**main:**

**1.ParsedProperties(args):**read file to the properties

**2.Configuration.setConfig(Porperty):**get protocol and put them to the protocol container. The structure of protocol container is: Map<String,Integer>. Each Protocol has a protocol id(pid) and pid will be linked to the node. The protocol order will be sorted. Of course, protocols can be chosen by add include.protocol in the configuration file.

**3. execute CDSimulator.nextExperiment() or EDSimulator.nextExperiment() according to the type of simulation.**

**4.if CDSimulator.nextExperiment():**

**a. Network.reset():**

Remove all the nodes in the network, reconstruct network.size (defined in the configuration file) nodes and add them to the network.

When constructing a node, we should assign the node id, protocols of the node and construct the defined protocol (begin with protocol) in the configuration. (A node will have a node id, several protocols). While the program constructing the protocol object, the construct function of the protocol will read the protocol parameters in the configuration file. Here the function just initializes the protocol objects and does nothing else.

Usually, one node will have more than one kind of protocols. The basic one is link protocol which tells the node how to connect to its neighbors, how to change its neighbors and how to find its neighbors.

The other protocols are concerned with specific application such as average function, max value function, etc. This kind of protocol will use the linkable protocol defined above to find the neighbors. After find the neighbors the protocol will do predefined activities according to the specific application.

Of course, we can use include in the configuration file to choose the protocol objects we want to construct.

After constructing a node, the program will clone network.size nodes with the node constructed just now.

**b. runInitializers ()**

Each initialization object has a protocol. When constructing a new initialization object, the program will assign a protocol id(pid) to the initialization object, and some other parameters.

The initialization object is used to build the network topology and the properties of the existed nodes, which distinguish themselves from the control objects which control and observer the process of simulation.

The initialization object can be chosen with include in the configuration file.(include.init xxx xxx).

**C().loadControls()**

After initialization, the program will load control objects to control the simulation. In the configuration file, the control objects will start with string “control”. The control object also have the protocol id, and it also have the method “excute()”. The control object will execute specific activities to the network nodes, network topology and calculate some values we are interested in which can be output to the standard output or files.

Along with control object construction, a schedule object will be constructed too. Schedule object can active control object.

**D():execute all the controller**

After loadControls(), the program will execute the controller. Each object implemented Control interface will have a execute function. The control objects usually change the structure of network, execute specific protocol and calculate value of the nodes in the network.

**E():FullNextCycle.excute()**

When we use cycle driven simulation, each simulation will construct a FullNextCycle object which will execute protocols of each node in the network(eg, AverageFunction, SimpleNewscast which extended from CDProtocol).

The protocol based on the cycle driven simulation should extend interface CDProtocol and should implement nextCycle function.

When the protocol executes the nextCycle function, it need to know the specific linkable protocol it has (which is defined in the configuration file). In order to get the linkable protocol quickly, the program stores the corresponding linkable protocol for each cycle drive based protocol in the static array(implemented in the fastconfig).

**F():Control.execute();**

Each control object has a execute function which can observe and modify the simulation. Eg, the Control object can calculate the average value of all the nodes in the network or remove some nodes in the network.

Output of the Control Object: usually the Control object will output the observation result to the standard output. The format of the output is:

***Minimum, maximum, number of items, average, variance, number of minimal items, number of maximal items.***

5. if EDSimulator.nextExperiment();  
 if the simulation is based on the event driven, the EDSimulator.nextExperiment() function will be executed.

In the event driven simulation, all functions are executed by event. There are several kinds of event, such as control event which is used to control the network or observe the status of network and executable protocol event used for communication between different nodes or cycle driven event can also exist in the network which is used to execute specific activities periodically. All the events are stored in the heap, a sorted queue according to the priority of the items in the queue. Each time the program will retrieve an event will execute specific function according to the event type. Each event has a time which addresses the time when it will be processed. If the time is greater than the endtime then the event will not added to the heap. When the heap is empty the program will stop.

1. Network.reset(): a function is the same as that in the cycle driven simulation, which is used to construct nodes and corresponding protocols of the nodes in the network
2. runInitializers(): the function is used to initialize the network and customaries protocols. Eg, in Kademlia protocol, we need to assign a random 32bits binary number to each node in the network.
3. scheduleControls(): sheduleControls function in the event driven simulation is different from that in the cycle driven simulation. In the event driven simulation, we will treat the Controller as a control event and add the Controller to the event heap. The delay of the event is one time unit.
4. After adding control event, the program will execute event retrieved from event heap. There are three kinds of event: Control event, nextCycleEvent(used like cycle driven simulation) and Event driven simulation event. The control Event is used to modify and observe the network status and structure and produce protocol event. In Kademlia protocol, TrafficGenerator Controller is used to produce find node event. Turbulence Controller is used to add or remove nodes from the network. KademliaObserver Controller is used to observe the network status and conduct statistic and report the statistic results. Each Controller will have a scheduler used to schedule when to execute the specific controller. The controller has a parameter named step used to specify the execution time of the event.
5. In the kademlia protocol, we first send a find node request and the protocol will call find(Message m, int pid) function. In find(Message m, int pid), the function create a findoperation and add all neighbors in the source node’s neighbor table to the operation. Then the function will send alpha requests to the neighbors concurrently. The kademlia protocol uses UniformRandomTraffic which produces uniform random delay. In the example, the delay is 100.
6. The timeout event: when the protocol send a message to the target node, it will wait for the response. Each message has a unique message id. We can use message id to determine whether a message has got its response. But we cannot wait for the response always, so we must specify a timeout mechanism to stop the wait for the response. In Kademlia protocol, when the protocol sends a message it will send a timeout event which contains the message id of the message to the event heap, too. The protocol will store a message map with message id as key. When the protocol receives the response it will remove the message from the message map with the message id. And If the protocol receives the timeout event, it will check whether the message id contained in the timeout event is in the message map. If it contains, then it means the message is timeout, and the protocol will send the message again.

***Protocol relation:***

***protocol.link: node connection.***

***Protocol.avg: application protocol. It has a linkable parameter used for finding neighbors. It has a transport parameter which shows that the protocol uses a transport protocol for node connection. In this case, the avg protocol use a UnreliableTransport transport protocol which could lose some packets and the UnreliableTransport transport protocol uses the UniformRandomTransport transport protocol which may randomly choose delay time to send the packets.***

**Data Structure:**

1. **There is one network in the program.**
2. **The network has many nodes and each node has at least one protocol.**
3. **There are two type protocols in the network (for now, I just know there are two type). One is link protocol used to connect each node in the network and construct the network topology. The other one is application related protocol based on cycle driven simulation or event driven simulation, which is used to realize detailed application such as calculation, transportation, etc.**
4. **When protocols work in the work, we need to know the status of each node and sometimes we may want to change the status of the nodes or properties of the network, so we need have some Control Objects to implement these functions.**