Raft

Server

Client who sends info to server

There are multiple nodes which can be in 3 states: Follower, Candidate, Leader state

All nodes start in follower state, each with its (Variable: node name, and Term: 0)

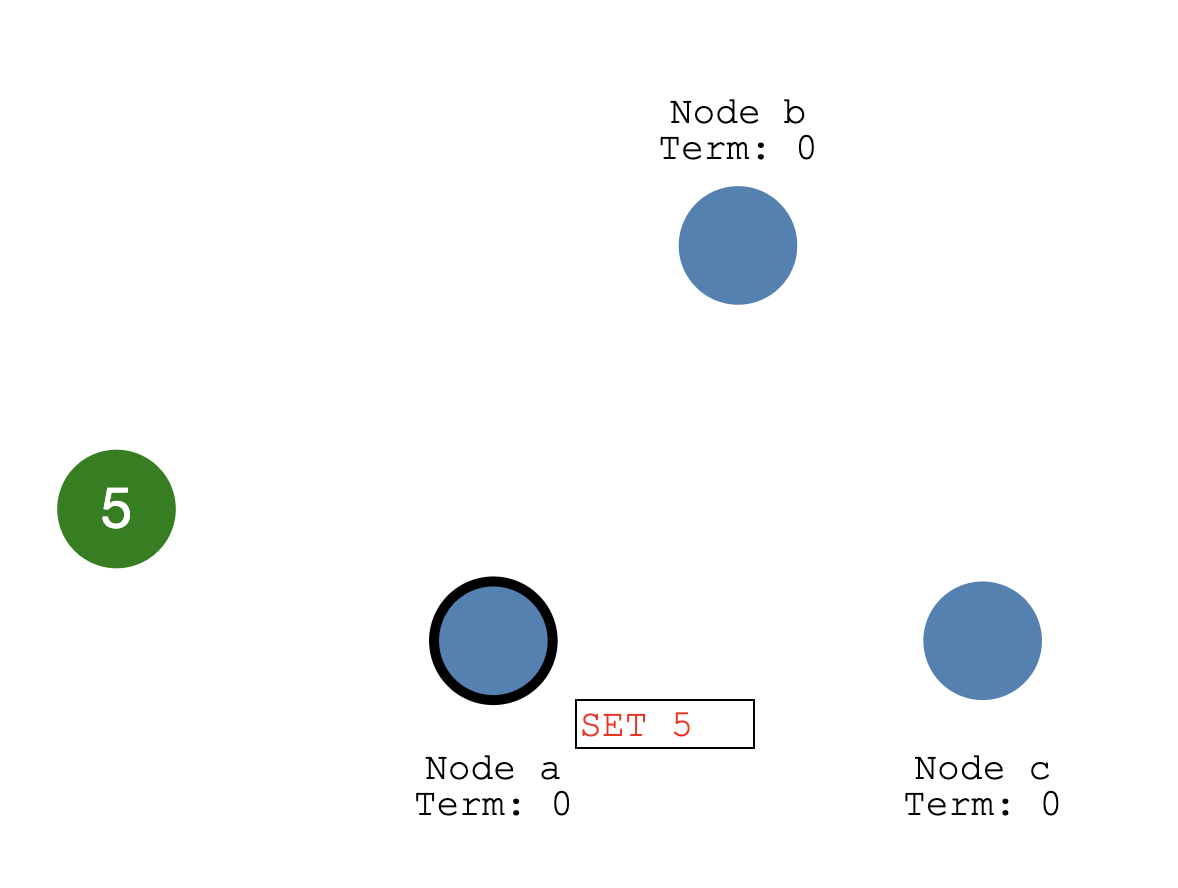
If followers did not hear from a leader 🡪 become a candidate state (Variable add: Vote Count: null)

The candidate then requests **votes** from other nodes. (send requests to other nodes who may in follower state rather than candidate state)

Other nodes will reply (with the **vote**) to the candidate node who asks the request

The candidate becomes the leader node if it receives votes from majority of nodes (The process of **Leader Election**) And all changes to the system now go through the leader.

Each change is added as an entry in the **node's log**.



This log entry is currently uncommitted so it won't update the node's value.

A screenshot of a computer program

Description automatically generated

To commit the entry the node first replicates it to the follower nodes...

then the leader waits until a majority of nodes have written the entry.

The entry is now committed on the leader node and the node state is "5".

The leader then notifies the followers that the entry is committed.

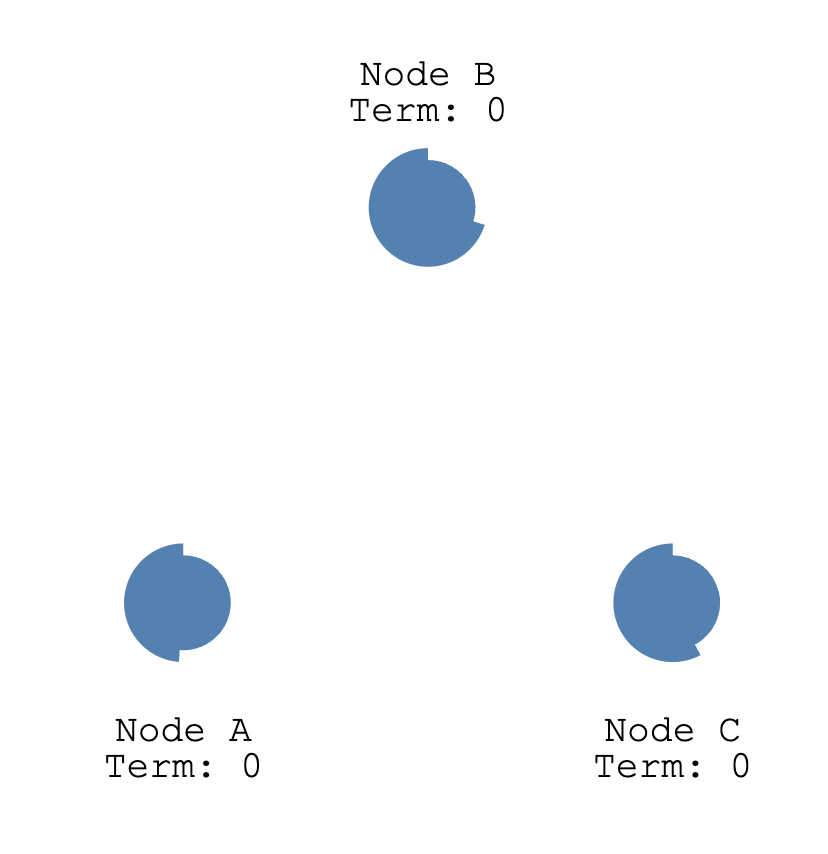
然后所有的followers的entry都被正式set为5（由红变黑）

A screenshot of a computer program

Description automatically generated

The cluster has now come to **consensus** about the system state. (The process of **Log Replication**)

Leader Election:  
In Raft there are two timeout settings which control elections.



First: election timeout

The election timeout is the amount of time a follower waits until becoming a candidate.

The election timeout is randomized to be between 150ms and 300ms.

After the election timeout the follower becomes a candidate and starts a new *election term*...下图。Variable change: the term and vote count increments 1

A diagram of a number of blue circles

Description automatically generated

...votes for itself...

...and sends out *Request Vote* messages to other nodes.

A diagram of a number of circles and numbers

Description automatically generated with medium confidence

If the receiving node hasn't voted yet in this term then it votes for the candidate...

[Their voted For (**possible variable change**) change to the name of the node they vote for, the term increments 1]

A screenshot of a computer screen

Description automatically generated

...and the node resets its election timeout.

Once a candidate has a majority of votes it becomes leader.

The leader begins sending out *Append Entries* messages to its followers.

A screenshot of a computer

Description automatically generated

These messages are sent in intervals specified by the heartbeat timeout.

Followers then respond to each *Append Entries* message.

A screenshot of a computer

Description automatically generated

This election term will continue until a follower stops receiving heartbeats and becomes a candidate. (**When will a follower stops receiving heartbeats? We could probably stop by hand**)

Let's stop the leader and watch a re-election happen.

[we stop node A, node B and node C did not receive anything from A such that their timeout times out, but node B’s timeout precedes before the node C’s so Node B become the leader and starts to send out messages to node A and node C] (**How could Node B become the leader? Isn’t leader need to receive the majority of votes?**)

Node B is now leader of term 2.

A diagram of a diagram

Description automatically generated

Requiring a majority of votes guarantees that only one leader can be elected per term.

[Because node A is stopped, only node C sent reply votes to B now]

If two nodes become candidates at the same time then a split vote can occur.

Let's take a look at a split vote example...

Two nodes both start an election for the same term...

A group of blue circles and green circles

Description automatically generated

...and each reaches a single follower node before the other.

A group of blue circles with black dots

Description automatically generated

Now each candidate has 2 votes and can receive no more for this term.

The nodes will wait for a new election and try again.

Node A received a majority of votes in term 5 so it becomes leader.

A diagram of a diagram

Description automatically generated

[We can have at most 1 leader

Only leader receives the client request]

**Log Replication**

Once we have a leader elected we need to replicate all changes to our system to all nodes.

This is done by using the same *Append Entries* message that was used for heartbeats.

Let's walk through the process.

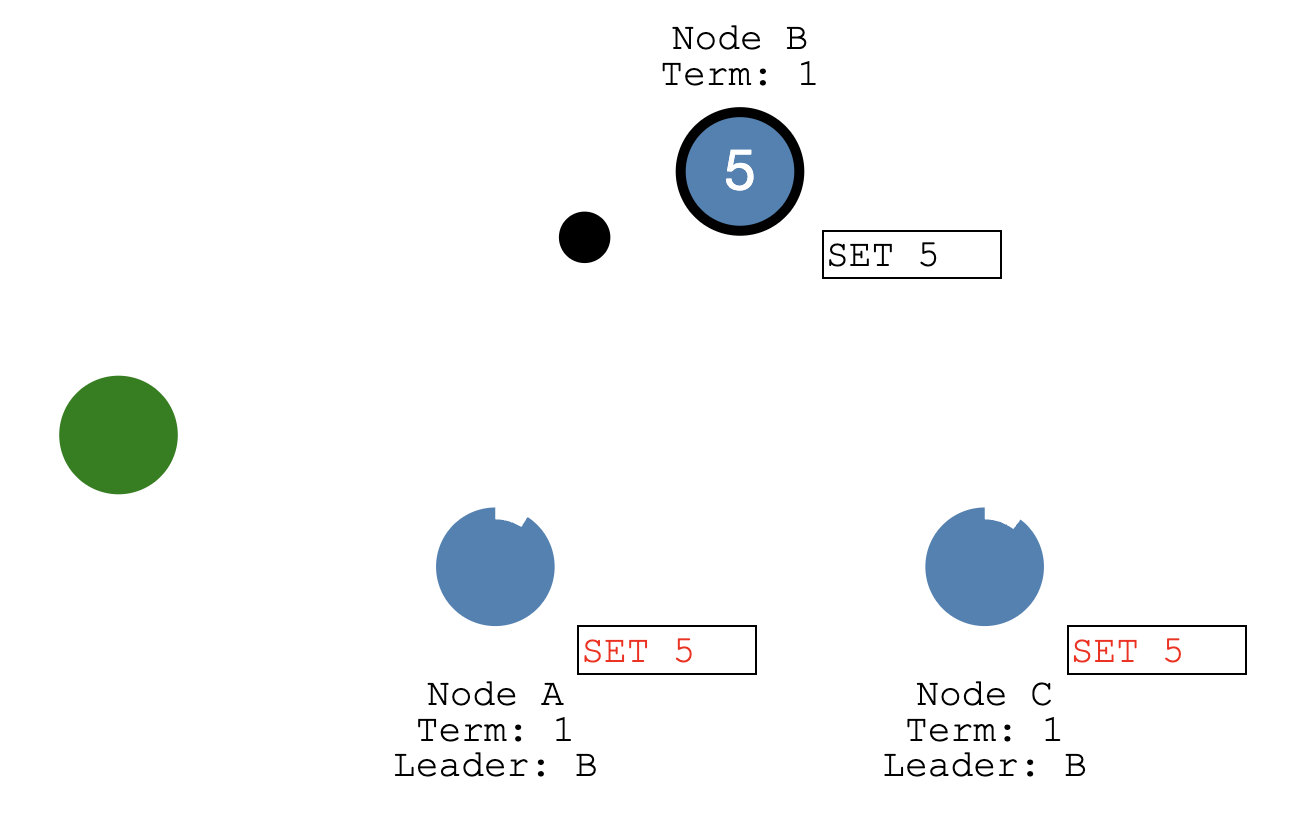
First a client sends a change to the leader.

The change is appended to the leader's log...

...then the change is sent to the followers on the next heartbeat.

An entry is committed once a majority of followers acknowledge it...(the acknowledgement is done by sending reply vote?)

...and a response is sent to the client.(**from leader to client**)



Now let's send a command to increment the value by "2".

A screenshot of a computer

Description automatically generated

Our system value is now updated to "7".

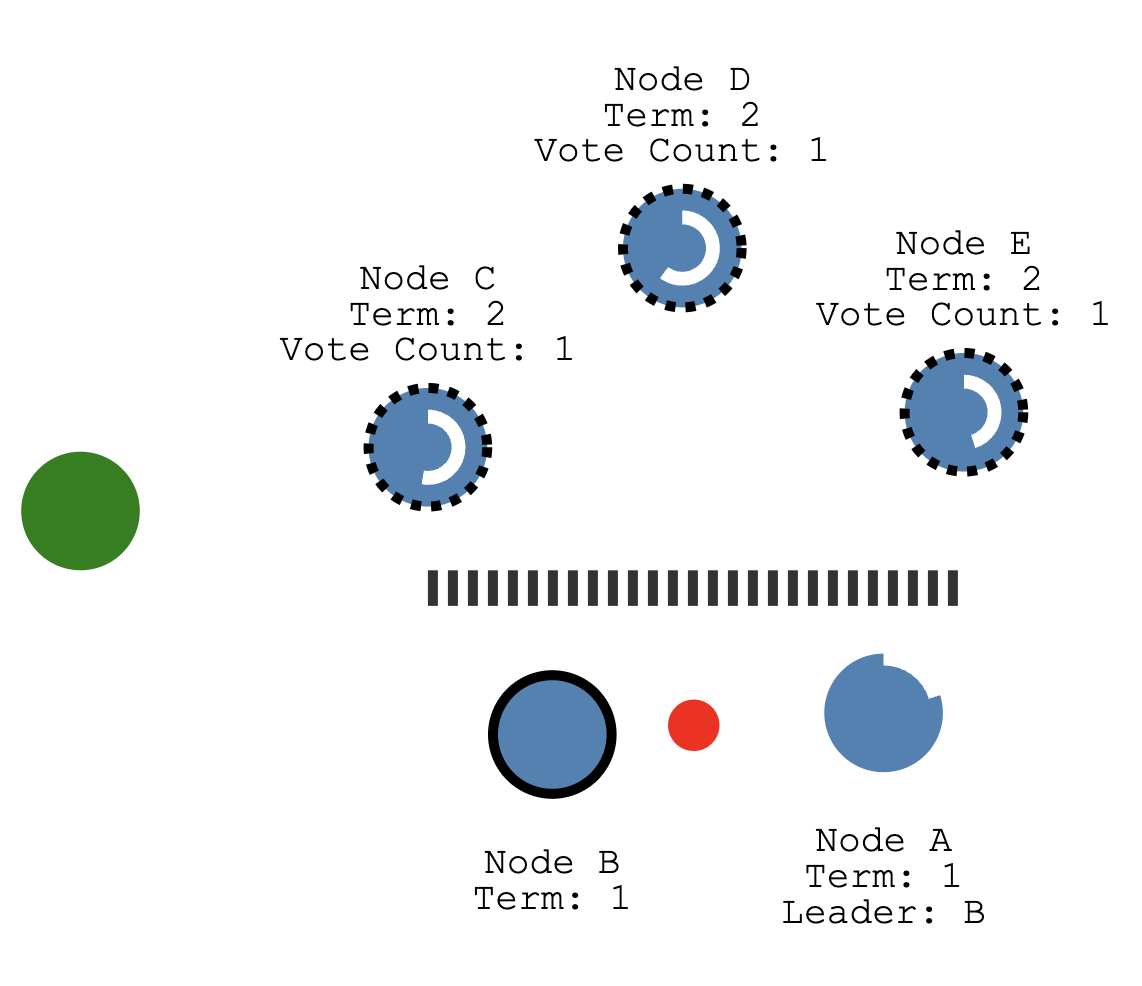
[term increments, Entry updates formally to 5, system value updated, new variable ADD = 2]

A screenshot of a computer

Description automatically generated

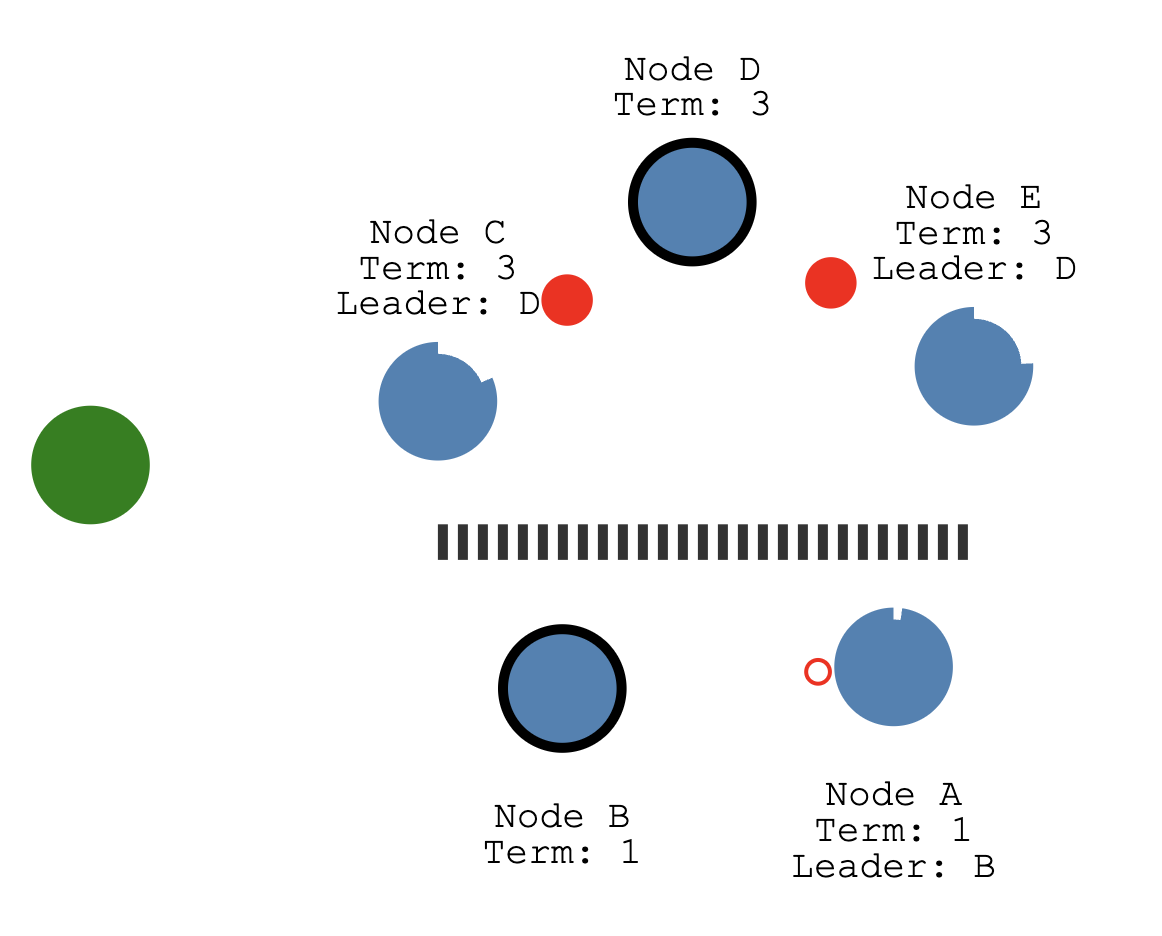
Raft can even stay consistent in the face of network partitions.

Let's add a partition to separate A & B from C, D & E.



Because of our partition we now have two leaders in different terms.

[Node D finishes early, so it becomes the leader and starts to send messaages to node C and E]



Let's add another client and try to update both leaders.

One client will try to set the value of node B to "3".

Node B cannot replicate to a majority so its log entry stays uncommitted.

A diagram of a code

Description automatically generated

The other client will try to set the value of node D to "8".

This will succeed because it can replicate to a majority.

Now let's heal the network partition.

Node B will see the higher election term and step down.

A diagram of a group of blue circles and red dots

Description automatically generated

Both nodes A & B will roll back their uncommitted entries and match the new leader's log.

A diagram of a group of circles and numbers

Description automatically generated

Our log is now consistent across our cluster.

A screenshot of a computer program

Description automatically generated