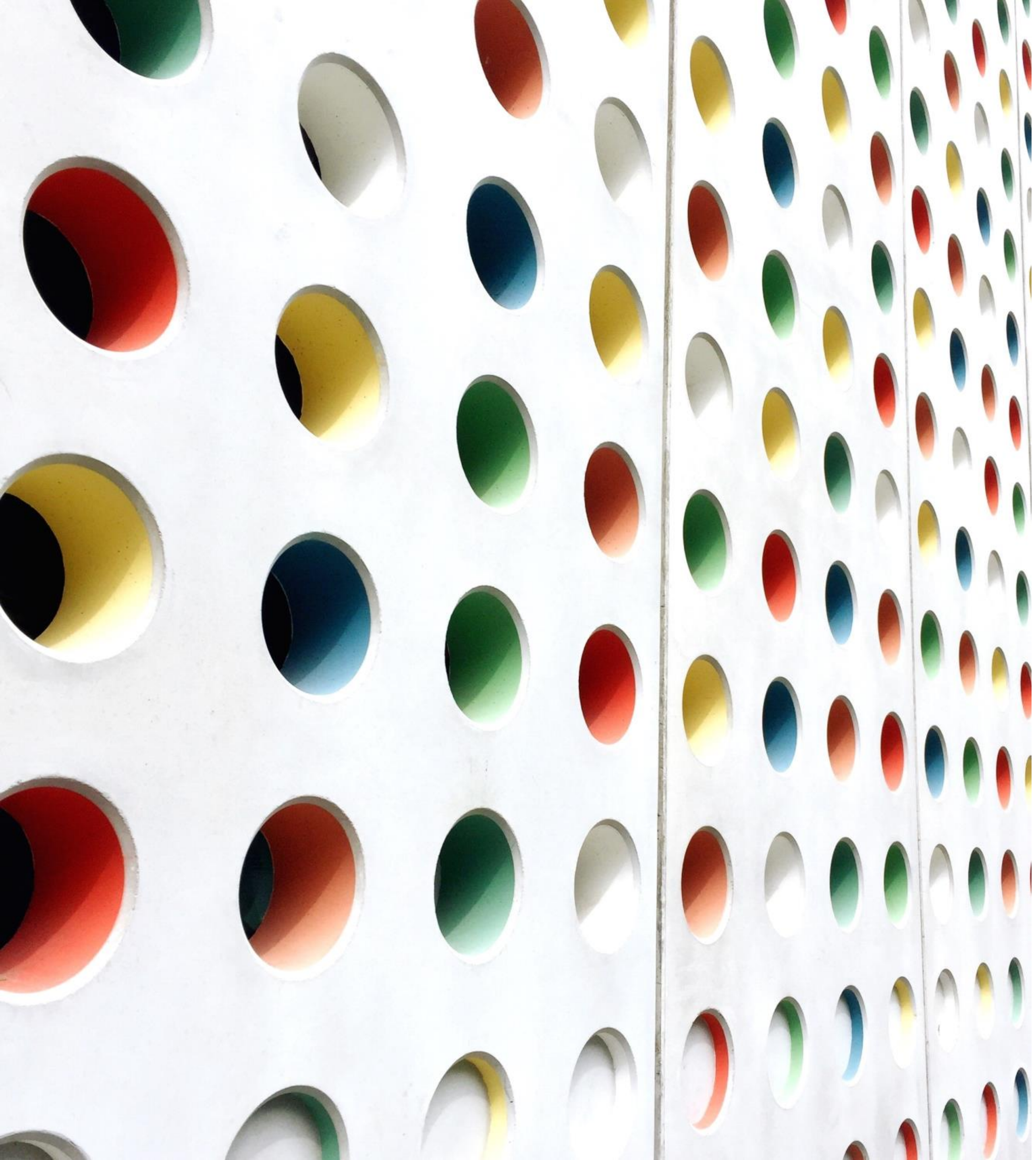


Liars Lie

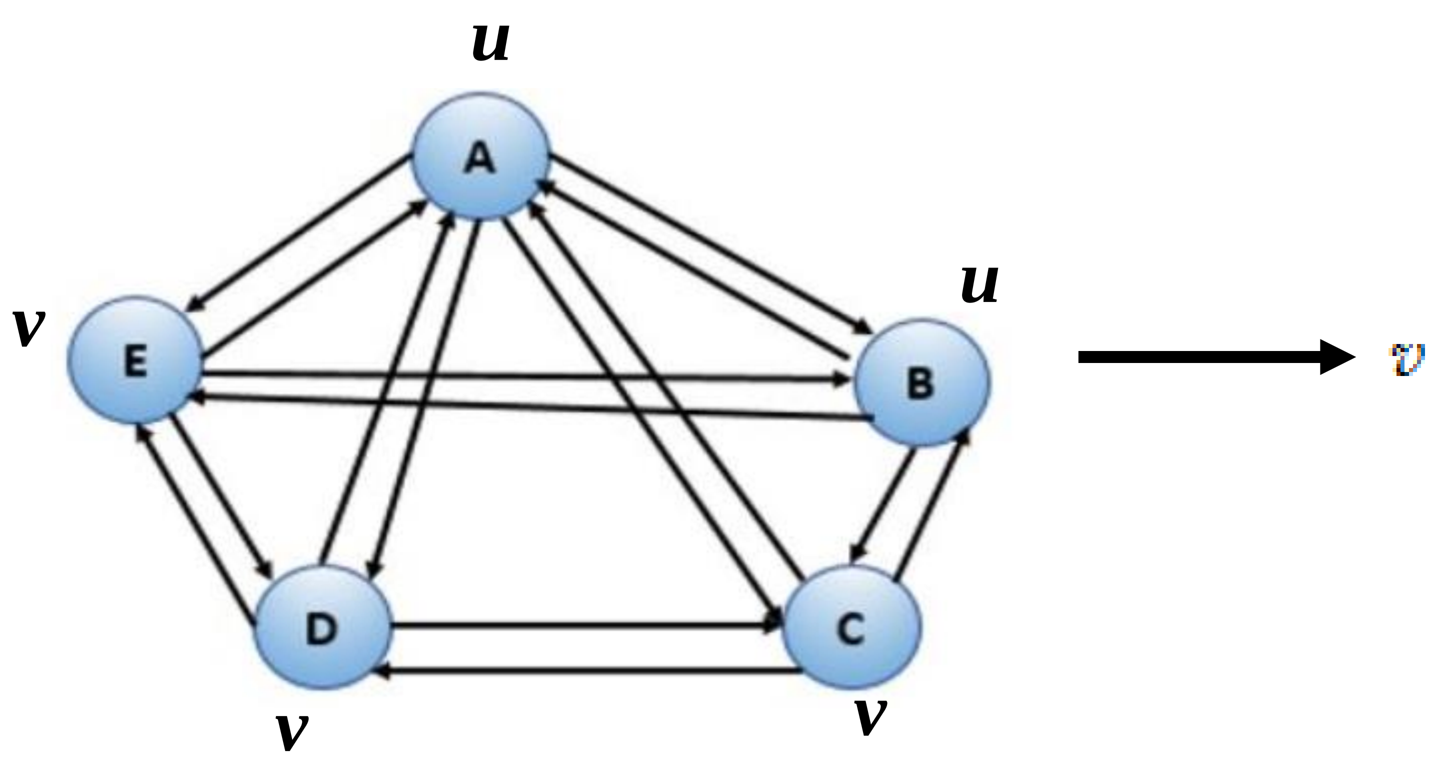
VIKRAM BHATTACHARJEE

CONTENTS

PROJECT OVERVIEW

Project Description :-

The presentation describes the efforts and outcomes related to the development of the CLI tool called "liarslie" which is used to determine is to determine the true value in a network among a combination of truth-tellers and liars.



FUNCTIONAL REQUIREMENTS

Functional Requirements :-

- A network where configurable subset of agents tell the truth and will reveal the true value when asked.

- On the other hand, liars always lie and each liar will respond with an arbitrary (random) value, but always the same arbitrary value, when asked.

- The CLI should be able to query the network value of the agents.

- The CLI should compute the “true” value *v* by asking the agents their individual values through two modes “standard” and “expert”.

- The agents in the network should be able to communicate with other agents.

STANDARD MODE

STANDARD MODE: SETUP

Command setup the configuration of the game.

CMD:- start --value v --max-value max --num-agents number --liar-ratio ratio

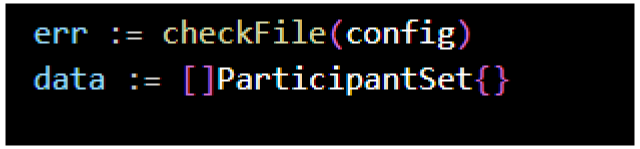
Features:-

- ✓ **Each agent should be on its own thread or process.**
- ✓ **The CLI accesses the agent info over a .config file.**
- ✓ **The .config file can be appended by new agent information in expert mode.**
- ✓ **The .config file does not expose the value assigned to the agent.**

STANDARD MODE: SETUP

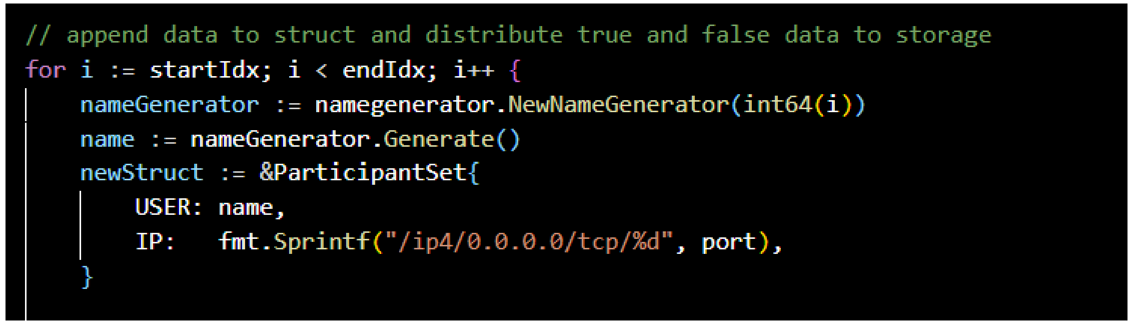
reader.go

CMD:- start --value v --max-value max --num-agents number --liar-ratio ratio



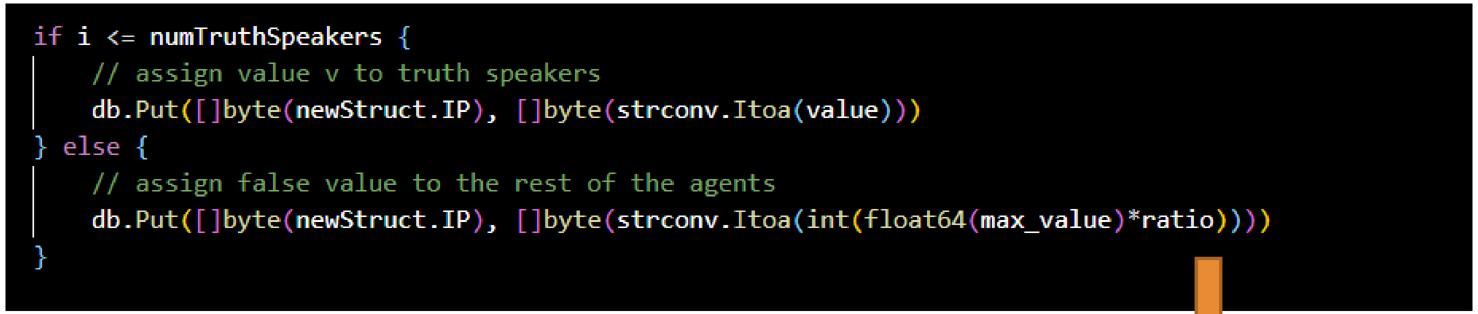
```go
err := checkFile(config)
data := []ParticipantSet{}
```

- Uses `checkfile` method to check the availability of the config file.



```go
// append data to struct and distribute true and false data to storage
for i := startIdx; i < endIdx; i++ {
    nameGenerator := namegenerator.NewNameGenerator(int64(i))
    name := nameGenerator.Generate()
    newStruct := &ParticipantSet{
        USER: name,
        IP:   fmt.Sprintf("/ip4/0.0.0.0/tcp/%d", port),
    }
```

- The start and end of the sequence is generalized to allow appending of data under both standard and expert modes.



```go
if i <= numTruthSpeakers {
    // assign value v to truth speakers
    db.Put([]byte(newStruct.IP), []byte(strconv.Itoa(value)))
} else {
    // assign false value to the rest of the agents
    db.Put([]byte(newStruct.IP), []byte(strconv.Itoa(int(float64(max_value)*ratio))))
}
```

- While appending to config, an embedded KV store/vault that appends agent names to initially set values in the network.

agents.json **[{"USER":"divine-cloud","IP":"/ip4/0.0.0.0/tcp/11581"},{"USER":"…..**

STANDARD MODE: PLAY

Command to play liarslie in standard mode.

CMD:- play

Features:-

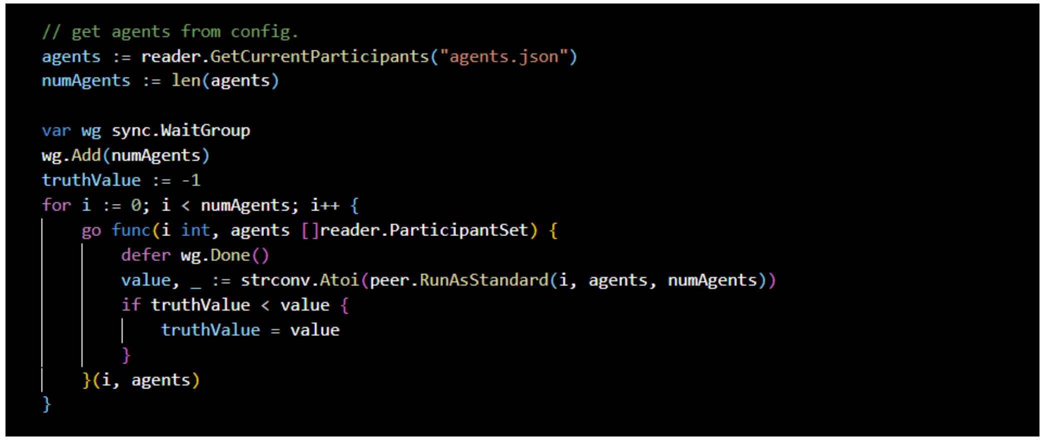
- ✓ **The same command can be invoked multiple times.**
- ✓ **Upon each invocation, the client should read the agents.config file (which is to be treated as coming from an external service),**
- ✓ **The CLI can connect to the agents, play a round of the game, and print the network value v.**

STANDARD MODE: PLAY

The command reads the config and connects to the agents to play a round of the game, and print the network value v.

standard.go

CMD:- play

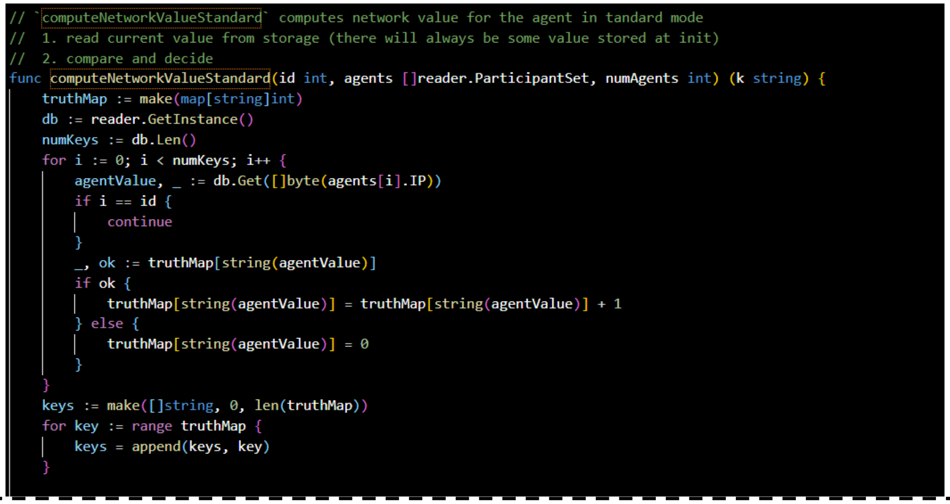


```go
// get agents from config.
agents := reader.GetCurrentParticipants("agents.json")
numAgents := len(agents)

var wg sync.WaitGroup
wg.Add(numAgents)
truthValue := -1
for i := 0; i < numAgents; i++ {
    go func(i int, agents []reader.ParticipantSet) {
        defer wg.Done()
        value, _ := strconv.Atoi(peer.RunAsStandard(i, agents, numAgents))
        if truthValue < value {
            truthValue = value
        }
    }(i, agents)
}
```

- Read Data from config
- Compute Network Value

discovery.go

compute network value



```go
// `computeNetworkValueStandard` computes network value for the agent in tandard mode
//  1. read current value from storage (there will always be some value stored at init)
//  2. compare and decide
func computeNetworkValueStandard(id int, agents []reader.ParticipantSet, numAgents int) (k string) {
    truthMap := make(map[string]int)
    db := reader.GetInstance()
    numKeys := db.Len()
    for i := 0; i < numKeys; i++ {
        agentValue, _ := db.Get([]byte(agents[i].IP))
        if i == id {
            continue
        }
        _, ok := truthMap[string(agentValue)]
        if ok {
            truthMap[string(agentValue)] = truthMap[string(agentValue)] + 1
        } else {
            truthMap[string(agentValue)] = 0
        }
    }
    keys := make([]string, 0, len(truthMap))
    for key := range truthMap {
        keys = append(keys, key)
    }
```

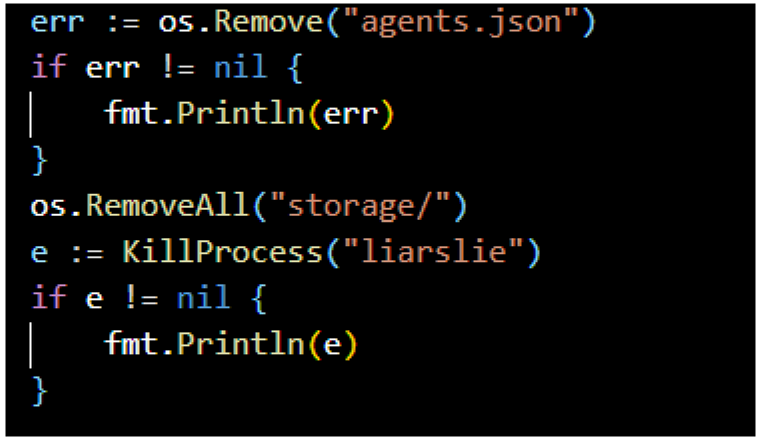
- Compare frequency of occurrence of the values from other peers.
- choose the value with highest frequency.

STANDARD MODE: STOP

Command to remove the information about stopped agents from the file and exit from the executable.

`standard.go`

CMD:- stop



```go
err := os.Remove("agents.json")
if err != nil {
    fmt.Println(err)
}
os.RemoveAll("storage/")
e := KillProcess("liarslie")
if e != nil {
    fmt.Println(e)
}
```

- `Remove all information from .json file`
- `Remove storage`
- `Kill process “liarslie”`

Features:-

✓ **Removes all artifacts from liarslie.**

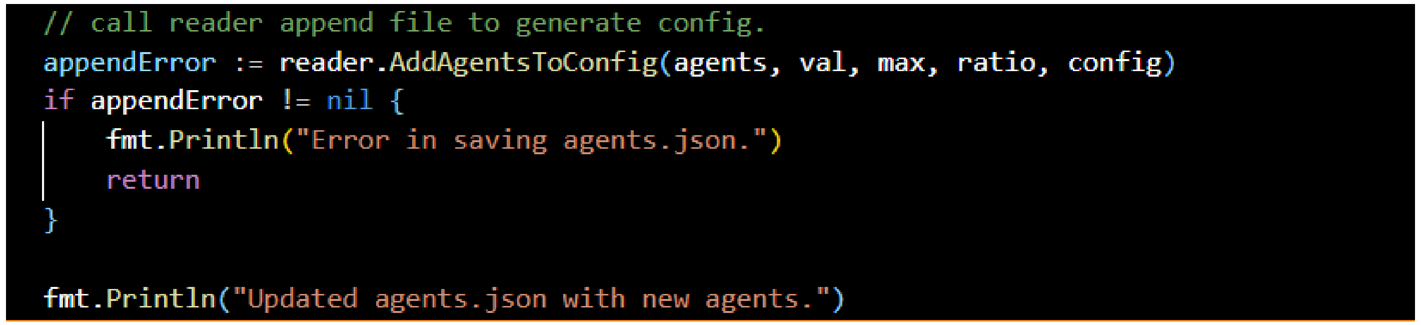
EXPERT MODE

EXPERT MODE: EXTEND

This command enables the ability to compute true network value by querying only a subset of agents.

expert.go

CMD:- extend --value v --max-value max --num-agents number --liar-ratio ratio



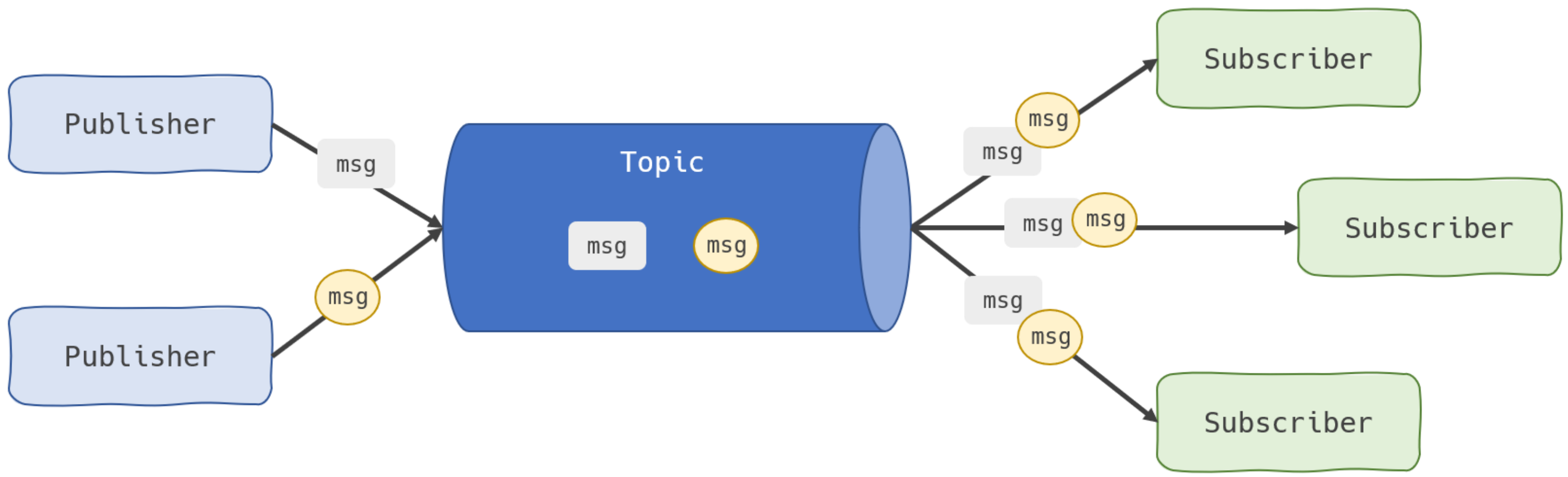
```
// call reader append file to generate config.
appendError := reader.AddAgentsToConfig(agents, val, max, ratio, config)
if appendError != nil {
    fmt.Println("Error in saving agents.json.")
    return
}

fmt.Println("Updated agents.json with new agents.")
```

- **Append additional agents by `num_agents` using `checkfile` method and assign values to the new entries using `AddAgentsToConfig` in reader.**
- **Launch the agents into the network.**
- **Each host starts gossiping to its peers and starts updating its vault based on votes from peers through libp2p pub-sub.**

Features:-

✓ **Checks for the existence of config and appends new information to agents on existing config.**

EXPERT MODE: EXTEND (PUB-SUB ARCHITECTURE)



Definition:-

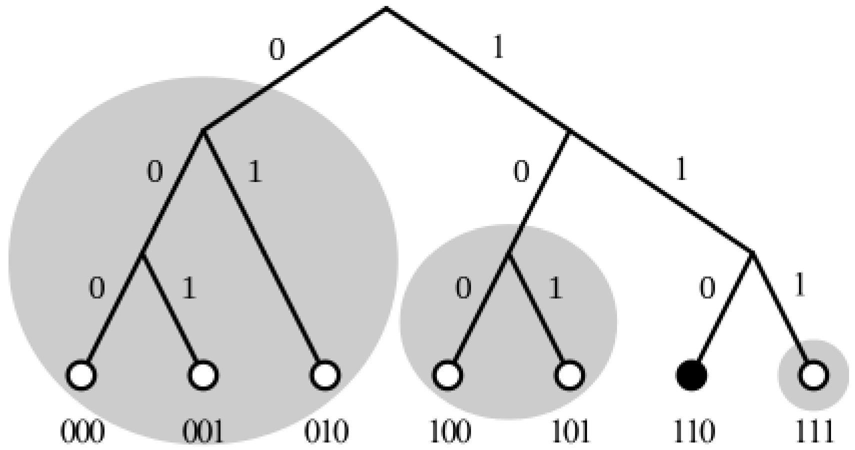
- **Subscribers declare their topics of interest; publishers publish in one of the system's topics. The pub/sub system then matches the two and delivers new messages (or more commonly called events) to all subscribers under a topic [1].**

Disadvantages:-

- **Reliable Delivery**
- **Load Balancing**
- **Resilience & Resource-Efficiency**
- **Scalability**

$$\begin{tikzcd} \textbf{Disadvantages} \arrow[r] & \textbf{GossipSub} \arrow[r] \arrow[dr] & \textbf{Ambient Peer Discovery with DHT [2]} \\ & & \textbf{Message Propagation} \end{tikzcd}$$

EXPERT MODE: EXTEND (KADMELIA DHT)



Kademlia uses Tree-based routing.

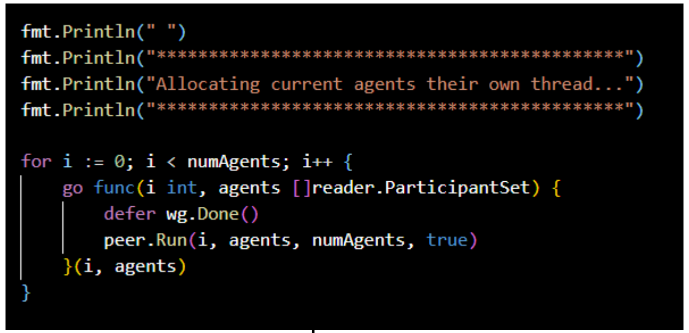
Benefits:-

- **Operation cost is as low as other popular protocols**
  Look up: O(logN), Join or leave: O($\log^2$N) [3]
- **Fault tolerance**

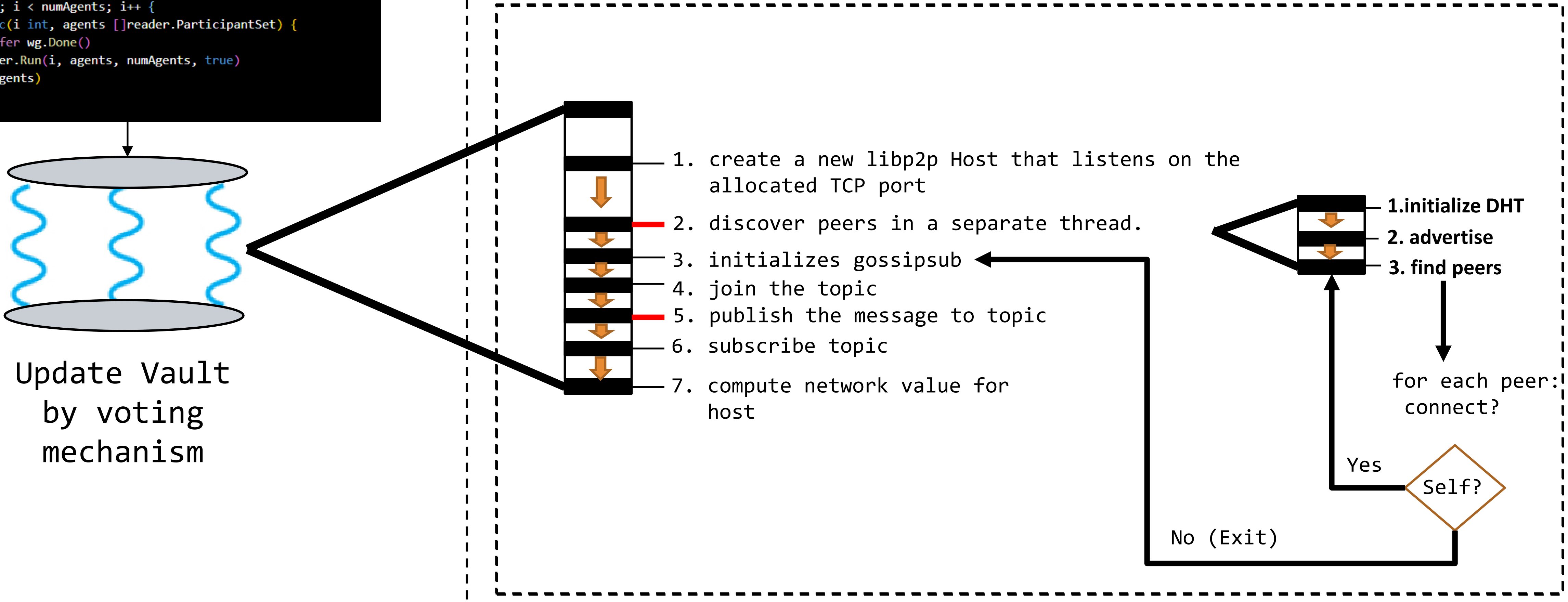
EXPERT MODE: EXTEND (CONTINUED)

CMD:- extend --value v --max-value max --num-agents number --liar-ratio ratio

expert.go

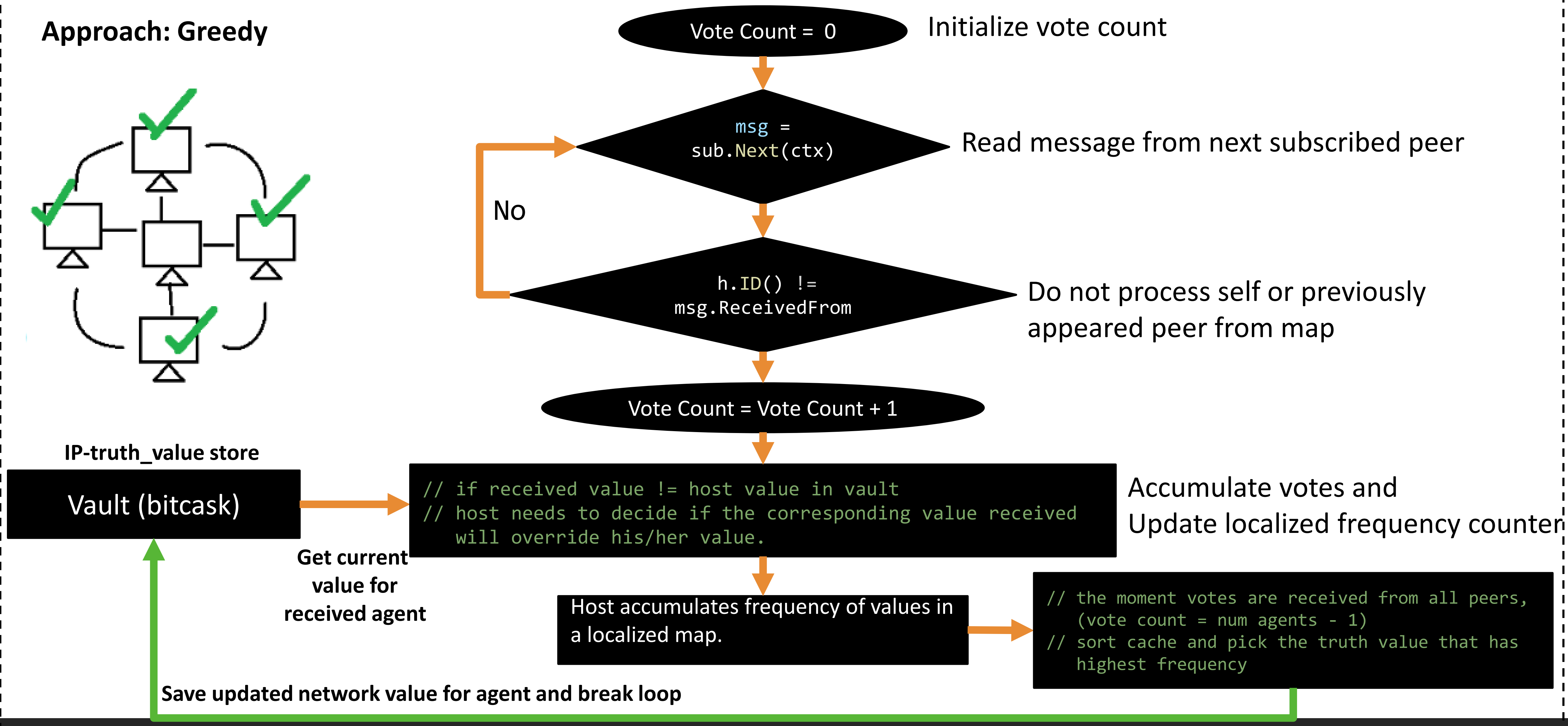


```
fmt.Println(" ")
fmt.Println("**********************************************")
fmt.Println("Allocating current agents their own thread...")
fmt.Println("**********************************************")

for i := 0; i < numAgents; i++ {
    go func(i int, agents []reader.ParticipantSet) {
        defer wg.Done()
        peer.Run(i, agents, numAgents, true)
    }(i, agents)
}
```


Update Vault by voting mechanism
discovery.go
1. create a new libp2p Host that listens on the allocated TCP port
2. discover peers in a separate thread.
3. initializes gossipsub
4. join the topic
5. publish the message to topic
6. subscribe topic
7. compute network value for host
1.initialize DHT
2. advertise
3. find peers
for each peer: connect?
Self?
Yes
No (Exit)

EXPERT MODE: PLAYEXPERT (VOTING MECHANISM)

Approach: Greedy


Vote Count = 0
Initialize vote count
No
Read message from next subscribed peer
Do not process self or previously appeared peer from map
Vote Count = Vote Count + 1
IP-truth_value store
Vault (bitcask)
Get current value for received agent
Accumulate votes and Update localized frequency counter
Host accumulates frequency of values in a localized map.
Save updated network value for agent and break loop

```
msg =
sub.Next(ctx)
```

```
h.ID() !=
msg.ReceivedFrom
```

```
// if received value != host value in vault
// host needs to decide if the corresponding value received
   will override his/her value.
```

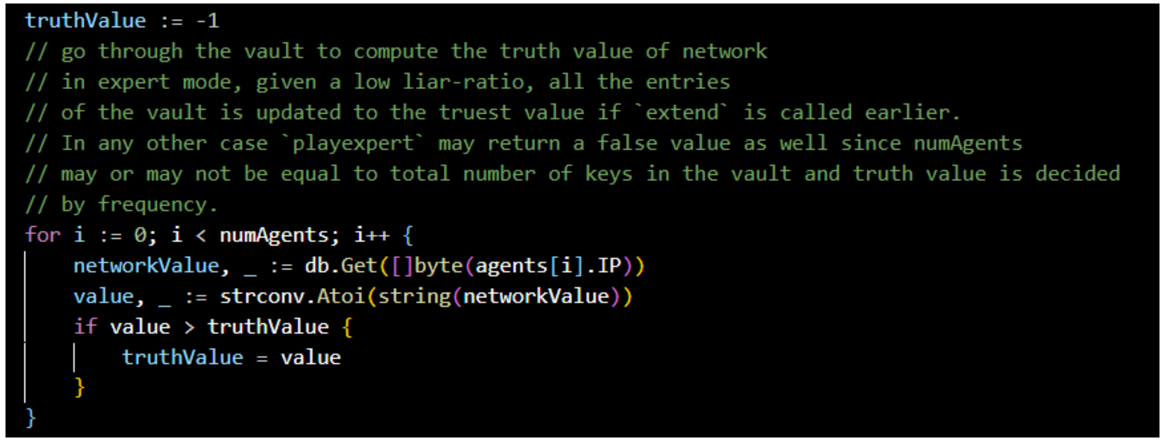
```
// the moment votes are received from all peers,
   (vote count = num agents - 1)
// sort cache and pick the truth value that has
   highest frequency
```

EXPERT MODE: PLAYEXPERT

Command to query network value by only querying only a subset of the network.

expert.go

CMD:- playexpert --num-agents number --liar-ratio ratio



```
truthValue := -1
// go through the vault to compute the truth value of network
// in expert mode, given a low liar-ratio, all the entries
// of the vault is updated to the truest value if `extend` is called earlier.
// In any other case `playexpert` may return a false value as well since numAgents
// may or may not be equal to total number of keys in the vault and truth value is decided
// by frequency.
for i := 0; i < numAgents; i++ {
    networkValue, _ := db.Get([]byte(agents[i].IP))
    value, _ := strconv.Atoi(string(networkValue))
    if value > truthValue {
        truthValue = value
    }
}
```

- Similar to standard `play` query the vault to get the value from the selected set of agents.

EXPERT MODE: PLAYEXPERT

CMD:- playexpert --num-agents number --liar-ratio ratio

Features:-

- ✓ **Determines the network value by directly querying at most the given number of agents.**

- ✓ **Liars cannot modify their own value, but could tamper with other types of messages. This is taken care of by:-**
  - ✓ **Value is computed till all votes are received from all peers.**

- ✓ **The agents can communicate over TCP.**

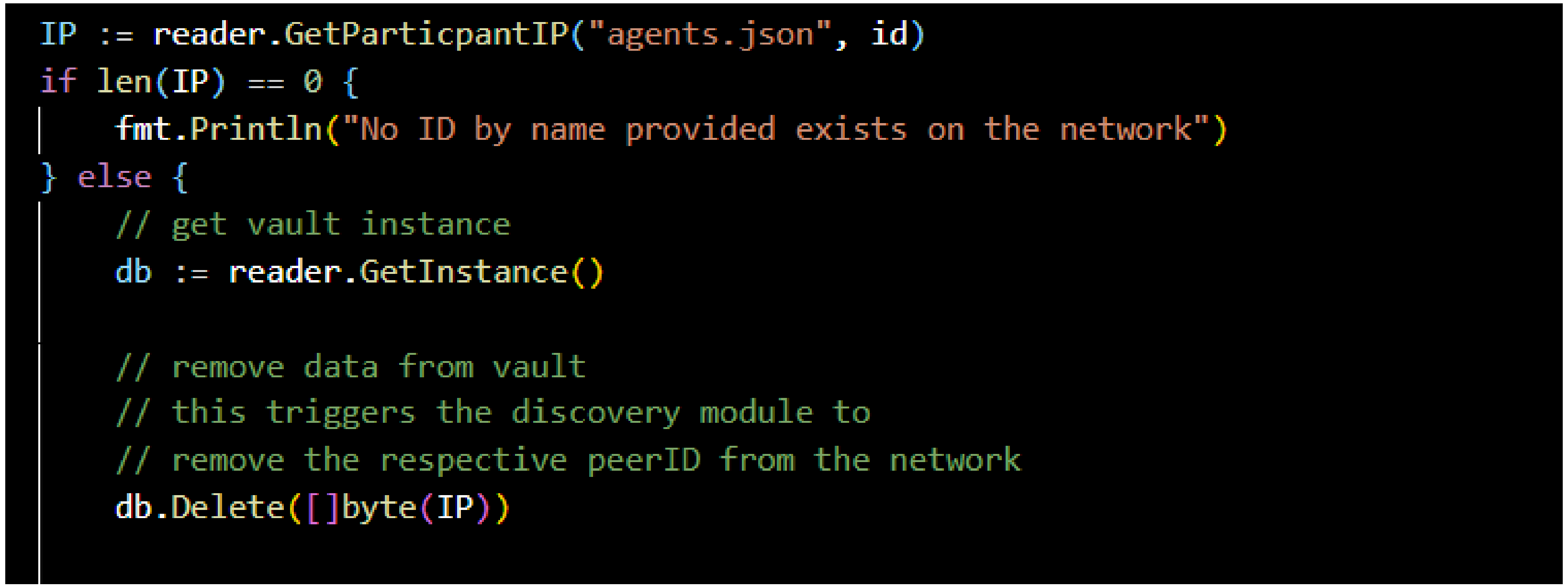
- ✓ **The rest of the features are similar to standard play mode.**

EXPERT MODE: KILL

Kill prunes the network.

expert.go

CMD:- kill --id id



```go
IP := reader.GetParticpantIP("agents.json", id)
if len(IP) == 0 {
    fmt.Println("No ID by name provided exists on the network")
} else {
    // get vault instance
    db := reader.GetInstance()

    // remove data from vault
    // this triggers the discovery module to
    // remove the respective peerID from the network
    db.Delete([]byte(IP))
```

CONCLUSION

- The “liarslie” CLI following the descriptions for standard and expert mode was developed and the progress was communicated.

- In standard mode the CLI can directly query the agents and get the network value without them having to communicate between themselves.

- In expert mode the network can be extended and more agents can be added.

- In expert mode, the network value can be queried from a set of agents.

- In expert mode, the true network value is computed using a pub/sub protocol followed by a voting mechanism where the vault is updated after each agent receives votes from its peers.

REFERENCES

[1]. Vyzovitis, Dimitris, et al. "GossipSub: Attack-resilient message propagation in the Filecoin and ETH2. 0 networks." arXiv preprint arXiv:2007.02754 (2020).

[2]. Golang libp2p Source: https://github.com/libp2p/go-libp2p

[3]. Meskanen, Tommi, and Valtteri Niemi. "On DHT, libp2p and HELIOS Platform." (2021).

THANK YOU