Study of Raft Algorithm

Group Member	Study ID	Work	Email
Cunze cui	**U201512157**	**Collect and analyze algorithm data**	**461391223@qq.com**
Dongming hu	**U201514523**	**Write and debug the code**	**2439846178@qq.com**
Shuo wei	**U201514482**	**Integration of information, writing documents**	**2411602273@qq.com**
Yinfang chen	**U201514600**	**Translate and proofread the document**	**514274740@qq.com**
Zhuo he	**U201514514**	**Help write some code and test**	**1290919302@qq.com**

Abstract

The algorithm is mainly to solve the problem of general Byzantine.

Byzantine is located in today's Istanbul, Turkey, is the capital of the Eastern Roman Empire. Due to the vast territory of Byzantine Empire at that time, for defense purposes, so each army is far apart, generals and generals can only pass messages between letters. During the war, all the generals in the Byzantine Army must reach an agreement on whether there is a chance to win before they attack the enemy's camp. However, there may be traitors and enemy spies inside the army. The decision of the generals will disrupt the order of the entire army. When a consensus is reached, the result does not represent the opinion of the majority. At this time, the remaining loyal generals reached a unanimous agreement without the influence of a traitor or spy, knowing that there were unreliable members, and the Byzantine problem thus took shape. The Byzantine hypothesis models the real world and computers and networks may experience unpredictable behavior due to hardware errors, network congestion or disconnection, and malicious attacks.

Lamport has been studying such issues, published a series of papers. But a comprehensive summary is to answer the following three questions:

1.Is there a solution to the problem of distributed consistency like General Byzantine?

2.What kind of conditions need to be satisfied if there is solution?

3.On the basis of specific preconditions, a solution is proposed.

The first two questions are relatively simple,and Paxos algorithm can solve the third problem,but this algorithm is very difficult to understand. Raft algorithm is built on the hope of getting a better-understood alternative to Paxos algorithm.

1.General introduction of Raft Algorithm

Before we understand Raft, we first understand Consensus conception of consistency. It refers to the state of multiple servers agreed, but in a distributed system, because of various accidents may, some servers may collapse or become unreliable, It can not agree with other servers. This requires a Consensus protocol, which is to ensure fault tolerance, that is, even if there are one or two servers in the system that do not impact its processing. In order to agree in a fault-tolerant way, we can not expect 100% of all servers to be consistent.As long as more than half agree ,it’s already okay . Assuming N servers, $N / 2 + 1$ servers are over half, which make the most ones.

In Raft, at any one time a server can play one of the following roles:

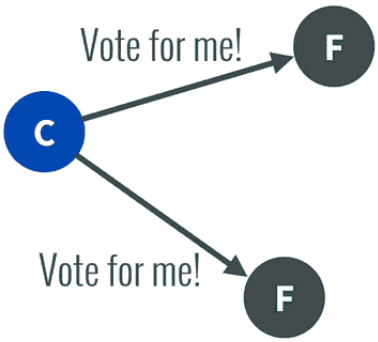
1.Leader: deal with all client interaction, log replication, generally only one Leader at a time.

2.Follower: similar to voters, completely passive

Candidate Candidates: Lawyers like Proposer can be chosen as a new leader.

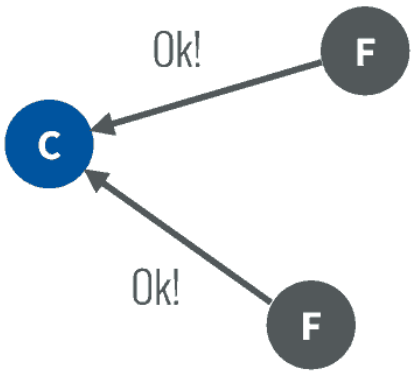
The Raft phase is divided into two phases, starting with the election process and then conducting normal operations under the leadership of the elected leaders, such as log copying. The following diagram shows this process:

1. Any server can become a candidate Candidate, it issued to other server Follower Election of their own request:



Picture 1-1

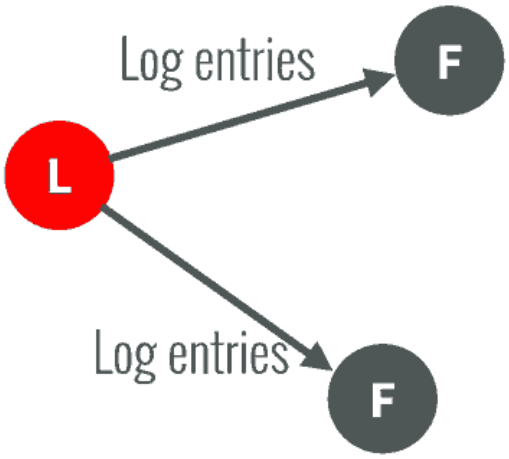
2.Other servers agreed, issued OK.



Picture 1-2

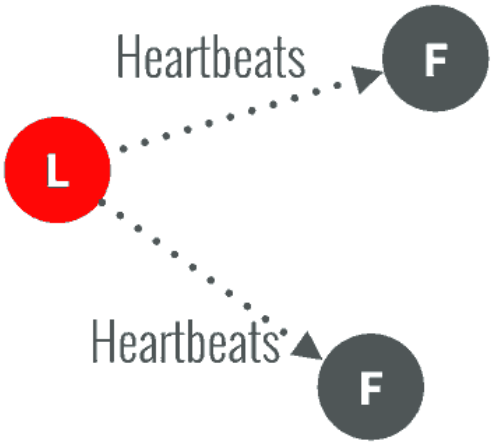
3.Note that if there is a Follower in the process that does not receive the request to request an election, the candidate can elect himself. As long as the majority vote of N / 2 + 1 is reached, the candidate can still become a Leader.

So this candidate becomes the Leader leader, which can give instructions to voters, who followers, such as log copying



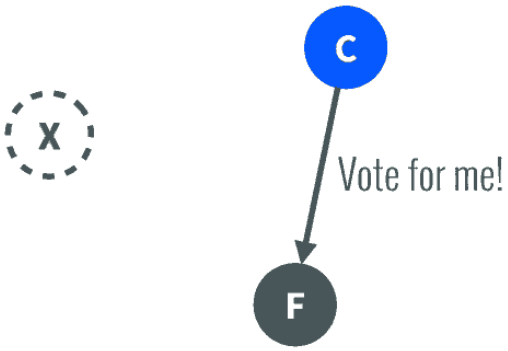
Picture 1-3

4. Later notification of heartbeat log copying



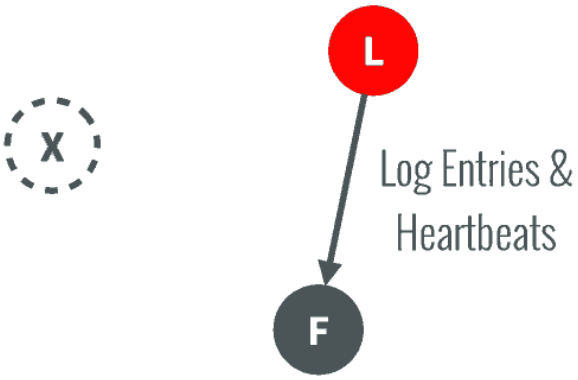
Picture 1-4

5. If, once the Leader crashes, there is a candidate in Follower who issues an invitation to vote.



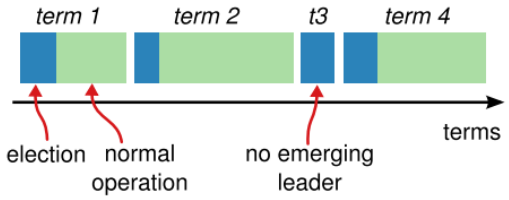
Picture 1-5

6. Follower agreed, it became Leader, continue to undertake the guidance of the work such as log copy:



Picture 1-6

Raft divides time into terms of arbitrary length, as shown in Picture 1-7. Terms are numbered with consecutive integers. Each term begins with an election, in which one or more candidates attempt to become leader as described in Section 5.2. If a candidate wins the election, then it serves as leader for the rest of the term. In some situations an election will result in a split vote. In this case the term will end with no leader; a new term (with a new election) will begin shortly. Raft ensures that there is at most one leader in a given term.

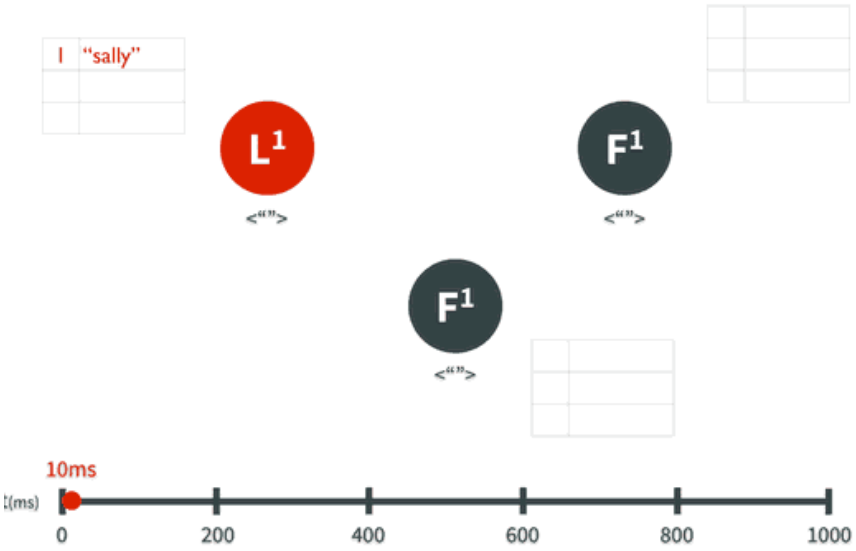


Time is divided into terms, and each term begins with an election. After a successful election, a single leader manages the cluster until the end of the term. Some elections fail, in which case the term ends without choosing a leader. The transitions between terms may be observed at different times on different servers.

Picture 1-7

The following example shows the log copy Raft algorithm, assuming the leader of Leader has been selected, then the client sends a request to add a log,

such as the log is "sally":

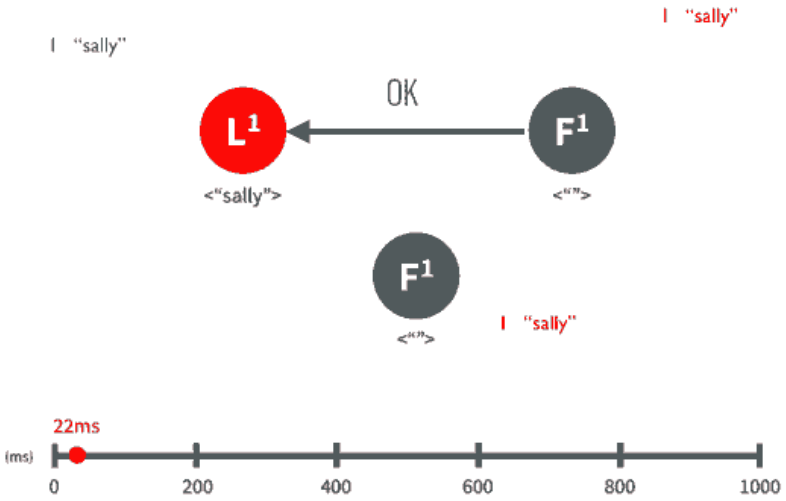


Picture 1-8

Leader asked Followe to follow his instructions, appending this new log to their respective logs:

Picture 1-9

Most follower servers log the disk file, confirm the additional success, issued by Commited Ok:



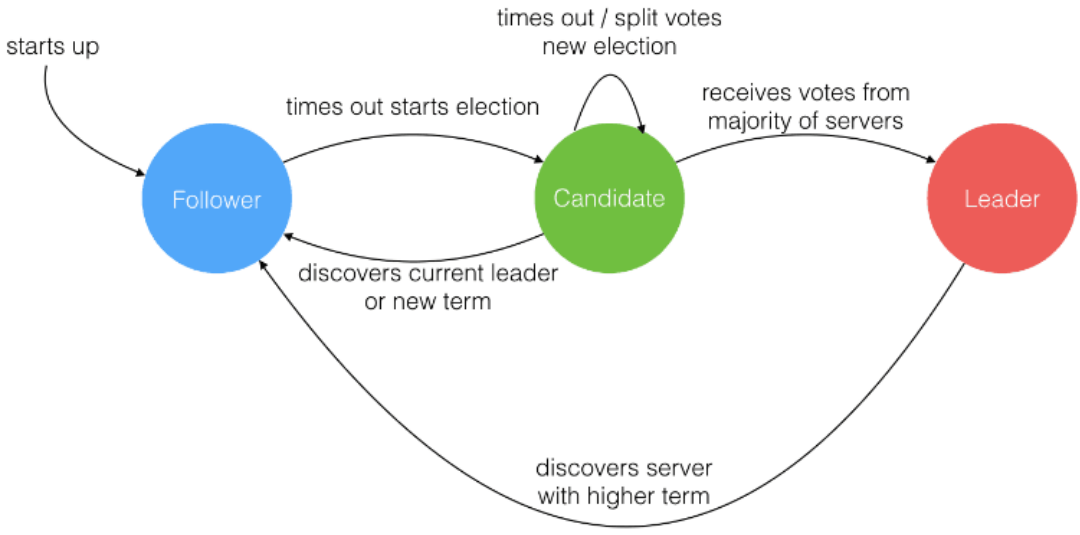
Picture 1-10

In the next heartbeat, Leader notifies all Follwer to update the commited items.

Repeat this process for each new log record.

If, in the process, a network partition or network communication failure has prevented the Leader from accessing most Follwers, then the Leader can only update those Follower servers it can access normally, and most server followers do not have a Leader, They re-elect a candidate as a Leader, and then the Leader as a representative to deal with the outside world, if the outside world asked to add a new log, the new Leader according to the above steps to notify most Followers, if the network fault is fixed, then the original The Leader becomes a Follower, any updates to this old Leader can not be commit during the out-of-sync phase, both roll back and accept the new updates for the new Leader.

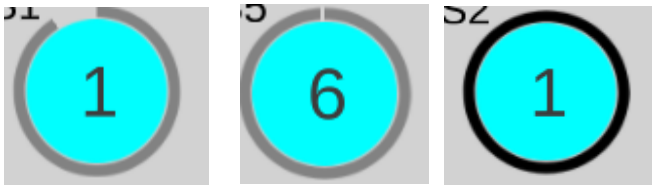
The total flow chart:



Picture 1-11

2.The results of Raft protocol test

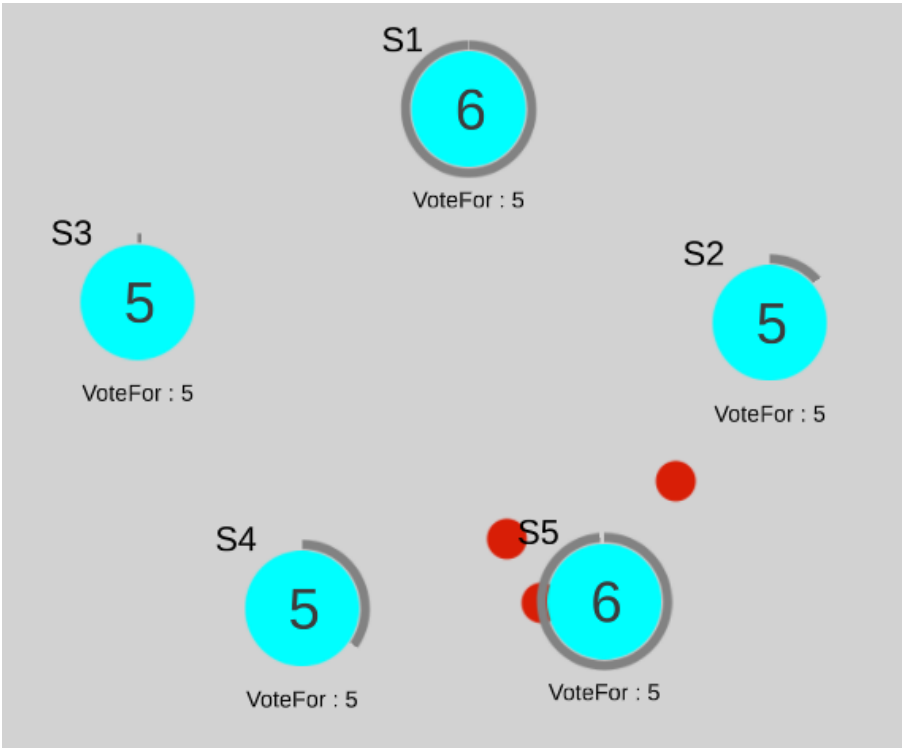
In this protocol,a node can be in 1 of 3 states:the Follower state、the Candidate or the Leader state.As is shown in Figure 1.1 below,all our nodes start in the follower state.



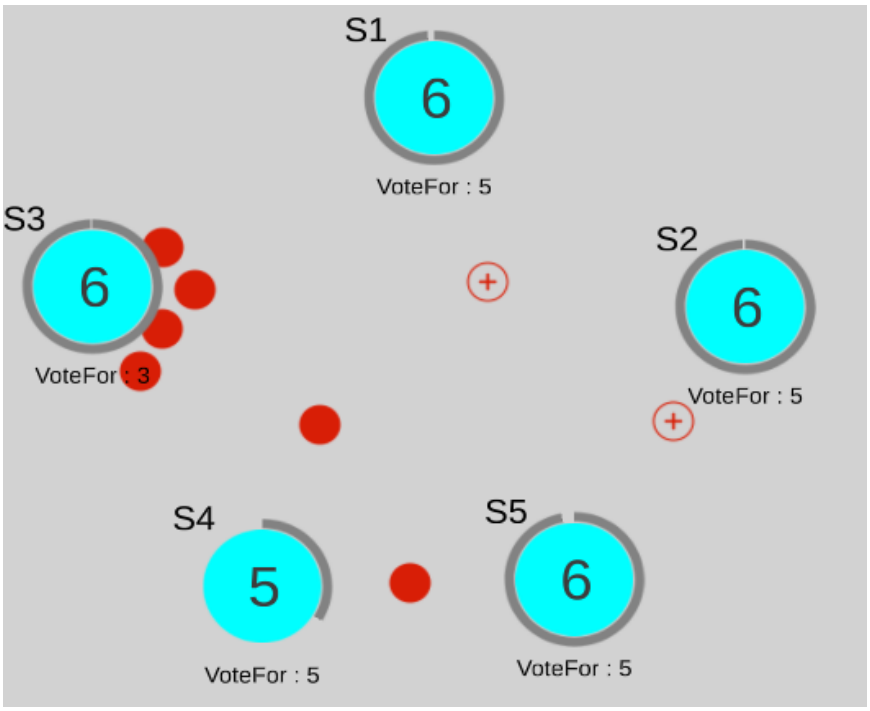
Picture2-1: followers, candidates, leaders from left to right

（1）The test of the leader election process

The following will simulate the process of election leaders appear.If followers don’t hear from a leader then they can become a candidate.And then the candidate requests votes from other nodes.Nodes will reply with their vote. See Figure 1.2 .

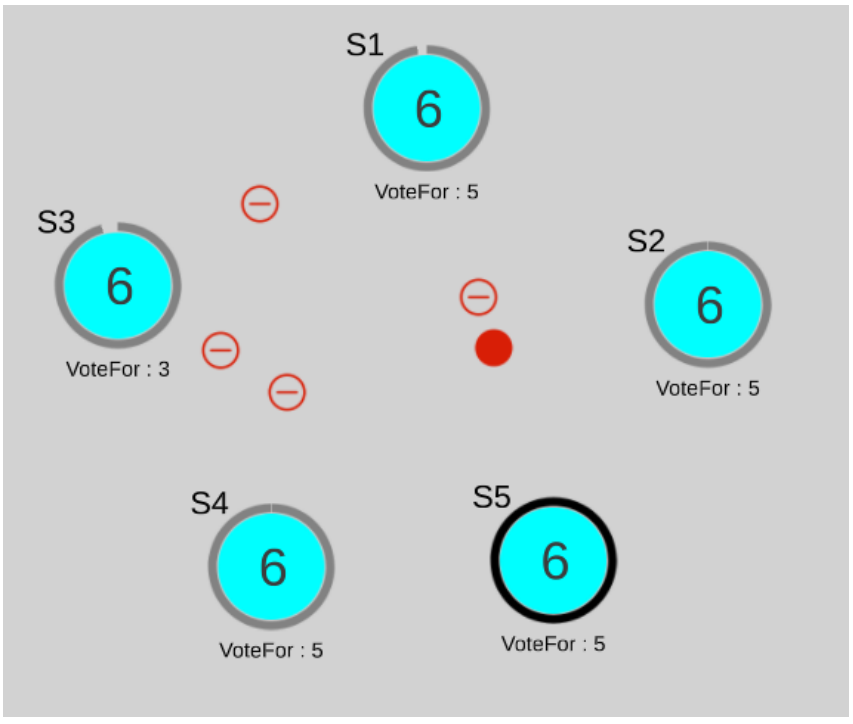


Picture2-2(a) :Candidate S5 requests a vote



Picture2-3 (b) :Node S1, S2 replies S5 and S3 requests a vote

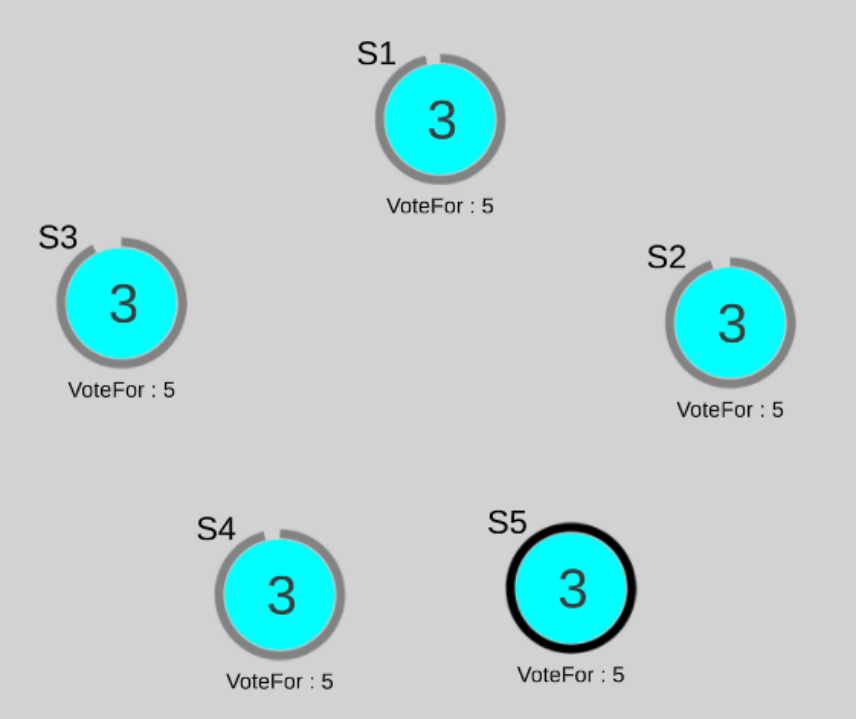
And at this time， the candidate becomes the leader if it gets votes from a majority of nodes. See Figure 1.3 .



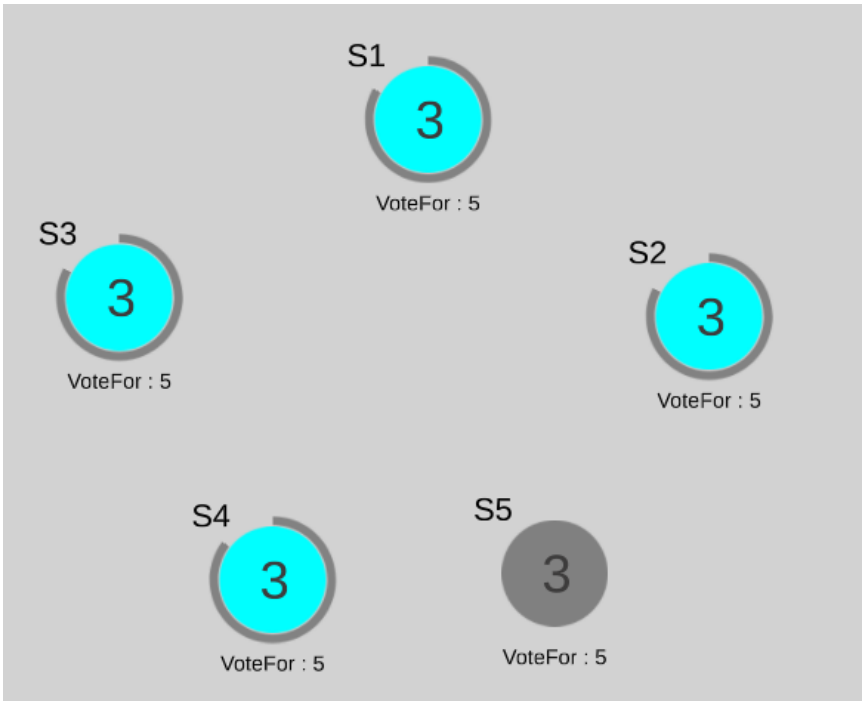
Picture2-4: Node S5 becomes the leader

The above is the simplest example of a leader. Next we test the timeout settings and split vote.In Raft there are two timeout settings which control elections.First is the election timeout.The election timeout is the amount of time a follower waits until becoming a candidate. It is randomly allocated between 150ms and 300ms. After election timeout, followers become candidates and begin a new election deadline, vote

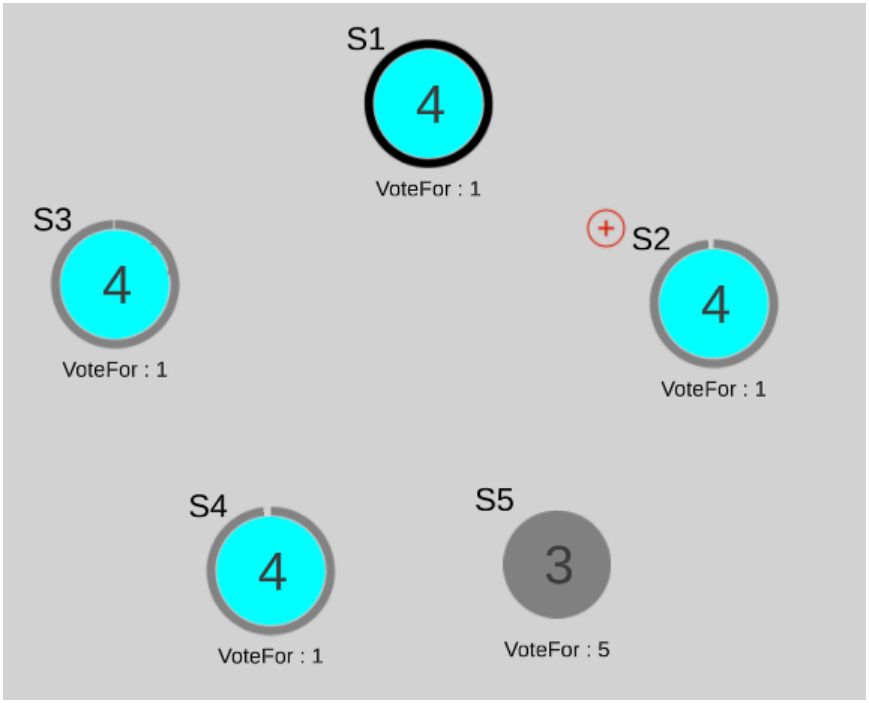
for themselves and send a request for voting messages to other nodes. If the receiving node did not cast a vote, it votes for the candidate and the node resets its election timeout. Once the candidate receives the majority vote, it becomes the leader. Then, the leader begins to send an additional entry message to the follower. These messages are sent at the interval specified by heartbeat timeout. Followers respond to each additional entry message. This election will last until the followers stop accepting heartbeat and become a candidate. Most votes required to become a leader to ensure that each term can only choose one leader. Figure 1.4 shows the emergence of a new leader after the leader has been shut down.



Picture2-5 (a): S5 is the leader

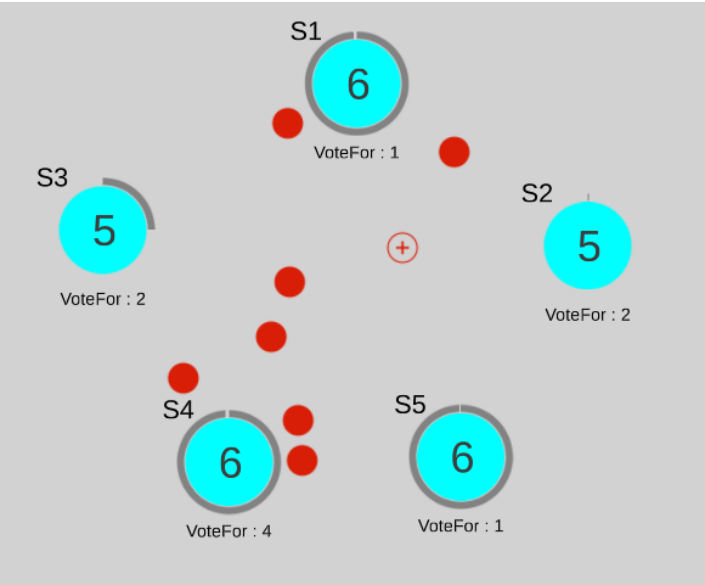


Picture2-6 (b) :S5 shutdown



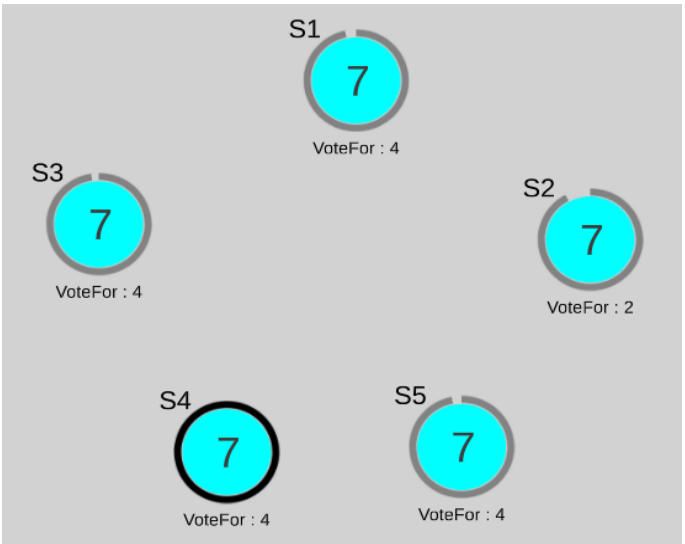
Picture2-7 (c) :The S4 election is the new leader

If both nodes become candidates at the same time, a split vote may occur. As shown in Figure 1.5 below, to test a split vote example, the result is normal.



Picture2-8 (a): S1 and S2 have two votes simultaneously

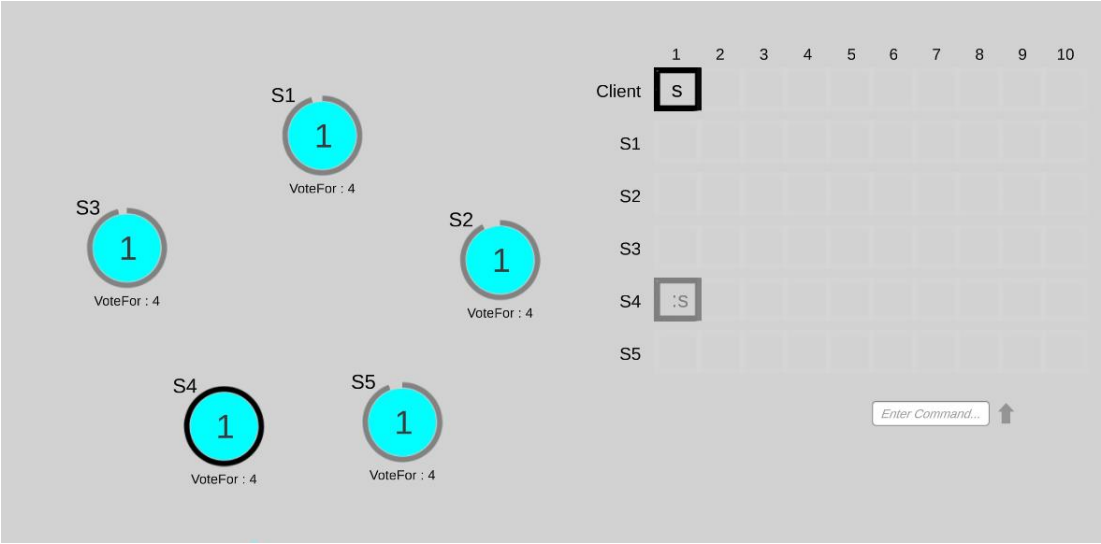
Picture2-9 (b): Node resets election timeout



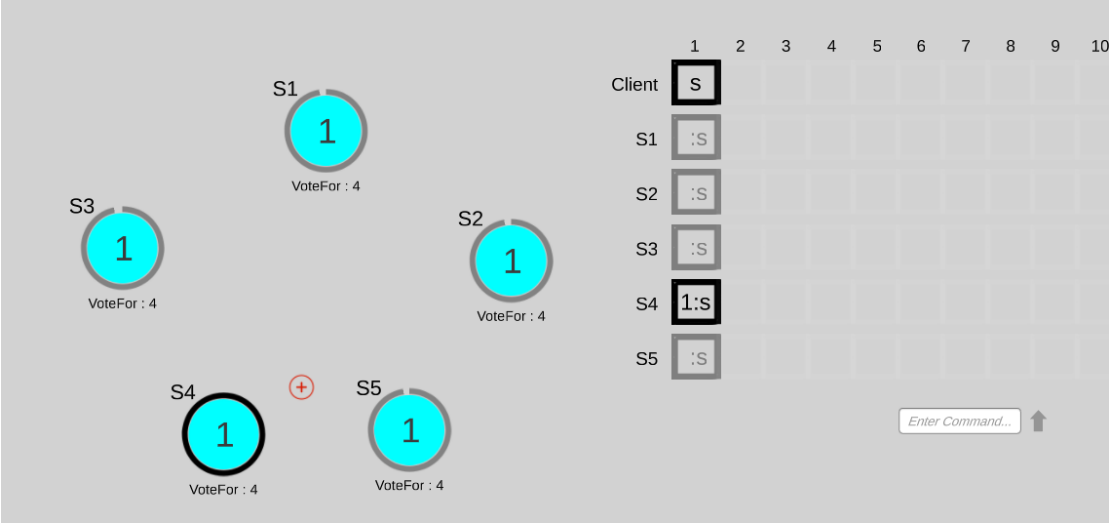
Picture2-10 (c) :S4 is the new leader

（2） Log copy process test

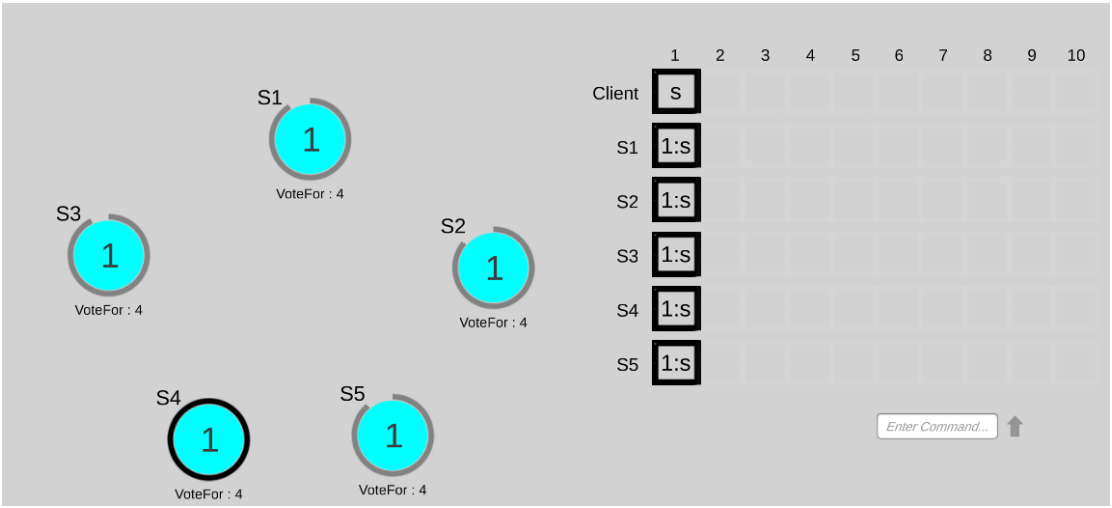
Now that all the changes in the system are made through the leader, each client task is added as an entry in the node log. The leader log entries are not updated until they are not submitted. To submit an entry, the node first copies it to the follower node. The leader then waits until most nodes have written entries. Then, the entry is submitted on the leader node, the node status depends on the specific content of the entry. Then, the leader notifies his followers and executes the items shown. When follower actions, the cluster has now reached a consensus on the status of the system. The test results are shown in Figure 1.4.



Picture2-11 (a) :client publishes entry to leader S4



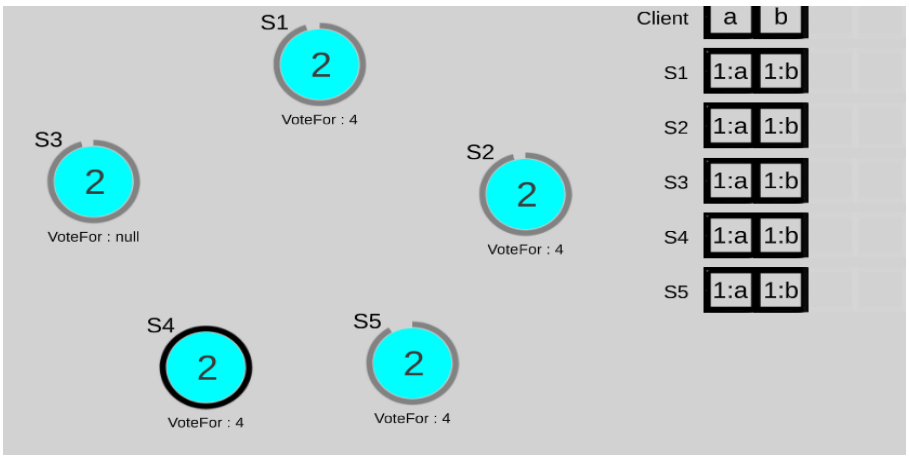
Picture2-12 (b) :The leader copies the entry to followers



Picture2-13 (c): Leader and follower agree

Next suppose one of our followers, while receiving the leader's copied entry, disconnects.After some time the leader sends a new entry, at which point the follower connects again .The leader sends all the entries thar were not received by this follower to him.So the last status of this follower will be synchronized with other nodes, as shown in Figure 2.2 below.

Picture2-14 (a): Follower S3 is disconnected and a is not received



Picture2-15 (b) :Follower S3 returns with the same status

3. references

[1] Diego Ongaro and John Ousterhout. In Search of an Understandable Consensus Algorithm (Extended Version). Stanford University,pp.1-5.

[2] http://blog.csdn.net/erlib/article/details/53671783

[3] https://www.cnblogs.com/mindwind/p/5231986.html