On Worldwide Consensus
The future is distributed: a summary of the Stellar Consensus Protocol white paper

Read the full white paper and explore more at Stellar Galaxy.

Financial infrastructure is currently a mess of closed systems. Gaps between these systems mean that transaction costs are high and money moves slowly across political and geographic boundaries. This friction has slowed the growth of financial services, leaving billions of people underserved.
To solve these problems, can we build financial infrastructure that supports the kind of organic growth and innovation we’ve seen from the Internet, yet still ensures that financial transactions are recorded accurately? A decentralized worldwide financial network could remove barriers to entry, allowing new, innovative participants—who may possess only modest financial and computing resources—to become part of the network’s infrastructure and extend access to unserved communities.

A network with low barrier to entry will spur organic growth, but it also means no longer relying solely on established financial institutions to record transactions accurately. Rather, all participants ensure accuracy by agreeing on the validity of one another’s transactions. This agreement hinges on a mechanism to reach worldwide consensus.

We introduce the Stellar Consensus Protocol (SCP), a model suitable for worldwide consensus. SCP is the first provably safe consensus mechanism that simultaneously enjoys four key properties: decentralized control, low latency, flexible trust, and asymptotic security.

<table>
<thead>
<tr>
<th>MECHANISM</th>
<th>DECENTRALIZED CONTROL</th>
<th>LOW LATENCY</th>
<th>FLEXIBLE TRUST</th>
<th>ASYMPTOTIC SECURITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proof of work</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proof of stake</td>
<td>✓</td>
<td>maybe</td>
<td>maybe</td>
<td></td>
</tr>
<tr>
<td>Byzantine agreement</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Tendermint</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Stellar Consensus Protocol</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

The main approaches to consensus enjoy at most three out of the four key properties enjoyed by FBA.

**The Model: Federated Byzantine Agreement**
SCP is a construction for federated Byzantine agreement, a new approach to consensus.

**What is distributed consensus?**

In a distributed system, all nodes must periodically update the state that they're replicating—for example, a transaction ledger. We identify each update by a unique **slot**; a consensus protocol ensures that all nodes agree upon slot content.

When nodes decide that an update is safe to apply, they **externalize**, or publish, the agreed-upon statement to their replica of the ledger. **Consensus** is reached when all nodes update their ledgers and externalize the same value.

**Tolerating Byzantine failure**

We want to ensure consensus even when individual nodes act arbitrarily, behavior known as **Byzantine failure**. To tolerate Byzantine failure, SCP is designed not to require unanimous consent from the complete set of nodes for the system to reach agreement, and to tolerate nodes that lie or send incorrect messages.

**Quorum slices: introducing federation**

In a distributed system, a **quorum** is a set of nodes sufficient to reach agreement. Federated Byzantine agreement introduces the concept of a **quorum slice**, the subset of a quorum that can convince one particular node of agreement.
There are several important differences between traditional non-federated Byzantine agreement and federated Byzantine agreement. Byzantine agreement guarantees distributed consensus despite the Byzantine failure of participants. However, it requires unanimous agreement on system membership by all participants. Each node in the network must be known and verified ahead of time.

FBA brings open membership and decentralized control to Byzantine agreement. The key difference between a Byzantine agreement system and a federated Byzantine agreement system (FBAS) is that in FBA each node chooses its own quorum slices. The system-wide quorums result from these decisions by individual nodes.

In FBA, there’s no gatekeeper and no centralized authority, so individual nodes decide which other participants they trust for information. Nodes can have multiple slices, and these individual node choices may be based on extrinsic criteria. For example, a particular bank might be viewed as reputable, causing other nodes to require its acknowledgment of all transactions; a company might already have a financial relationship with a credit union and want to make sure it and the bank both sign off on all transactions.

Good quorum share nodes and lead to quorums that overlap. We call this overlap quorum intersection. When quorums don’t intersect, we end up with disjoint quorums. If quorums are disjoint, quorum A can, for example, agree on a statement to order pizza, while quorum B can agree on a statement to order hamburgers. Because they can independently agree on contradictory statements, disjoint quorums can undermine consensus.
Making good choices

Each node is responsible for ensuring that its choice of quorum slice doesn't violate quorum intersection. Making a responsible choice generally boils down to ensuring slices are large enough and that the nodes they contain are important enough not to risk their reputations by lying and feeding different information to different people.

Problems on the way to agreement

In slice selection, nodes must maintain a balance between safety and liveness. We want the system to be responsive, but not at the expense of correctness.

- Nodes lack liveness when they get blocked on the way to agreement, slowing down the system.
- Nodes lack safety when they externalize values inconsistent with those externalized at other nodes, undermining system-wide agreement. Such nodes are divergent.
- A divergent state occurs when ledgers held by different nodes store contradictory, irreconcilable states. A blocked system is less dangerous than a divergent one.

Federated Voting: Vote, Accept, Confirm

FBAS nodes use a federated voting technique to get to agreement.
**Lunchtime consensus**

To describe in more detail the process through which a node votes for and eventually accepts a statement, allowing a system to reach agreement, we're going to use an example familiar to many. Let's say that a group of people is voting on what to order for lunch. In this example, all people names are nodes and all food options are statements that the nodes need to consider.

Vanessa works in a public coworking space where it’s optional to order lunch with a larger group. There are a variety of options, and not everyone needs to pick something; when enough of the group has decided, they place an order.

Vanessa and the coworkers decide to solve this problem using SCP.

**Initial voting**

Let’s say that Vanessa wants pizza but needs to remain open in case a large part of the group votes for one of the other options.

Voting is preliminary and only happens on the node level. In the first step of the federated voting process, Vanessa votes to remain open to the possibility of accepting pizza by asserting that pizza is valid and promising she has never and will never individually vote for any option contradicting pizza. She may,
however, end up accepting something other than pizza if enough of her coworkers vote otherwise (peer pressure!).

**Acceptance**

Thanks to quorum intersection, slices influence one another. Imagine an alternate path (indicated in the diagram by “voted hamburger”), in which Vanessa initially votes for hamburgers. But remember, votes are only preliminary.

If Winnie, Andrew, and Eva are in quorum slices with Vanessa, they can block the progress of accepting hamburgers. A \textit{v-blocking} set of nodes contains at least one node from each of Vanessa’s slices and can block action in all quorums that contain Vanessa, causing Vanessa to accept pizza instead.

Vanessa actually \textbf{accepts} pizza if:

1) she has never accepted a statement contradicting pizza (for example, burritos), and

2) each member of a v-blocking set claims to accept pizza, or each member of a quorum including Vanessa either votes for or claims to accept pizza

**Ratification**

When every member of a quorum votes for pizza, we say that the quorum \textbf{ratifies} pizza. A node doesn’t need to ratify a statement firsthand.

For example, Scott often relies on Andrew and Iris to decide what to eat. They are his quorum. If they all three vote for pizza, the quorum has ratified pizza.

A coworker can vote for one lunch option and later accept a contradictory one. Voting for pizza doesn’t assert pizza as lunch—pizza will be accepted as lunch only if it’s ratified.

**Confirmation messages**

Confirmation is the final step of the voting process and implies system-wide agreement.

To ensure agreement, nodes exchange confirmation messages. A system \textbf{agrees} on a statement if, once sufficient messages are delivered and processed, and no matter what events subsequently transpire, every responsive, accurate node will accept the statement.
Eva, for example, can assert that she’s accepted pizza by sending the confirmation message, “accept(pizza).” This message is an abbreviation for “I have accepted pizza.”

When Eva sends this confirmation message, Winnie, Andrew, and Graydon, the other people in her quorum, broadcast “accept(pizza).”

These messages may convince additional people to accept pizza. As in the acceptance example above, if Vanessa initially votes for an option that contradicts pizza, such as hamburgers, she can still accept pizza if a v-blocking set accepts it. These additional people convince as many others as possible, broadcasting “accept(pizza)” until a point when everyone who can accept pizza has accepted it.

A subsequent quorum of confirmation messages allows Vanessa to confirm pizza, which implies the system agrees on pizza. The company orders pizza. Everyone is happy.

**The Protocol: SCP**

SCP takes on the main challenge of distributed consensus: the system can’t agree on a statement without the risk of getting blocked and losing liveness.

It’s possible for a statement to get stuck in a permanently indeterminate state before the system reaches agreement on it. The goal of SCP is to minimize such blocking and the potential for divergence. The protocol therefore crafts statements so that it’s possible to neutralize blocked statements if they get stuck in the voting process—all this magic is rooted in a ballot-based approach to the problem.

A **ballot** is a referendum on the value associated with the ballot, i.e., “What is the value that we’re voting on?” The ballot-based approach means that, to eventually externalize a value, a node must commit the ballot tied to that value.

**Commit or abort**

Each node may either commit or abort any ballot. To return to the lunch order analogy, the group of coworkers could get stuck, with the group unable to come to agreement. We need a way to neutralize an option—say, hamburgers—if the team gets stuck on it so they can make progress and eventually place an order.
To neutralize hamburgers as an option, coworkers accept “abort hamburgers,” hamburgers become irrelevant, and the group moves on to vote on another lunch option.

If, on the other hand, coworkers ratify the statement “commit pizza,” then the group will agree on the value, pizza, tied to that ballot. A statement “commit pizza” is valid, and therefore can legitimately appear in votes, only if all incompatible ballots less than pizza have been aborted.

In an SCP system with quorum intersection, there are no blocked states for intact nodes. There’s always a sequence of events through which intact nodes can reach agreement on and commit a value.

If nodes depend too heavily on bad nodes, they’re called befouled. In an FBAS, befouled nodes form a dispensable set, which means intact nodes can ratify statements without the cooperation of befouled nodes and befouled nodes cannot undermine agreement among intact nodes. If no intact node has voted to commit any ballot, they can then move on to any ballot higher than any they’ve pledged to abort. Lack of response from befouled nodes won’t block intact nodes from assembling quorums and making progress.

The protocol proves that we can reach agreement and order lunch, but what happens if a bunch of people vote for different things and no quorum ratifies any choice? This question—a split vote—is precisely why we need to associate each lunch option with a ballot. The ballot process, including how to deal with scenarios like split votes, is intricate and contains details not described here. For a fuller description, as well as theorems and proofs, read the white paper.

This summary covers the major points and terminology of “The Stellar Consensus Protocol: A Federated Model for Internet-Level Consensus,” a white paper by Prof. David Mazières, Chief Scientist at the Stellar Development Foundation.

Questions? Ask them on StackExchange.

Still too complicated? Want pictures? Read the first chapter of our graphic novel, Adventures in Galactic Consensus.