Ripple Inter Server Protocol

Johan Nygren, johan@ripple.archi

Ripple is a money system invented by Ryan Fugger in 2003/2004. It takes banking down to the smallest possible level, person-to-person relationships, and routes payment via multiple hops using such “people-banks” as intermediaries. Ripple was originally implemented with RipplePay, a single-server platform, and Fugger envisioned a multi-server system as the next logical step: Ripple Inter Server Protocol (RISP). Such a system was never developed. Instead, global consensus platforms similar to those pioneered by Satoshi Nakamoto (Craig Wright) were used for the next step, with Ripple.com. The reason RISP was never developed is because a fundamental component was ignored: a consensus mechanism at the account level (a person-to-person relationship). Instead, consistency tried to be shoehorned into the account level, using the same type of mechanisms that were used for multi-hop interaction. But account consensus is a different problem, and requires a different solution. The same type of solution Satoshi pioneered, and the same type of solution ripple.com used (although at an unnecessary large scale): validator alternation so a state can be distributed over multiple actors, in this case, two users.

Payment finality, from seller to buyer or buyer to seller?

Ripple is a system where agreements have to happen at two levels: both in direct person-to-person relationships, and also over multiple-hops. These are fundamentally different types of problems, that requires fundamentally different types of solutions. What happened with Ripple Inter Server Protocol from the early 2000s and onwards, is that the latter problem (multi-hop) was understood (it is also more conventional in web development), but the former problem was neglected. Since it was neglected, it was never possible to be certain about the state (because of the two general problem). Thus, avoiding agreement and maintaining uncertainty, was favoured as a design.

For the multi-hop coordination, social incentives were meant to compensate for the lack of certainty. During payment routing, anyone who has incurred a net negative balance and not yet received the payment from the previous hop, would have an incentive to forward the payment signal. Thus with timeouts that were incrementally longer the further away and account was from the seller, the signal would propagate - because of social incentives - all the way, in an “all-or-nothing” payment finalization. If the payment failed, there was no way to resolve it, it was assumed that social incentives would ensure it did not fail.

The logical way to do multi-hop payments, is to rely on the social incentive of the person who has a constant net negative in the payment: the buyer. The buyer, then sends an “all-or-nothing” payment commit, but one that the buyer can always cancel if it were to fail. Any intermediary that fails to propagate this commit (because their “temporary commit” had timed out), can formally cancel it, and return a cancel signal. Once an intermediary has passed on the signal, they are not allowed to cancel anymore. If the commit succeeds to reach the seller, they will signal the buyer, and the buyer will revoke their right to cancel, and then send a “finalize” signal where everyone writes their pending payment to their balance.

Notice that the second scenario, required more coordination steps, that each assume certainty about the state between each person-to-person relationship. It requires an “all-or-nothing” commit that can still be formally cancelled from the buyer or the “front” of the signal. It requires a pending payment object in the state, that gets stored indefinitely until the payment finality signal is received. That coordination requires certainty during signal propagation, that each “hop” can be certain it succeeded, ensured by the account consensus mechanism. Lacking an account consensus mechanism, a “social incentive” could - *as a story* - compensate for that lack of certainty, and since if you send the “all-or-nothing” from the seller to the buyer you would have a social incentive for each intermediary to propagate the signal, you could appeal to this as the solution - *despite that such a solution too would require an account consensus mechanism.* And since this solution is inherently flawed - a failed propagation means actual wrong balances - you need to extend it with a “global commit mechanism”, and since such a mechanism itself could have faults, you need to use a very reliable one and you end up with something similar to Nakamoto consensus ledgers.

Thus the focus of Ripple shifted towards resolving problems that existed because the two general problem was not solved (there was no account consensus mechanism), and the “tool” to solve this became a massively-oversized consensus mechanism, global public ledgers. And from there, Ripple.com was a natural next step.

Payment finality, from the buyer to the seller

Multi-hop payments are simple, as long as each hop can be certain about what they agreed on. The two-general problem means such uncertainty can not be achieved by communication alone (by a web developed mindset) but you need a formal consensus mechanism. Such mechanisms rely on agreeing on a single general, potentially with alternation of who is the general. They have been understood for at least 30-40 years (maybe longer I am not educated on it), and made more popular with Bitcoin and later Ethereum. For RISP, you simply let the two users of an account take turns to validate, one after the other, decided by a “turn bit” (0 or 1) and a “turn counter” on which you do modulo 2.

The payment is easily coordinated by relying on that the buyer has a social incentive to ensure the payment goes through. Thus, they become the “general” so to speak, and decisions are made by them. When the buyer finds a path, they request everyone to temporarily commit funds to it. Then once they have been signalled by the seller that this is complete (the signal reached the seller via all intermediaries from the buyer), they request everyone to permanently commit funds. This can fail at any hop (if the temporary commit timed out), but anyone who has passed on the signal revokes their right to initiate a cancel. Thus once the seller signals the buyer that everyone has permanently committed, the buyer remains the only one who can cancel. The buyer then revokes their right to cancel, and sends a “finalize” signal that tells everyone to write their pending payment to their balance. With this there can never be any wrong balances. The worst case scenario is that there can be “garbage collection”-type problems. If an intermediary segment gets “cut off”, they would be left with pending payment objects that they cannot clean up easily. But this is not a system breaking problem, it is manageable.

Notice that with payment finality from the buyer to the seller, there is never any “race against time” such as in the approach that previous attempts at RISP have had. Thus, there is no major damage made if there is any faults in the network. Thus, there is no need to introduce a “global commit mechanism”. All this, because you can trust the state at each account to be correct, because the users have resolved the two-general problem with the use of a consensus mechanism, agreeing on a single general.

What motivated from seller to buyer finalize idea

There is a “fake recipient” attack vector that requires the recipient (seller) to signal to the buyer that they are the one the path is made to. This signal can be sent either via the intermediaries in the form of a “secret message” (such as a preimage to a hash), or directly via a private channel (my RISP does the latter). Thus, there was reason to consider such “handshake”.

Traditionally, the recipient of an IOU is authoritative. Thus it was easy to assume that for Ripple, the two-general problem was resolved by that the recipient was inherently the general. While this is not true, Ripple needs bilateral authority for account balance, it could favour the idea of letting the “final signal” go backwards from seller to buyer.

The backwards signal also provided a story where the social incentive could ensure propagation (thus such story served as an imagined solution to the two general problem).

These all converged into that, with the backwards “race against time” - a “global commit mechanism” could be used to prevent the problem that such backwards propagation could fail. And, such mechanism also then gave a central general for account states, which… solved the two-general problem, but in an extremely overscaled way.

All Ripple needs is that two users who have an account take turns validating decisions, one after the other.