Introduction to Hyperledger Fabric

How would Gophers vote for a president?

Sukrit Handa
Software Engineer, SecureKey Technologies
@sukrithanda

Hyperledger Fabric

Distributed Ledger / Blockchain Platform

Built for private/permissioned networks

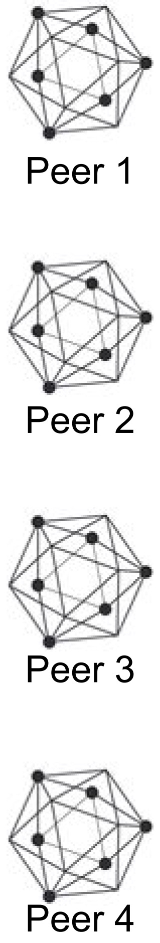
Open Source. Code base is mostly Go!

1.0 recently released

A New Voting System

Voting done online

Vote counting done using code


Peer 1
Peer 2
Peer 3
Peer 4

The Ledger

Contains the world state, a persistent key/value store

Contains a transaction log of all the updates to the world state

The ledger is replicated among all of the peers in a network

Smart Contracts

Known as chaincode in Hyperledger Fabric.

Business logic executed by the Peers to modify/access the world state.

Executed based on incoming transactions.

Written in Go!

Peer 1

Peer 2

Peer 3

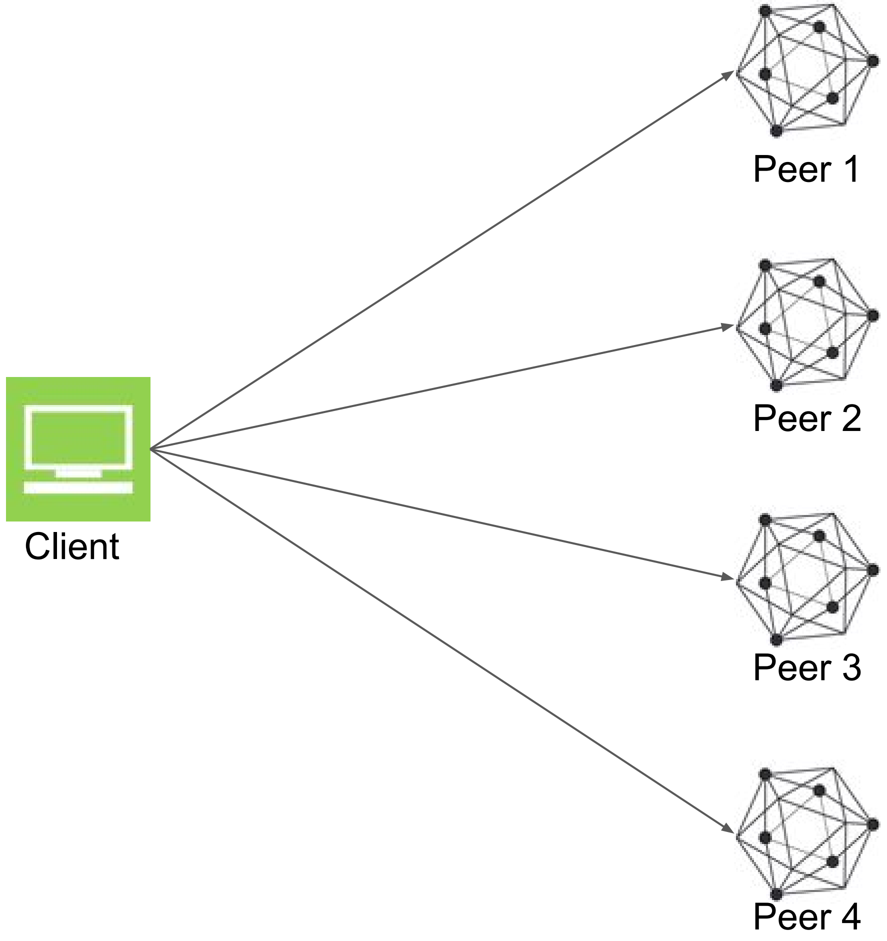
Peer 4

Client

Deploying chaincode

Generating and sending transactions

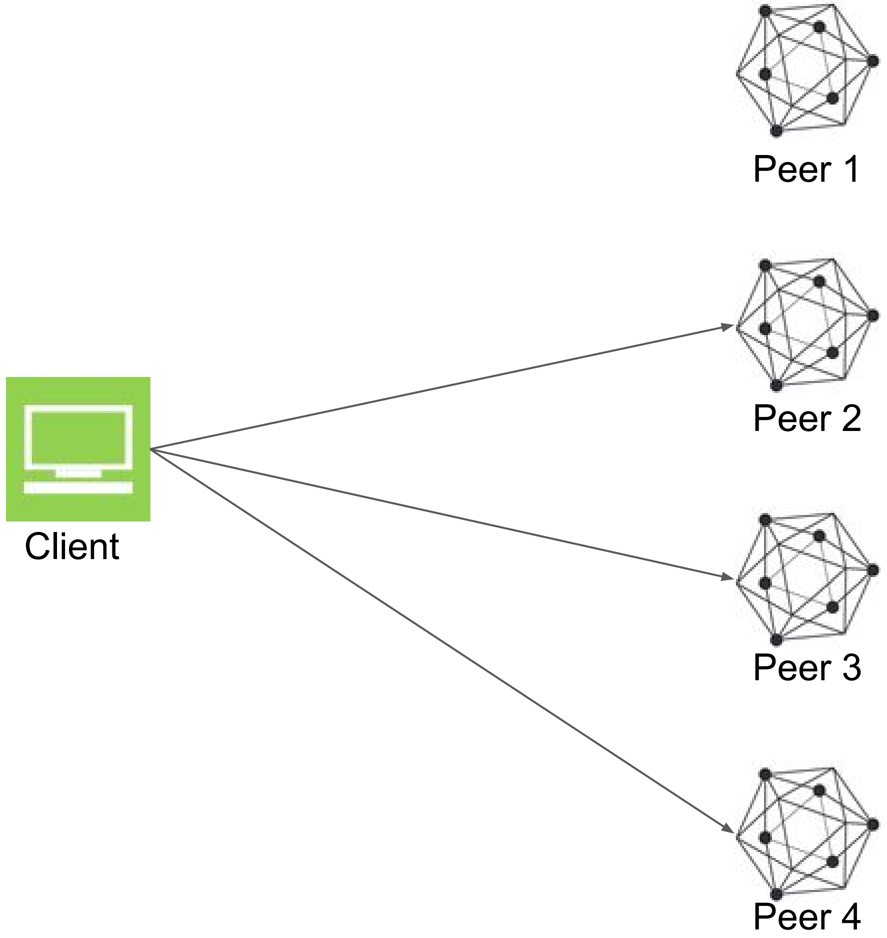
Collecting results

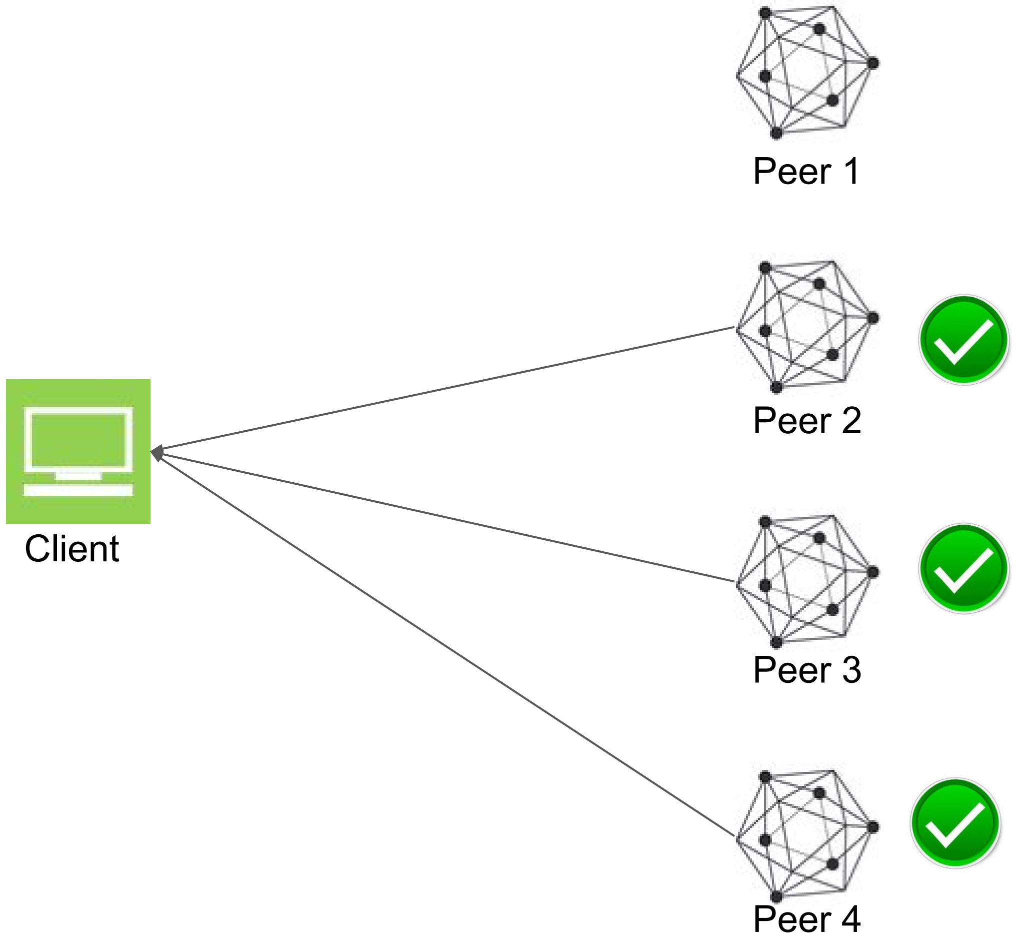

Client
Peer 1
Peer 2
Peer 3
Peer 4

Consensus Mechanism

When chaincode is deployed an endorsement policy is configured

A way for peers to agree on changes to the ledger.


Peer 1
Peer 2
Client
Peer 3
Peer 4

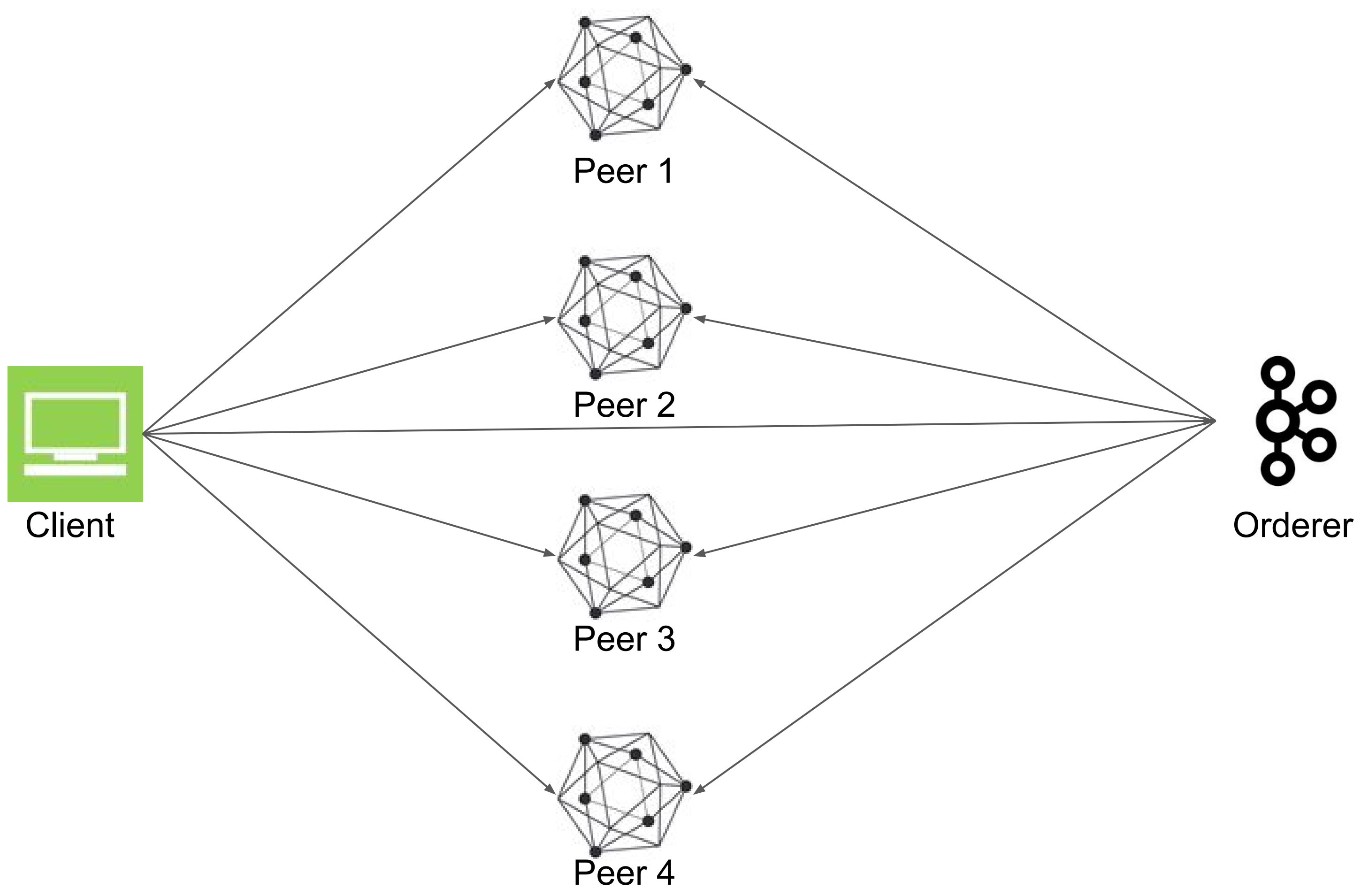

Peer 1
Peer 2
Client
Peer 3
Peer 4

Ordering Service

Listens to endorsed transactions from clients

Broadcasts the endorsed transaction to all the peers

Guarantees total - ordered and atomic broadcasts


Peer 1
Peer 2
Client
Peer 3
Peer 4
Orderer

Let’s Vote

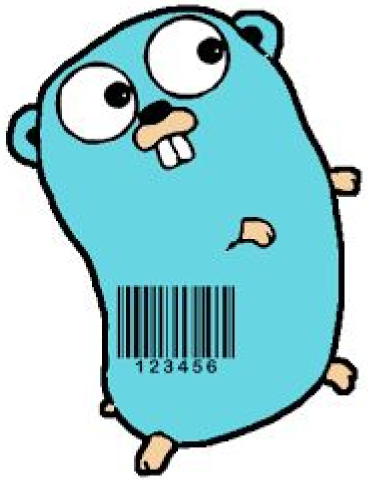
Some Assumptions:

Voting chaincode is deployed

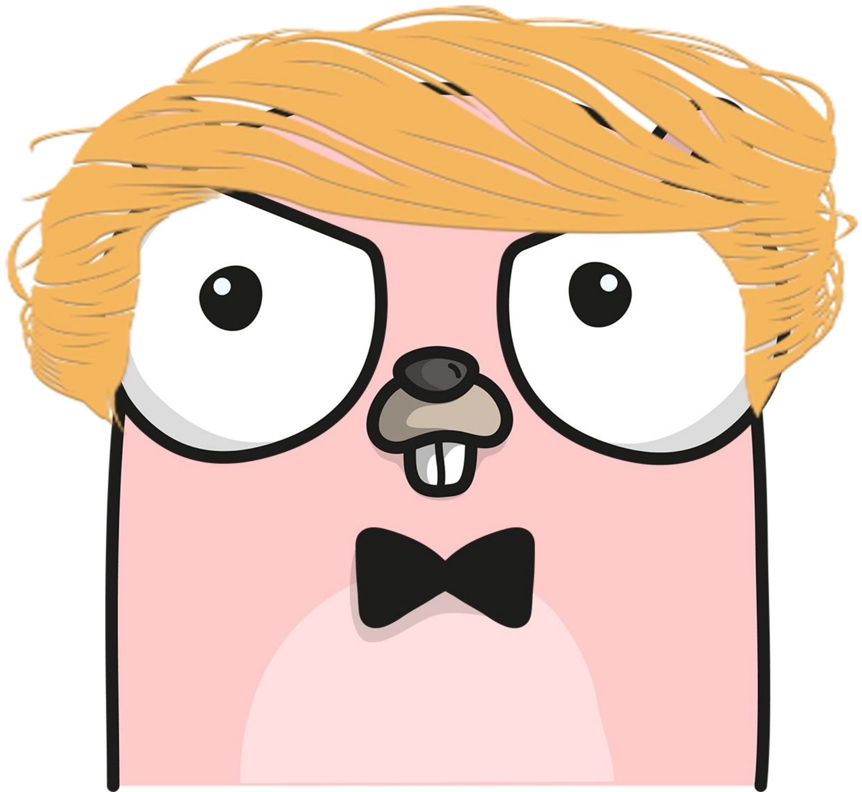
Endorsement Policy = 3 Peers (75%)

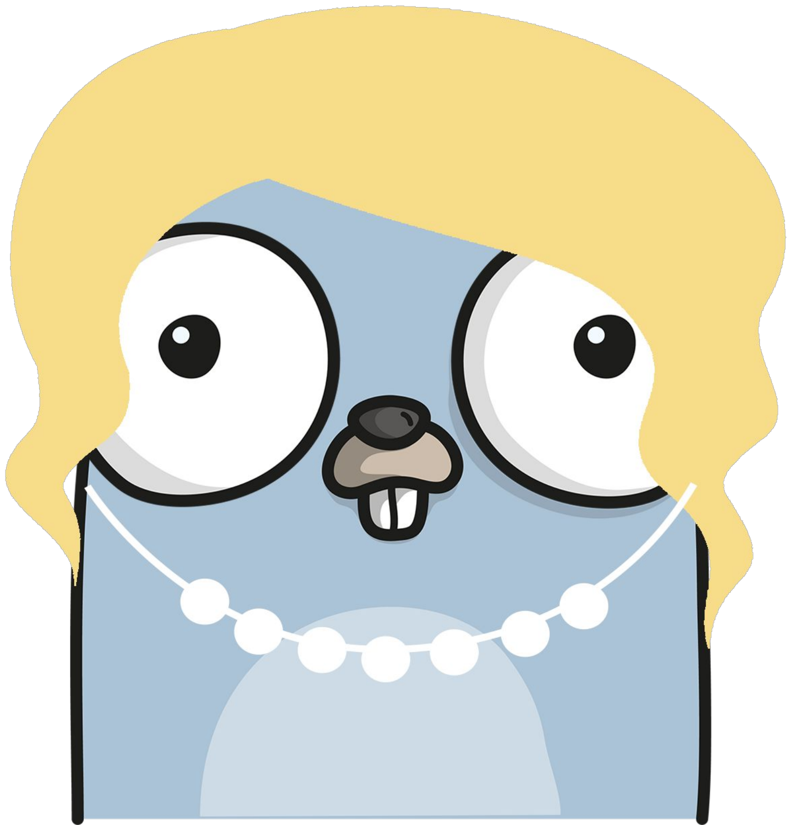
Every Gopher owns a public/private key pair

Public keys have already been registered

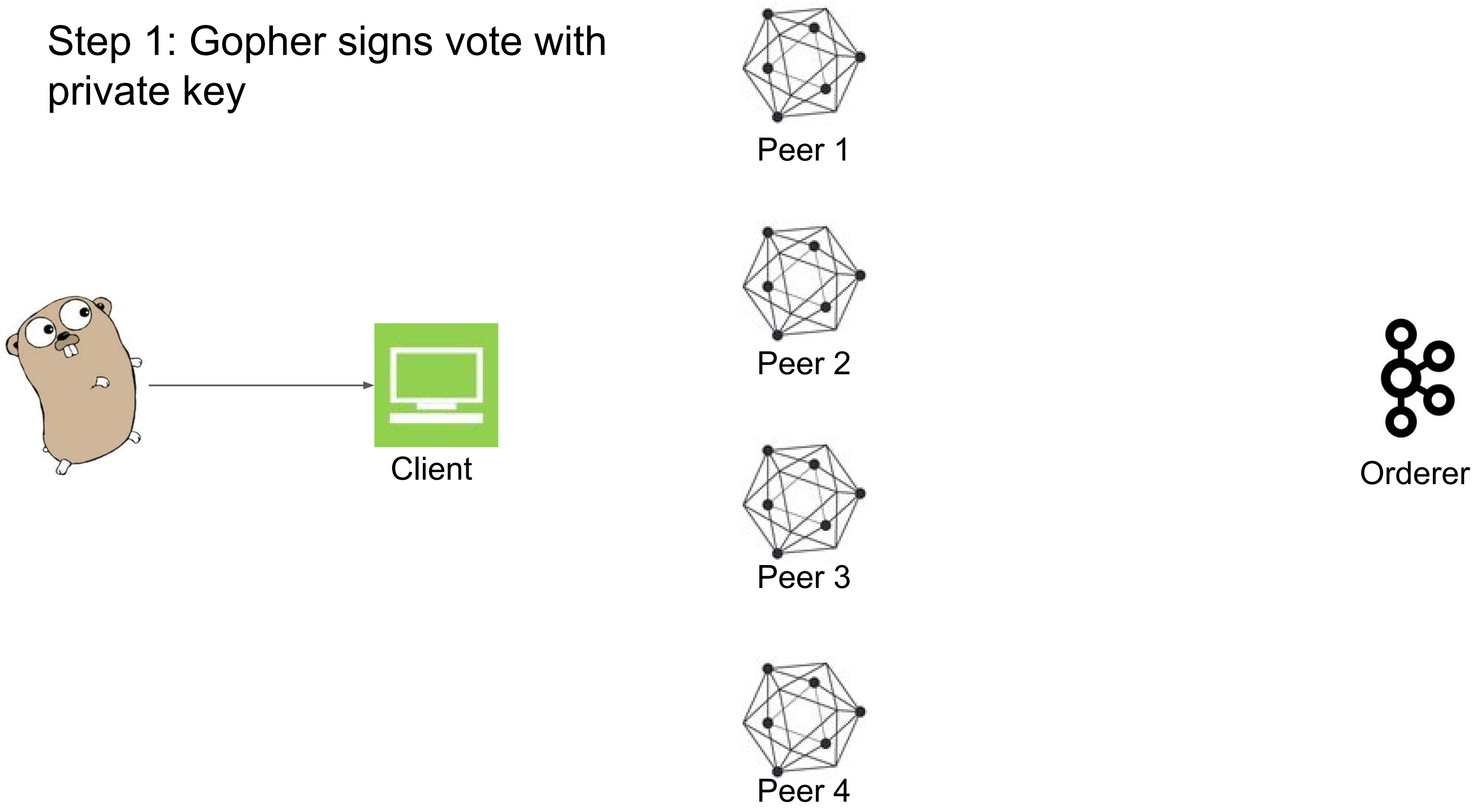


The Candidates

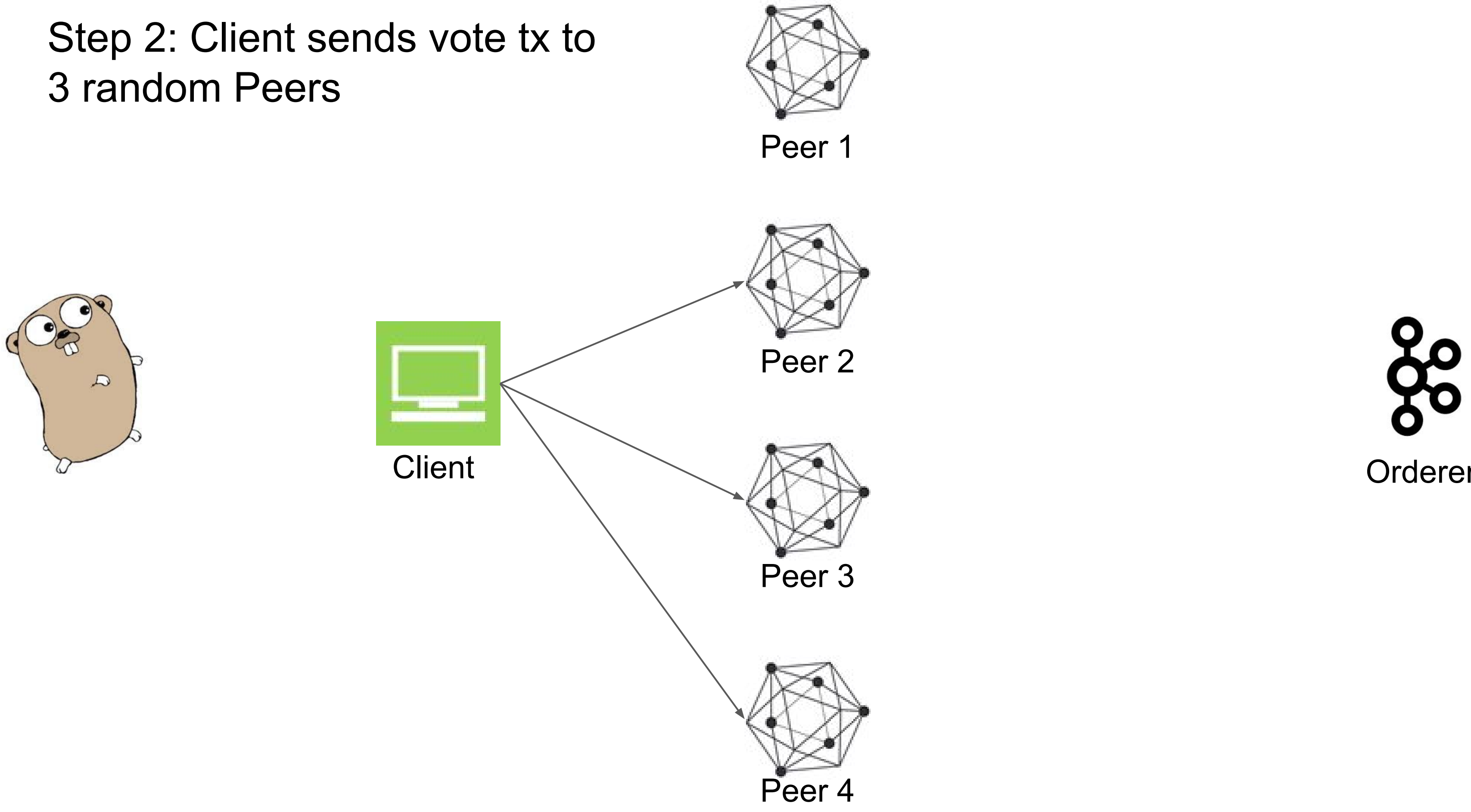




Step 1: Gopher signs vote with private key



Step 2: Client sends vote tx to 3 random Peers



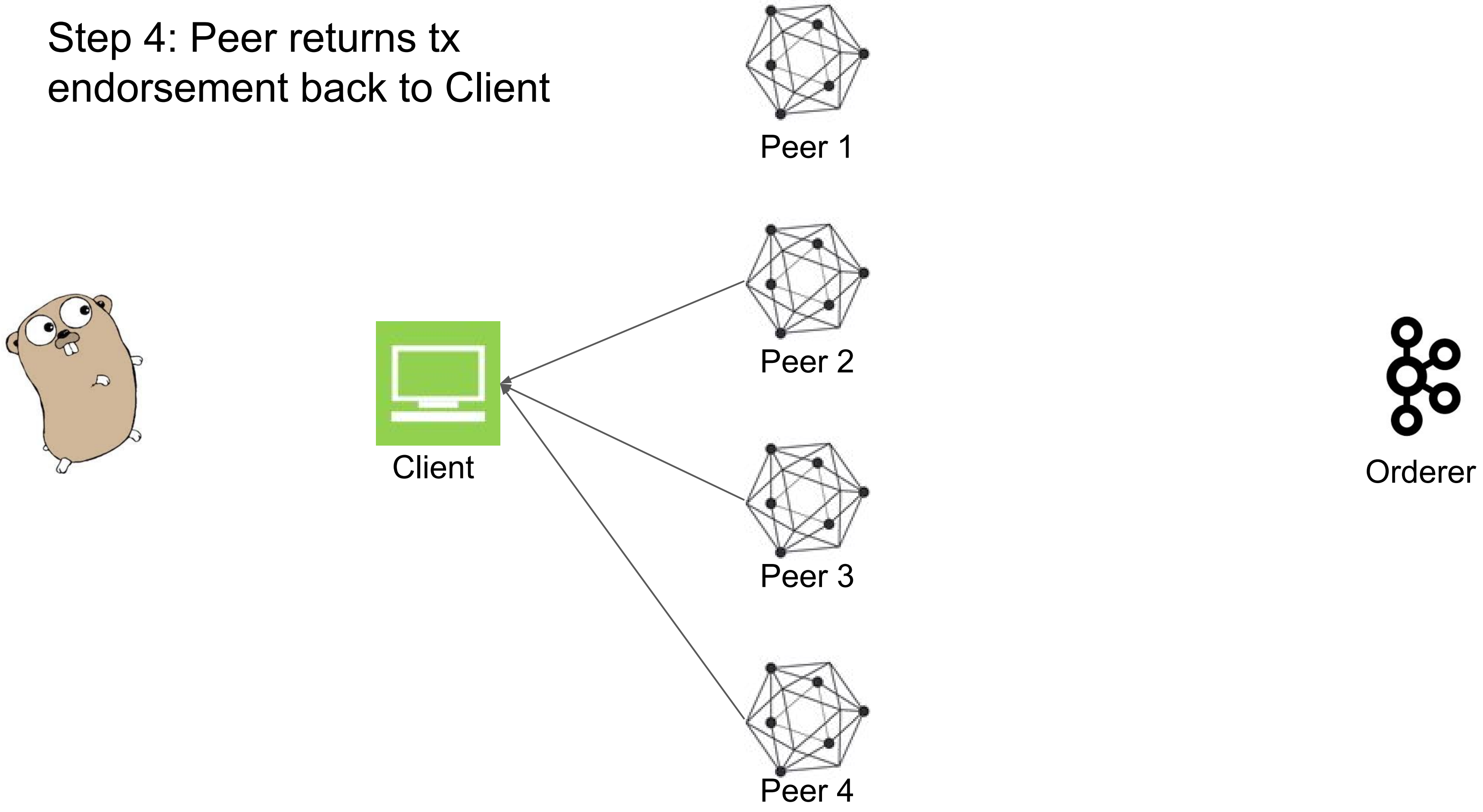
Voter Chaincode

```
//Register a new voter
 Register()
//Cast a Vote
 Vote()
//Signal the start of the voting period
 StartVotePeriod()
//Signal the end of the voting period
 EndVotePeriod()
//Get the final vote count
 GetVoteCount()
```

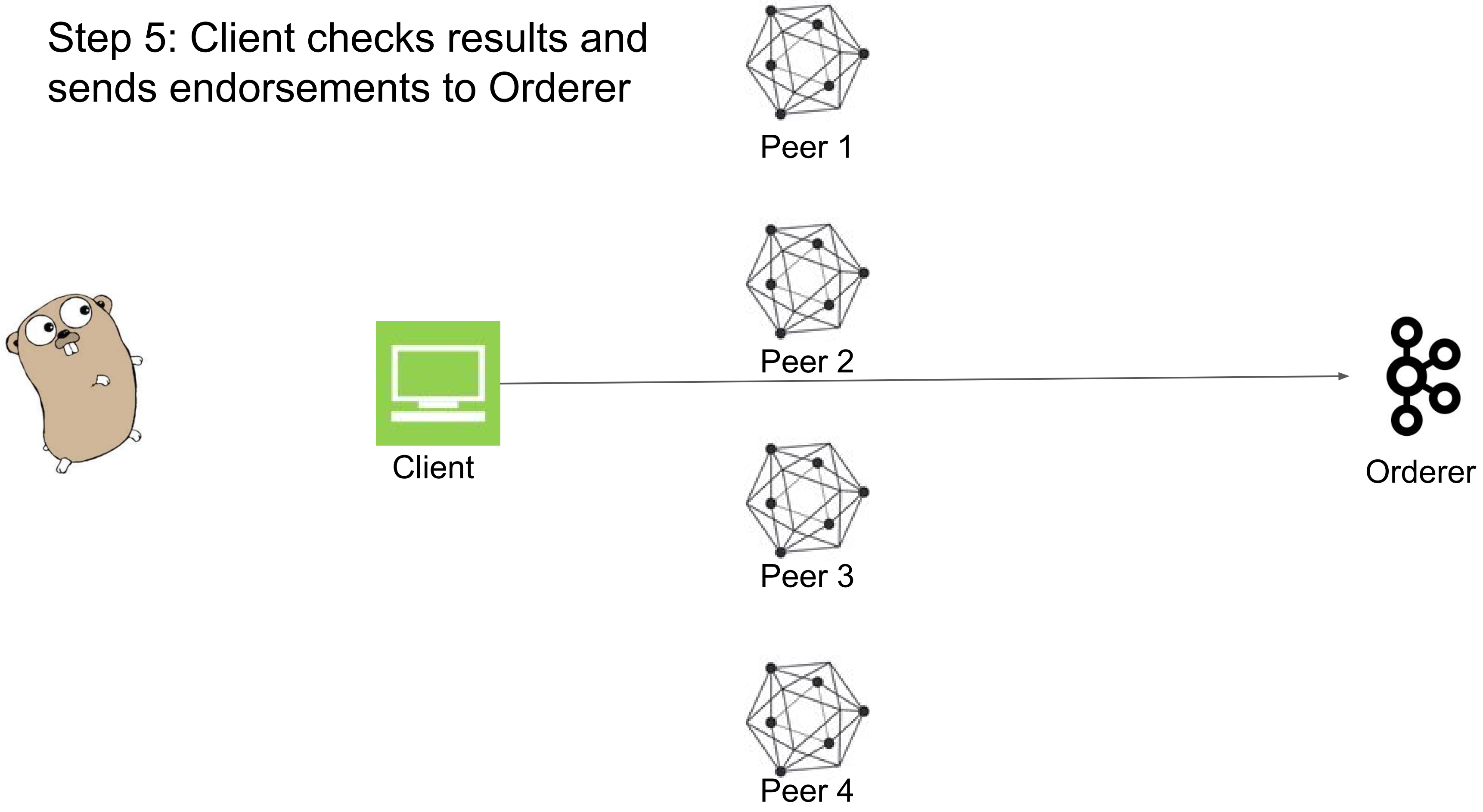
Step 3: Peer executes chaincode

```go
//Vote handles a client sending a Vote transaction
func (t *VoteChaincode) Vote(stub shim.ChaincodeStubInterface, publickey []byte,
signedvote []byte) pb.Response {

    //check the vote is valid and extract the candidate voted for
    candidate, err := checkandExtractVote(publickey, signedvote) //handle error

    //get the list of votes for the candidate
    votelist, err := stub.GetState(candidate) //handle error
    votelist = append(voteslist, signedvote)

    //store new list
    err = stub.PutState(candidate, votelist) //handle error

    //change voter state so the voter cannot vote again
    err = stub.PutState(publickey, []byte("voted")) //handle error
    return shim.Success(nil)
}
```

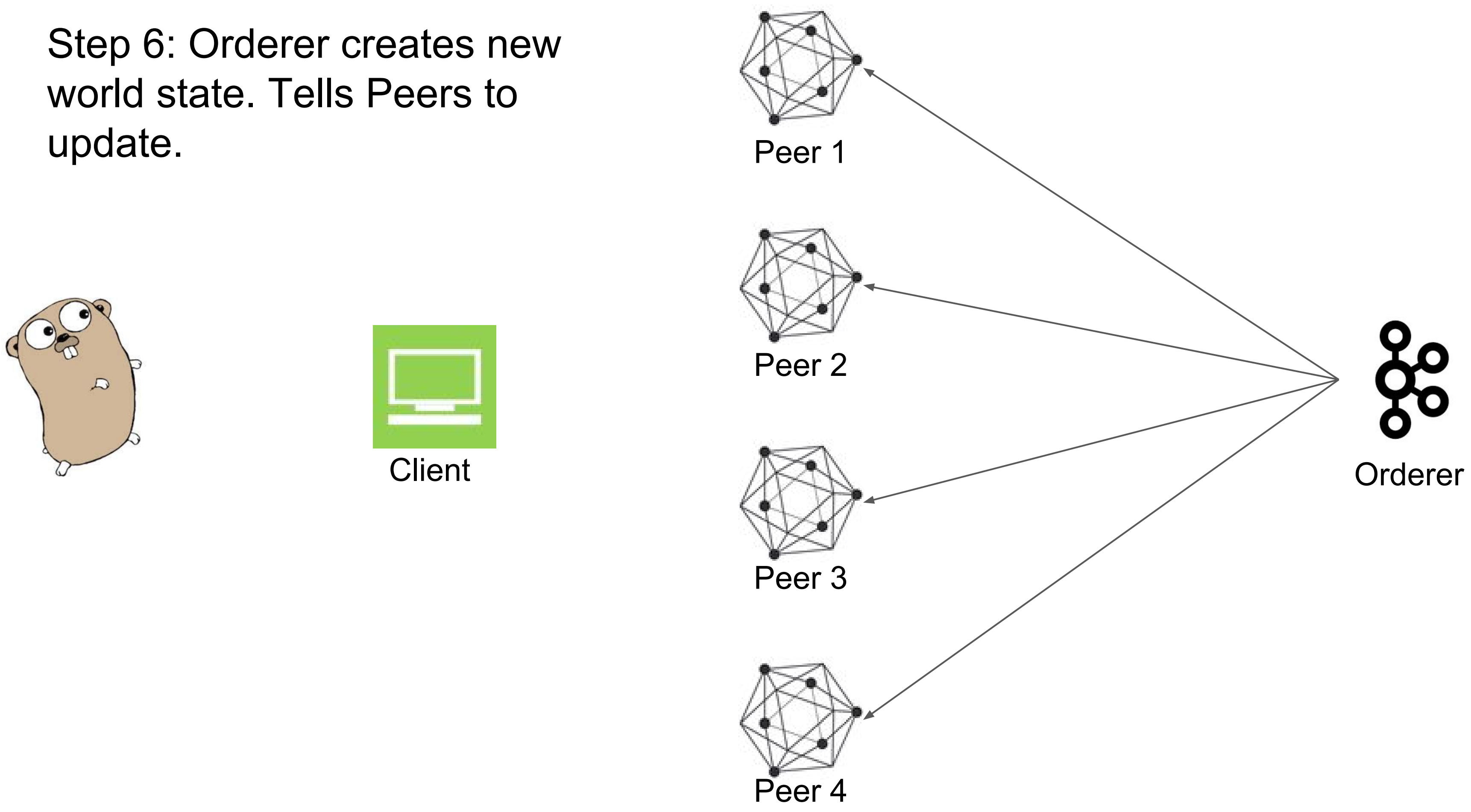
Step 4: Peer returns tx endorsement back to Client



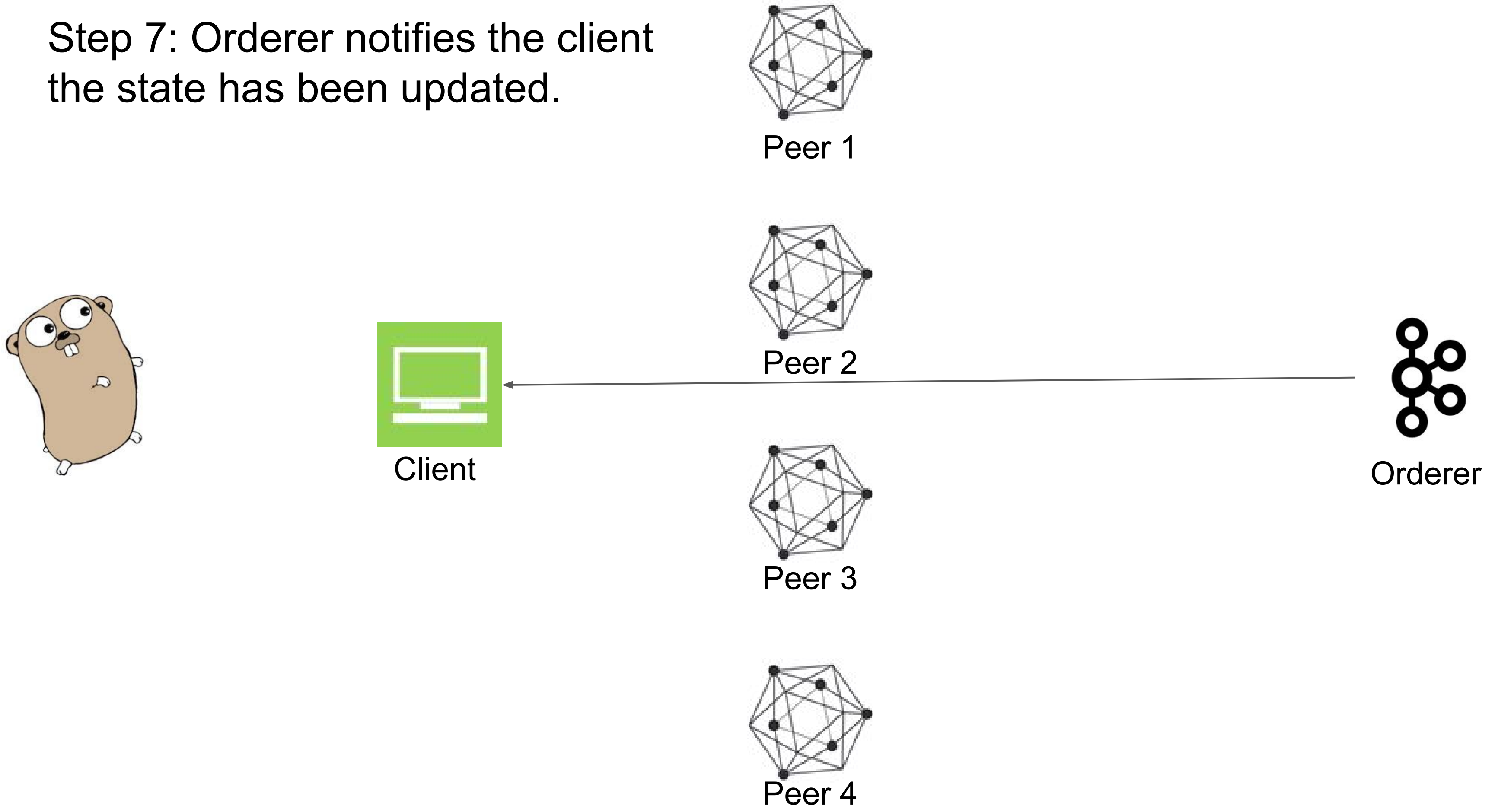
Step 5: Client checks results and sends endorsements to Orderer



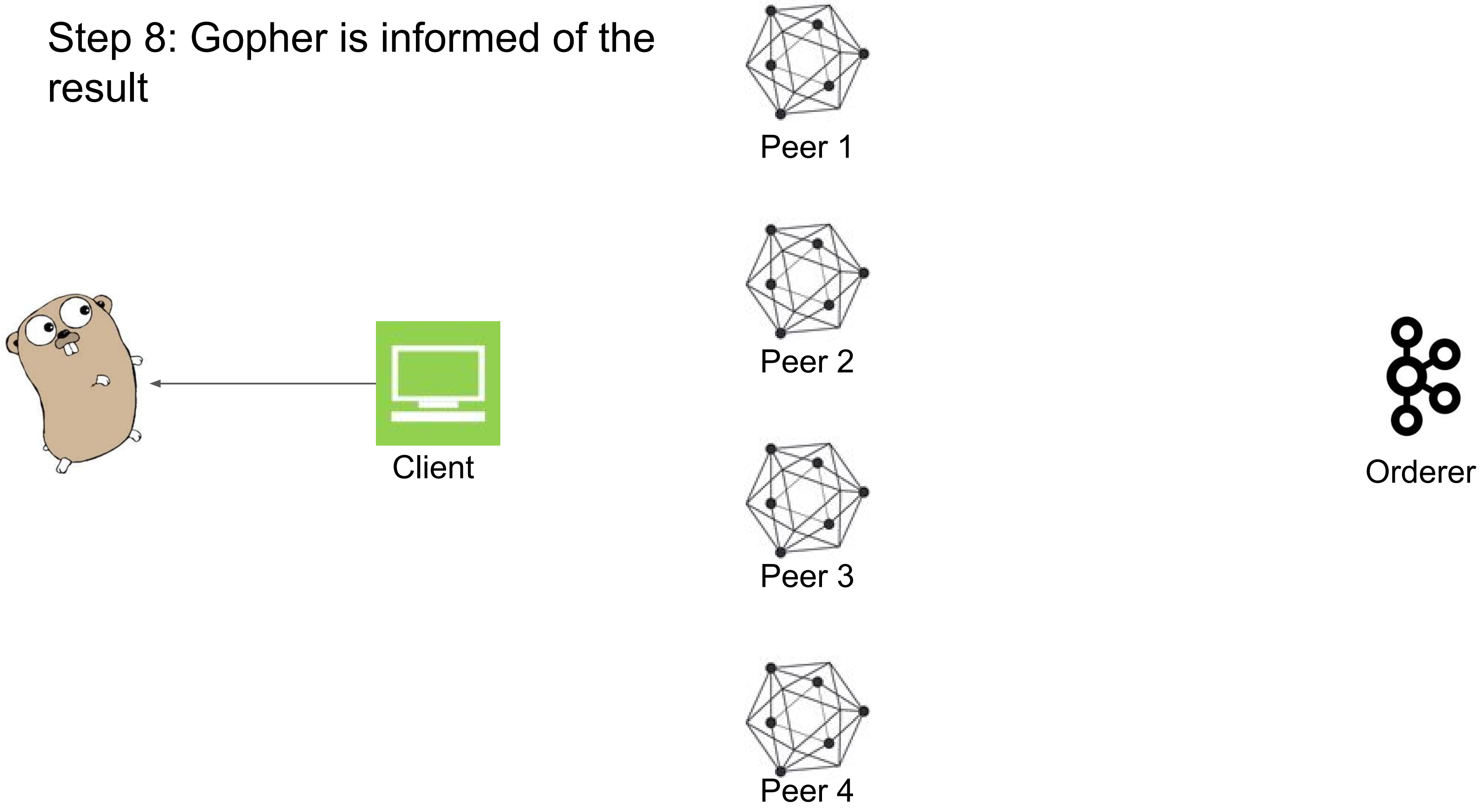
Step 6: Orderer creates new world state. Tells Peers to update.



Step 7: Orderer notifies the client the state has been updated.



Step 8: Gopher is informed of the result



Final Vote Count!

```go
//GetVoteCount returns the vote count for the candiates
func (t *VoteChaincode) GetVoteCount(stub shim.ChaincodeStubInterface) pb.Response {
    //make sure voting period has ended
    active := checkVotingPeriod()
    if active {
        return shim.Error("Voting Period is still active")
    }
    //get votelist for candidate A
    listA, err := stub.GetState("candidateA") //handle error
     //get votelist for candidate B
    listB, err := stub.GetState("candidateB") //handle error

    //Marshal response
    finalresult := result{
        CandidateA: len(listA),
        CandidateB: len(listB),
    }
    response, err := json.Marshal(finalresult) //handle error
    return shim.Success(response)
}
```

In Summary

Shared Ledger

Smart Contracts

Consensus

Synchronization

These are key components to many of the Distributed Ledger/ Blockchain Protocols

More Info on Hyperledger Fabric

Github: https://github.com/hyperledger/fabric

Docs: http://hyperledger-fabric.readthedocs.io/en/latest/

Thank You!