delivered. Thus it follows that agreement is impossible. The solution to the two-general problem is to agree on a single general. The ideal way to do so, is to take turns being the general.

Lockstep, a single-hop consensus mechanism

People in a trustline-relationship agree to take turns to be “general” who gets to say which transaction should happen next. They coordinate this with the use of a counter, and agree that one person will validate when `counter mod 2` is 0 and the other when `counter mod 2` is 1. The person who is not the “general” at a turn can propose transactions to the “general”. Each person stores the instruction they last validated in permanent storage until they receive the next rounds validated transaction (thus continuity is guaranteed.) In permanent storage you thus maintain a turn bit (0 or 1) for `counter mod turnbit`, and a turn counter for same operation, and the last validated instruction (an instruction being a command with arguments). The “state transition function” in Lockstep includes incrementing the turn counter and setting the last validated instruction, as well as the state changes that the instruction performed, and is “atomic”, all-or-nothing. Thus, two accounts are in perfect agreement over every decision that they make.

Integrating Lockstep with the role as an intermediary

With single-hop guaranteed, two-hop at an intermediary is guaranteed since it is on a single machine. But it still has to be organized practically, on that machine. A simple approach is that the command handler returns an “inter-lockstep” callback. This callback is executed only if the Lockstep state transition succeeded. Since such callback could fail, it is ideally combined with “failsafe” routines. For example, a callback that queues a transaction could fail because the queue is full. A failsafe routine can derive the same decision from the permanently stored data, and run periodically to catch any failures.

Guaranteed single-hop consensus makes multi-hop consensus possible

Lockstep guarantees that any signal that propagates over multiple hops has been agreed on by every previous hop. But it does not guarantee that a signal propagates, nor is it able to discover propagation failure. Thus, multi-hop agreements have to conform to the fact that propagated signals can be trusted but signal propagation cannot. The fundamentals are: assume perfect consensus about that propagated signals are honest, assume zero consensus about if a signal has propagated (i.e., the receiver can be certain, the sender cannot). It follows that only the end receiver of a signal can know if an agreement succeeded.

Multi-hop payment with honest signals but uncertain signal propagation

With honest signals but uncertain signal propagation, we can design the steps for perfectly secure multi-hop payments. For the important parts of the payment, we rely on that the end receiver of a signal can be certain that everyone in the chain has agreed. For our multi-hop payments, we need a few fundamental rules. The first rule, is that the buyer will be the last person with the right to cancel. I.e., *the sender has no certainty*, and the buyer is the only sender who takes on a net negative balance. The second rule, is that everyone else can only cancel if their state did not allow them to propagate a signal futher (i.e., their payment request had timed out). Thus they can refuse to accept a signal, but they cannot "take back" their decision if they already propagated it. I.e., *the receiver requires perfect certainty.* And the third rule, since the buyer is the last person with the right to cancel, and, the only one with a net negative balance, the buyer will be the "end receiver" of whether or not the payment was agreed on.

The first two (and most important) steps then become (assuming a path has already been found and credit been set aside with a timeout):

1. The buyer sends a "commit" signal (over Lockstep). On "commit" credit is set aside without a timeout and in permanent storage (the timeout of the previous temporary commit has to be considered, this can be done in one of three ways.)

2. When the seller receives the "commit" signal, they reply directly to the buyer (over their direct channel that does not use "Lockstep") and tell them to finalize.

Thus, for "commit" the signal travels in a straight line from the buyer and (via direct channel from seller) back to the buyer. The buyer becomes the end receiver of the signal, and has perfect certainty that the agreement succeeded. This is the only signal where a decision has to be made based on a multi-hop consensus.

The next step, does not lead to any further decisions. Thus, no single person needs to be told if it succeeded to reach everyon in the chain or not. It can propagate organically.

3. When the buyer receives the reply, they revoke their right to cancel (and at this point the payment is already set in stone), and they send the "finalize" signal.

The buyer is then done. They delete their pending credit line (in "finalize" command handler that runs over Lockstep) and add the amount to their actual credit line (this can potentially be done already at step 1). And they are done. Had they instead cancelled, they would have sent a "cancel" signal, executed the "cancel" command handler, removed their pending credit line, and been done. In either case, the responsibility to propagate falls on the next person in the chain (who if the buyer did "cancel" is now the only one with a net negative balance.)