**Paxos Overview**
There are 4 traditional roles in Paxos, the Coordinator, Proposer, Acceptor and Learner.

There are also 4 phases, Prepare, Promise, Accept and Accepted.

Let's look at an analogy. Imagine you and your friends are deciding what to eat.
You are a proposer, and your friends are the acceptors.

First, you prepare your friends by calling out to them and getting their attention.
Naturally, your friends will pay attention to the last person to call out to them, you, and promise to ignore anyone before you.
You can then propose to eat Pizza and ask your friends to accept your proposal.
Some of your friends will agree on Pizza and have accepted your proposal.
If a majority of your friends agree on Pizza, then you will all eat Pizza.

**Our Classic Paxos Model**
In our Classic Paxos model, we have 2 explicit roles, Proposers and Acceptors.
Because we want the simplest representation of Paxos, our proposers are also coordinator, and acceptors are also learners. A quorum is defined as n/2+1, where n is the number of acceptors. In other words, a simple majority.

**Phase 1a**
In the prepare phase, you want to get the attention of your friends, so you call out to them. And if you're the latest person to call out to them, you will have their attention.

The proposer sends a Prepare message, which consists of a ballot number. If this ballot number is greater than any ballot number an acceptor has seen, it will reply to it.

Some implementations may include the proposer in the message too, but we chose not to because it is not needed. Because while it is possible for 2 proposers to use the same ballot number for their Prepare message, at most 1 proposer will get a quorum of replies.

In other words, 2 of you may try to get the attention of your friends at the same time, but only one of you can get the majority to pay attention to you.

We also allow the model to send a Prepare message with any ballot number at anytime. This models the asynchronous nature of the system where messages may be delayed and received out of order.

**Phase 1b**
If you are the last person to call out to your friend, they will pay attention to you. And if they have accepted any previous proposals, they will let you know the last one they accepted.

If an acceptor receives a Prepare message with a ballot greater than any it has seen, it replies with a Promise message. If the acceptor has accepted a proposed value before, attach that value and its associated ballot in the message too.

**Phase 2a**
If you have the attention of a majority of your friends, you may now propose what to eat. If nobody has accepted any proposals before, you can come up with a new one. Otherwise, you must proposed the same thing as the most recent one.

If a proposer receives a quorum of Promise replies for a ballot, broadcast an Accept message with that ballot and a value. If all acceptors has not accepted any values before, the proposer may propose any value. Otherwise, it will propose the value with the highest ballot number in the quorum of Promise replies it received.

**Phase 2b**
If your proposal is the latest one, your friend will agree with you and tell everyone else.

If an acceptor receives an Accept message with a ballot greater or equal to any ballot it has previously seen, it will accept that value, and broadcast an Accepted message containing the ballot number, itself and the value.

**Decide**
If your friend sees that a majority of them agree on the same thing, they will decide on that.

If an acceptor receives a quorum of Accepted messages for a ballot, it will decide on that value.

We have 2 safety properties to uphold.
Non-triviality: Only proposed values can be learnt.
Consistency: At most 1 value can be learnt.

That's the end of Paxos, so now we will look at Fast Paxos.

**Fast Paxos Overview**
In this case, if you already have the exclusive attention of your friends, you don't have to call out to them. You just ask all your friends to propose what to eat, they propose something, and you will make the final decision.

We introduce 2 new phases, Fast Accept and Fast Accepted, and we remove Prepare and Promise phase.

There are 2 possible scenarios, if a quorum of acceptors propose the same thing, there's no collision, and this protocol is faster.

If there is a collision, we have to fall back to using Classic Paxos, so it may be potentially slower.

**Our Fast Paxos Model**
For our Fast Paxos model, we have 2 explicit roles, Coordinator and Acceptors.
There is only 1 Coordinator. Acceptors are also Proposers and Learners.
We use the same messages as before, and we do not have Prepare and Promise phases.
Since there is only 1 coordinator in the system, we know that nobody else is sending a Accept message.
A fast quorum is defined as 3n/4. So whereas previously we only need a simple majority, we now need three quarters of the acceptors.

Same messages as before, but we don't need phase 1a and 1b. That's basically the getting their attention phases.

Here our quorum is 3n/4. Instead of just a simple majority, we now need 3 quarters of the acceptors to be considered a majority.

**Phase 2a (Fast)**

If no values was proposed previously, the coordinator can broadcast a Fast Accept message, with a ballot number and value of “any”.

**Phase 2b (Fast)**
If an acceptor receives a Fast Accept message with a ballot number greater or equal to any it has seen before, it can propose any value, and broadcast a Fast Accepted message with that ballot and value.

**Decide (Fast)**
If an Acceptor receives a fast quorum of Fast Accepted messages for a ballot of a fast round, of one value, it decides on that value. Because that means the majority has already accepted that value. This is the case of no collision.

**Phase 2a (Classic)**
So what if more than one value appears in the fast quorum? We will have the coordinator decide.

Upon receiving a fast quorum of Fast Accepted messages from acceptors, more than 1 value was proposed. Collision is resolved via the following rules:
1. The coordinator will select a value if it is the majority in the fast quorum.
2. If there is no majority, the coordinator is free to select any proposed value.

The system will then fall back to the Classic Paxos protocol we mentioned earlier, starting with the Accept phase, by sending an Accept message with a higher ballot number, with the selected value.

Then the Classic Paxos Accepted phase we mentioned earlier will happen, and this value will eventually be learnt.

**FLP Impossibility Theorem vs Paxos**
Quick note, the FLP Impossibility Theorem states that in an asynchronous network where messages may be delayed but not lost, there is no consensus algorithm that is guaranteed to terminate in every execution for all starting conditions, if at least one node may experience failure.

But Paxos works because it does not guarantee termination. In practice, we tend to ensure that it terminates using things like exponential timeouts, but in the protocol itself, it is easy to come up with a scenario where no proposers are able to receive a majority of Accepted messages.