Consensus

From Ripple Wiki

Contents

- 1 Overview
  - 1.1 Ledgers
  - 1.2 Not colluding
  - 1.3 Unique node lists
  - 1.4 Selecting validators
  - 1.5 Common consensus
  - 1.6 Liars
  - 1.7 Split personalities
  - 1.8 Consensus scalability
  - 1.9 Conclusion
- 2 Technical Description
  - 2.1 Overview
  - 2.2 More Details
  - 2.3 Scaling
  - 2.4 mini-FAQ
    - 2.4.1 What happens if say 33% of the nodes saw transaction A, 33% saw transaction B, and 34% saw transaction C first where all 3 conflict?
    - 2.4.2 How does the deterministic algorithm for including transactions in the ledger work?

Overview

Consensus is the process by which the entire network agrees on the same Ledger. It is what keeps everybody on the same page.

A walk through of how the consensus process works:

- White Paper (http://dev.ripple.com/consensus-whitepaper.html)
- Graphic
- Video (http://youtu.be/pj1QVb1vlC0)

Ledgers

A **ledger** is a snapshot of the state of the system. It includes everyone's balances and trust limits. The ledger is arranged as a Hash Tree such that it can be summed up with a single number.

Not colluding

Instead of trusting random validators, we choose specific validators who we "trust to not collude to defraud us". This is a far weaker criteria than "trusting" someone not to lie.

Imagine, we believe Wile E. Coyote (http://looneytunes.wikia.com /wiki/Wile_E._Coyote) and the Road-Runner (http://looneytunes.wikia.com /wiki/The_Road-Runner) are rivals. We believe that they would never collude in general and specifically that they would never collude to defraud us as they are practically at war with each other. If both of them attest to something, then it is very likely true. This illustrates that while they may be totally untrustworthy, we can rely on them not colluding to defraud us so that we can conclude that something is very likely to be true.

Unique node lists

The validators run Ripple nodes which process validations on their behalf. The list of validators that we trust not to collude to defraud us is called our Unique Node List (UNL).

For stronger assurance that something is true, we add more validators to our UNL. For example, we could add Itchy and Scratchy (http://en.wikipedia.org /wiki/The_Itchy_%26_Scratchy_Show). Who are also not very trustworthy, but are also unlikely to collude to defraud us.

Selecting validators

In real world terms, people should select a UNL list of 1,000 validators. They should choose 200 validators from 5 different continents. They should choose a mix of validators with different interests: merchants, financial firms, non-profits, political parties, religious groups, etc... By choosing a large number of reputable parties who are unlikely to lie or be coerced as a group and that are unlikely to collude to defraud us we can be assured the ledger is accurate.

In practice, most people will use the default UNL supplied by their client. But, the software will enable them to choose specific validators if they'd like.

Common consensus

If everyone chooses a completely disparate sets of validators the network will be unlikely to reach consensus that a particular version of the ledger is the one true and accurate ledger. But, in practice, people's UNL lists will overlap. This overlap causes the honest validators to come to the same consensus.

Every honest user of the system wants the system to reach a consensus. Validators will choose which other validators they trust specifically because they also wish to reach a consensus. Essentially, all honest users of the system cooperate to ensure a consensus is reached and maintained. And, of course, a lack of consensus is easily detectable.

Liars

If someone should be dishonest, the honest actors will notice them lying and can disregard their future attestations. That is, if you lie once, you have no gain as the network doesn't care and they won't care what you say in the future.

The ripple protocol requires validators to sign the consensus ledger after each validation interval. If a validator fails to do this, it will be easily detectable. Similarly, if a validator signs a ledger and then signs a ledger after that which cannot be reached by a combination of valid transactions, this will also be easily detectable.

Split personalities

However, someone say Sybil (http://en.wikipedia.org/wiki/Sybil_attack), might pretend to be a million nodes. Sybil hopes someone will choose to trust Sybil's nodes not to collude and then Sybil can lie. Ripple is immune to this type of attack because people will avoid choosing anonymous nodes to prevent this problem. People will only choose entities as validators who are not anonymous or have a reputation. Even if they should accidentally, choose a few of Sybil's nodes those nodes will not be in the majority and won't have significant impact.

In addition, there's no way to cause any actual harm without your actions being obvious. If a transaction is fully valid and was seen prior to the consensus window, there is no reason not to vote yes on that transaction. If a transaction set is accepted as the majority, there is no reason not to validate the ledger that transaction set generates. The worst damage we can foresee from such an attack is making the network unusable. Should that happen, all honest users of the system will determine which nodes caused the problem and stop trusting them. The attacker is now back to square one.

Consensus scalability

The consensus process works for Ripple with only a few validators. When the ripple network is starting there will be very few validators. However, those few validators have a very vested interest in the ripple network succeeding. Therefore, they are very unlikely to collude with the other validators to defraud someone or undermine the network. As soon as they do, they would be caught and ignored thereafter.

Adding more validators to the network only strengthens the assurance that the network is not going to defraud anyone.

Conclusion

Finding consensus by trusting validators not to collude is a simple and strong way to establish the valid ledger. Because, the network is not relying on proof work, this consensus process is fast and allows the Ripple network to validate ledgers in seconds.

Technical Description

Overview

The goal of the consensus process is to ensure all nodes agree on the same ledger. Ledgers are snapshots of everyone's balances and offers at a particular point in time. You can form a ledger by taking the previous ledger and applying all the transactions that have happened since then. So to agree on the current ledger, nodes must agree on the previous ledger and the set of transactions that have happened since then.

Each participant in the ripple network has a list of validators. This list is also known as a unique node list UNL. Every validating node that is online will see all transactions. Periodically all nodes will try to **validate** or **close** a new ledger. Each node will broadcast what Transaction set it thinks should be applied to the previous ledger. If a node sees that the majority of its UNL supports one set of transactions it will switch to this new transaction set.

This process is iterative. A node will propose a certain transaction set and then if it sees the majority of its UNL supports a different set it will switch and propose the set backed by the majority. After a few iterations nodes will converge on the same transaction set. What transaction set is picked is irrelevant. All that is important is that everyone agrees. Once a transaction set is chosen it is applied to the previous ledger. This is a deterministic process so everyone that started with the same previous ledger and transaction set will get the same result.

If some valid transaction was voted out of the currently closing ledger, it will be

inserted into the next closing ledger if it still valid. A new ledger is closed approximately every 5 seconds. So you only need to wait 5-10 seconds for your transaction to be confirmed by the network.

Some people wonder if this can lead to many different versions of the ledger since nodes each have a different UNL. As long as there is some minimal degree of inter-connectivity between UNLs consensus will rapidly be reached. This is primarily because every honest node's primary goal is to achieve a consensus.

In addition, if a transaction is announced early in the consensus process and is fully valid, all honest nodes will agree that it should go in the next set. If the transaction is received late or possibly conflicts with another transaction, no honest node particularly cares if it gets in the consensus set or not -- if the transaction is in fact valid, it will get in the next consensus set anyway since all honest nodes will agree it should. What they do value is consensus.

More Details

The consensus process occurs in multiple rounds for each ledger.

Each Node that participates in the algorithm proposes a Transaction Set (by a single hash of the root of its tree).

Nodes acquire any transaction sets proposed by nodes they trust. (Which will mean only faulting in the differences thanks to the tree structure.)

If a node doesn't know about a transaction, it implicitly votes no on it by not including it in any tree it proposes. This allows the system to handle overload sensibly -- if a transaction is not seen or processed by enough validators, the majority will implicitly vote no on it, the remainder will switch their votes to no in the consensus process, and it will be deferred.

The basic rule is that if 50% of the nodes on your UNL, including you, vote for a transaction, you include it. If not, you don't. After a few seconds, the threshold raises from 50% to 60% -- failure to agree is agreement to fail -- and continues to rise. This ensures that the voting on a transaction doesn't just bounce around 50% for an extended period of time.

Two important things to understand are: 1) Any legitimate transaction, if seen before the round begins, should get a super-majority. 2) Any transaction voted out (but legal) will get voted in by every honest node in the next ledger. The exception to 1 would be if the transaction came in right at ledger close time. In which case, we don't care whether it gets voted in or not, both results are "correct". They are both correct since either the transaction makes it in this ledger or the next one.

Now this agreed set of transactions is just a candidate set. Nodes vote yes if they

think it should go in the ledger, but it could (in theory) still fail to apply to the ledger.

There is no harm if two conflicting transactions are both voted into a consensus set. The process of applying the transactions is deterministic and at most one will succeed. Since the conflicting transaction is no longer valid, it will never get voted into another set.

The process of recovering transactions not voted into the consensus set is also deterministic. So if there are two conflicting transactions and neither garners a majority, one will be picked by the vast majority of honest nodes in the next consensus window and it will get voted into the next consensus transaction set.

The timing scheme chosen for the ripple network is called Continuous Ledger Close.

Scaling

There are several things that help with scaling. The fact that synchronizing is done by transaction trees means you only have to send small updates.

You only need to check each transaction once. If a node spits invalid transactions at you, you can punish it by disconnecting it and demanding proof of work when it reconnects. No important node will send you invalid transactions. Nobody can make an honest node send invalid data.

The rule is that you forward a transaction only if you believe it could claim a transaction fee. This prevents the network from getting clogged with transactions that cannot possibly succeed.

The main way we deal with a DoS attack of legitimate-looking transactions is by raising transaction fees. If 51% of the network requires a specific transaction fee, you won't get a consensus if you don't pay it.

Nodes deal with purely local load, say someone spitting invalid transactions at them, by disconnected possibly abusive nodes, demanding proof of work from borderline abusive nodes, and by raising the transaction fees they demand to forward transactions. Validators also publish transaction fee levels in their validations to apply backpressure in the event that more valid transactions are being sent than they can process.

If a validator cannot keep up with the load of transactions being voted into the consensus set, it will switch to issuing partial validations. These let those who trust it know that it is not split off from the network (and potentially validating other ledgers). In addition, validators will raise the transaction fees they demand to prevent the network from shrinking to a small set of "super nodes", as that

would increase the risk of collusion.

mini-FAQ

What happens if say 33% of the nodes saw transaction A, 33% saw transaction B, and 34% saw transaction C first where all 3 conflict?

Then no transaction will get voted in, and the deterministic algorithm will pick only one which will be a candidate for the next ledger.

How does the deterministic algorithm for including transactions in the ledger work?

Basically it applies transactions in an order designed for maximum efficiency until no new transactions get in. Transactions can pass, hard fail, or soft fail. If they pass, they're included. If they hard fail, they're dropped. if the soft fail, they stay as candidates.

Once a transaction gets in, all that conflict with it will hard fail.

The current algorithm is hash order first, with any soft fails repeated in account/sequence order. When no new transactions succeed in a pass, the operation completes.
