TrustSim: A Decentralized Reputation and Trust Model Simulator

S.Venkatesan
Department of IT
IIIT Allahabad, India
venkat@iiita.ac.in

Sandeep Kumar Shukla
Department of CSE
IIT Kanpur, India
sandeeps@cse.iitk.ac.in

Yuvaraj Rajendra
Department of IT
IIIT Allahabad, India
pcl2016003@iiita.ac.in

I. Appendix

A. *Structure*

The tool is developed using the following arrays.

- Node Main - This table is to store the resource request and response details from and to the neighboring nodes. The structure of the NodeMain array is as follows

  – [0] Entry Type (*entrytype*) : This is to refer the resource request or response. The value 1 indicates the request and value 2 indicates the response. The value 3 for the self-response or sending resource to neighbours without request.
  – [1] Entry Validity (*entryvaldity*) : This is to refer whether the query is expired or not. The value -1 refers expiry and value 1 refers non-expiry. This is required to indicate that the request is already addressed.
  – [2] Receiver Identity (*receiverid*) : This is to refer the destination node that is request or response received node identity.
  – [3] Unique entry identity (*uniqueid*) : This is to refer the unique identity of the record for serving the service in future at both the sender and receiver node.
  – [4] Sender Identity (*senderid*) : This is to refer the identity of the sender of the request or response.
  – [5] Message Identity (*messageid*) : Identity of the message that is requested or responded.
  – [6] Status Identity or Value (*statusid*): This refers the status identity or value of message that is requested or responded.
  – [7] Node Type (*nodetype*): This is to record the (sender that is array 4 node) node type that is genuine, malicious, disguise or sybil.
  – [8] Present Session (*presentsession*): This is to record the session of the simulation.
  – [9] Decision (*decision*): This refers the decision of the message whether the response is accepted or not. The decision are accept(1), reject(-1) and nodecision(0). - At present kept idle
  – [10] Response Identity (*responseid*): This is to record that who has responded to a request or who sent the non-request message that is a response without a request. It is set as 2 for response for the request can be -1/0
  – [11] Message Integrity (*messageintegrity*): This is to record the integrity of the message that is posted by a node. This data can be fetched from the node service array. The correct message refers 1 and incorrect refers to -1.
  – [12] Node Location (*nloction*): This is to record location of the node in the table.
  – $[totalattribute + 1]$ - first trust model decision, + 2 second trust model decision, and so on. The decision are accept(1), reject(-1) and nodecision(0).

- Fact - This to record the current fact of the environment those are posted by entities or nodes in the network. User can decide the fact structure in addition to the required structure. The structure of fact array is as follows.

  – [0] Fact Type (*ftype*) : Identity the type of the fact. If it is 1 then it is posting the fact. If it is 2 posting the fact to take the vote so that the sender will be given negative vote. The value -1 indicates the entry is free.
  – [1] Node Identity (*node*) : Identity of the node that posts the facts on the fact table.
  – [2] Message Identity (*messageid*) : Identity of the message that is posted in the fact table.
  – [3] Node Type (*nodetype*): This is to record the node type that is genuine, malicious, disguise or sybil.
  – [4] Present Session (*presentsession*): This is to record the session of the simulation.
  – [5] Count (*count*): This is to record the number of same facts posted by the node.
  – [6] Status Identity (*statusid*): This refers the status identity of the message. This data can be fetched from the node service array.
  – [7] Bloom identity (*bloomid*): This refers the bloom.
  – [8] Target identity (*targetid*): This refers the identity of may be the attacker.
  – [9] Valid vote count (*valid*): This refers to maintain the count that the attack reported by a victim is valid.
  – [10] Invalid vote count (*invalid*): This refers to maintain the count that the attack reported by a victim is invalid.

  - [11]Node Location (*nloction*): This is to record location of the node in the table.
  - User can add their own attributes.
- Opinion or Votes [*tcompute*]- This array records the opinion or votes given by a node to another node. Using the values in this table, the score will be computed according to the user defined function. This is the four dimensional array to support each and every trust model. The structure of opinion array is as follows.
  - [0] Vote Type (*votetype*) : This refers to the vote type. For example, vote (1), incentive (2), etc.
  - [1] Vote Value (*votevalue*) : The equivalent vote value of the vote type. 0 for negative vote.
  - [2] Voter (*voter*): This is to record the voter.
  - [3] Present Session (*presentsession*): This is to record the session of the simulation.
- Trust Score - This array records the nodes computed score and this will be updated regularly that is if there is any entry to the Opinion array then it will be updated. This will be the three dimensional array to support multiple trust models. The structure of trust score array is as follows.
  - [0] Present Score : This is to store the score of a node in the present session for the trust model that is insider attack.
  - [1] History: This is to store the previous history of the node.
  - [2] Attack: This is to store the number of attacks by a node.
  - [0] Present Score : This is to store the score of a node at any time for trust model 1 that is decentralized.
  - [1] votecount: Count for voting.
  - [2] miningcount: Count for mining.
  - [3] factcount: Count for fact contribution.
  - [4] privotevalue: Private Count or total votevalue.
  - [5] introducecount: Count for new node introducing.
  - [6] pubvotevalue: Public Count or total votevalue.
  - [0] Present Score : This is to store the score of a node at any time for trust model 3 that is TNASL.
  - [1] votecount: Count for voting.
  - [2] belief: value of belief.
  - [3] disbelief: value of disbelief.
  - [4] uncertainty: value of Uncertainty.
- Node Resource or Node Service- This maintains the availability of resources for each node and it will be updated if any new resource comes and that is acceptable. For every message type there will be an entry. The structure of the NodeResource array is as follows
  - [0] Service Availability : The value 1 indicate applicable and -1 not applicable
  - [1] Node Type: This stores the type of node, 2 indicates Sybil, 1 indicates Disguise, 3 indicates Genuine and 0 indicates Malicious.
  - [2] Status : This refers the status identity. The resource may refer to different things. For example, road open or closed is also a resource available with a node and that can be shared with other nodes. The status 0 refers open and 1 refers closed however user can have their own.
  - [3] Service Quality : This refers whether the service is correct or incorrect service; for example, -1 incorrect service, 1 correct service;
  - [4] Ownership : This refers whether the service is its own or received by some neighbours; This will have the identity of the owner.
  - [5] Node Location (*nloction*): This is to record location of the node in the table.
  - $[totalattribute + trustmodels]$ - This is to store whether the resource is updated based on this trust model or not (service value otherwise initially -1).
- Node Neighbor - This maintains the neighbors of the nodes. The number of attributes for the table is upto the user but in general it will be one less than the total number of nodes in the network. This table gets initialized based on the minimum neighbors later it get updated while process. The structure of the table is as follows.
  - [0] Total Neighbors : The number of neighbors for the respective node
  - $[1..n-1]$ Neighor Id : Identity of the neighbors
- Node Service Table - This maintains the possible services available in the network. The structure of the `nodeservicetable` array is as follows
  - [0] Message Status (*mstatus*): This refers the status identity of the message.
- Node Active Table - This maintains the activation of a node. The structure of the `nodeactive` array is as follows
  - [0] Active (*active*): Refers whether node is active (1) or not (0)
  - [1] Introducer (*introducer*): The node that introduces it.
  - [2] Nodetype (*nodetype*): The type of the node (0) malicious, (1) disguised, (2) sybil and (3) genuine.
  - [3] penalty (*Penalty*): This indicates the penalty is made already for the introduce or not (0) not, (1) already penalized.
  - [4] Node Location (*nloction*): This is to record location of the node in the table.
  - [5] Node Group (*nodegroup*): This is to refer whether a node is private (1) or public (2)
- Colluded Array Table - This maintains the details for the collusion activity. The structure of the `colludearray` array is as follows
  - [0] Type (*Type*): Refers the type of activity (1) attack claimvote posting; (2) facto posting; (3) indicates normal voting.
  - [1] Entry index (*Index*): This refers to index of the table if the above type is (1) , message identity in case the above identity is (2), target node in case of type (3)

  - [2] Neighbor (*Neighbor*): This refers to the neighbor identity if the type is (2), votetype (positive or negative) in case of type (3)
  - [3] - This refers the vote value if the type in index 0 is (3)
  - [4] - This refers message identity if the type in index 0 is (3)
  - [5] - This refers trust mode if the type in index 0 is (3)
- Sybil Array Table - This maintains the details for the collusion activity. The structure of the `Sybilarray` array is as follows and same as collude array.
  - [0] Type (*Type*): Refers the type of activity (1) attack claimvote posting; (2) facto posting; (3) indicates normal voting.
  - [1] Entry index (*Index*): This refers to index of the table if the above type is (1) , message identity in case the above identity is (2), target node in case of type (3)
  - [2] Neighbor (*Neighbor*): This refers to the neighbor identity if the type is (2), votetype (positive or negative) in case of type (3)
  - [3] - This refers the vote value if the type in index 0 is (3)
  - [4] - This refers message identity if the type in index 0 is (3)
  - [5] - This refers trust mode if the type in index 0 is (3)

B. Functions

The following are the various functions used in the simulator.

`func randomrangeint(from int, to int) int` - This is to find the integer random number between the range.
`func randomrangefloat(from float64, to float64) float64` - This is to find the float random number between the range. This function returns float but actually works on integer.
`func randomignorevalue(from int, to int, ignore int) int` - This is to find the random number between the range ignoring a given number.
`function randamrangefloat()` - This is to find the float random number between the range. This function returns float but actually works on integer.
`func uniqueRand(node int)` - This is to assign the unique random neigbours for each node at the initial stage.
`func nodeinitialize()` - This function initializes the nodes entry in the node table in the beginning of the simulation.
`func resource()` - This function allocates the resource to each every node in the beginning. A malicious node gets the malicious resource, a disguised node gets the randomized malicious or genuine resource, a sybil node gets the malicious resource and a genuine gets the genuine resource.
`func updateresource(nodei int, nodesend int, serviceid int, servicevalue int, messagecorrect int, nodetype int)` - This function helps to update the available resource of the nodes.
`func finalneighbors()` - This is to write the final neighbours of each node in the imagefinalfile.
`func neighbourallot()` - Allots the random neighbours by calling the uniqueRand function.
`func factinitialize()` - This function initializes the fact table.
`func autoserviceupdate()` - This function updates the service of each node in random period.
`func factadditionalattri(nodei int, entryrow int, addattri int)` - This function is to enter the additional attribute in the fact table according to the requirement.
`func factentry(nodei int)` - This function is used by a node to make the entry the fact table.
`func neighborcheck(nodei int, neighborid int) bookl` - This function is to check the availability of a node in the neighbours list.
`func nodetrustinitialize()` - This function initializes the trust computation table.
`func nodemaininitialize()` - This function initializes the node's main table.
`func scoreinitialize()` - This function initializes the score table.
`func nodevotepost(voter int, votee int, votetype int, votevalue int, messageid int)` - This function is to post the vote for votee node.
`func nodeincentivepost(nodei int, incentivevalue int)` - This function is to post the incentive to a node. That is for itself.
`func nodeincentivepost(nodei int, incentivevalue int)` - This function is to post the incentive to a node. That is for itself.
`func noderesponseaction(nodei int)` - This function to perform action on the received resources. This is for voting and accepting the message. This also calls the function to update the resource table of the respective node.
`func responsecompute(nodei int, messageid int) int` - This function is to decide the fate of the message based on the majority of the response come to the node. If the majority is more then accept the message otherwise reject.
`func factcompute(messageid int) int` - This function is to decide the fate of the message based on the majority of the facts in the fact table. If the majority is more then accept the message otherwise reject.
`func noderesponse(nodei int)` - This function is for the node response activity on the request of the neighbours
`func sendmessage(nodei int)` - This function is to send response to the neighbours.
`func noderequest(nodei int)` - This function is to send the request to the neigbours to send the resource.
`func reinitialize()` - This function re-initializes

everything except the neighbours at the end of every session.
`func reinitialize()` - This function re-initializes everything except the neighbours at the end of every session.
`func nodeactivity(nodei int)` - This function is to decide the node activity.
`func resultcompute()` - This function is to compute the final result of the simulation.
`func main()` - This creates the number of goroutine equal to the number of nodes in the network.
`func sendmessageadditionalattri(nodei int, neighborid int, neighborentryid int, nodeentryid int)` - This function is to add the values for the additional attributes of the sendmessage function.
`func noderequestadditionalattri(nodei int, neighborid int, neighborentryid int, nodeentryid int)` -Tthis function is to add the additional attributes for the noderequest message
`func noderesponseadditionalattri(nodei int, neighborid int, neighborentryid int, nodeentryid int)` - This function is for additional attributes in noderesponse.
`func nodeincenpostaddattri(nodei int, entryid int)` - This function is to add the additonal attributes in the tcompute table while adding incentives.
`func nodevotepostaddattri(nodei int, entryid int)` - This function is to add the additonal attributes in the tcompute table while posting vote
`func decisioncall(nodei int, f int) int` - This function is to decide and call the respective trust function to accept or reject the message
`func decent(nodei int, f int) int` - This function is to evaluate the score based on decentralized trust model.
`func insider(nodei int, f int) int` - This function is to evaluate the score based on insider trust model.
`func scorecompute(votee int, messageid int, votevalue int)` - This function is to to call the score compute functions.
`func decen(votee int, votevalue int)` - This function is the decentralized trust model score computation.
`func inside(votee int, votevalue int)` - This function is the insider trust model score computation.
`func decisioncallresponse(nodei int, f int, messageid int) int` - This function is to decide and call the respective trust model to compute the value based on the responses.
`func decentresponsecompute(nodei int, messageid) int` - This function is compute the response function for decentralized trust model.
`func insiderresponsecompute(nodei int, messageid) int` - This function is compute the response function for insider trust model.
`func decisioncallfact(nodei int, f int, messageid int) int` - This function is to decide and call the respective trust model to compute the value based on the fact.
`func decentfactcompute(nodei int, messageid) int` - This function is compute the fact value for decentralized trust model.
`func insiderfactcompute(nodei int, messageid) int` - This function is compute the fact value for insider trust model.
`func newnodeintroduce(nodei)` - This is to introduce the new nodes in the network on the fly.
`func blockmining(nodei)` - This is to get the incentive for mining the block.
`func neighboradd (nodei int, neighborid int)` - This is to add the new neighbors.
`func factvote(nodei int)` - This function is for voting in favor or against the facts posted by victims against a sender.
`func Factvotecompute(nodei int)` - This function is called by any node in the beginning of the session to analyse the give the vote of misbehaved node. This function is for giving negative vote for the nodes based on the votes available in the fact table.
`func factvoteentry(nodei int)` - This function allows the node to make the entry in the table for a recieved attack message and look for the votes.
`func factvotecast(nodei int, messageid int, index int)` - This function is to cast the vote and to support the `factvote` function.
`func actionsupport(nodei int, target int, votetype int, votevalue int, messageid int, tmodel int)` - This function is to support the node response function to go for the colluded vote.
`func Scoreclean(nodei int, tmodel int)` - This function is to clean the score to perform whitewashing.
`func Nodescore(nodei int, tmodel int) int` - This function is to find the score of a node and return for whitewhashing decision.
`func Reinitialize()` - This function is to re-initialize the score while starting the new session.
`func locchange(nodei int)` - This function is to change the location or region of the device during the simulation.
`func Decisionintroduce(nodei int, f int, sender int) int` - This function helps to check the reputation of a node before it introduces another new node.

C. Variables and Constants

The variables and constants used in this simulation are as follows.

- `const n = 100` - Number of nodes
- `const sn = 2` - Number of Sybil nodes
- `const dn = 4` - Number of disguised nodes
- `const mn = 4` - Number of malicious nodes
- `const mnei = 4` - Minimum neighbors
- `const activities = 100` - Maximum number of activities to be performed

- `const def = 1` - Duplicate entry in fact table by a node; 0 - no duplicate and 1 - duplicate
- `const threshold = 0.5` - Threshold that you want to set for score that can be accepted. A node with more than the threshold will be trusted.
- `const nos = 4` - Number of session you want to run
- `const nra = 1` - Node response considered for accepting the message; 0 - no response permitted, 1- permitted
- `const nf = 2` - Number of trust model going to be evaluated
- `const sta = 1` Number of node service table attributes
- `const na = 12` - Number of node main table attributes
- `const nae = 0` - Number of node (main table) additional attribute
- `const Nloc = 1` - applicability of the node location and attacker on the same location; 0-no need of node location and 1 for node location as well as same location neighbor intially (later it will change) and 2 for sybil attackers node in the same location as well as includes option 1; 3 is for location required as well as sybil attacker in the same location (no need of same location neighbor)
- `const Numloc = 4` - Number of locations or regions
- `const mi = 6` - Indicates node message services or number of message services.
- `const nsa = 5` - Number of Node service table attributes that is equivalent information for every message.
- `const fa = 6` - Number of Fact table attributes
- `const fae = 0` - Number of fact table's additional attribute
- `const tsa = 2` - Number of Trust score table attributes or number of functions to be used for evaluation.
- `const tsae = 1` - Number of additional trust score attribute
- `const sca = 4` - Number of Score computation table attributes
- `const scae = 0` - Number of additional score computation attribute
- `const bsm = 1` - Enabling the ballot stuffing and bad mouthing. 0 - no ballot stuffing and bad mouthing; 1- ballot stuffing and bad mouthing.
- `const fec = 1` - Limit set for the fact table (0 - no limit and 1 - limit)
- `const fe = 2` - Maximum entry for a node allowed in fact table
- `const ne = 2` - Multiples of total nodes to find number of entries for each node in activity table (to be checked for the need).
- `const fsize = 1000` - Fact table size that is maximum entry allowed.
- `const vtype = 5` - Number of vote type (-1 - negative, 1 - positive, 0 - neutral and 2 or more are for incentive)
- `var Vvalue = [] int1,1,1,1` - Number of possible values for each positive vote in the trust models. If it is 2 that means possible values are 1 and 2
- `var Nvalue = [] int0,0,0,0` - Number of Negative values for each positive vote in the trust models. If it is 2 that means possible values are -2, -1 and 0
- `var maxvote = []int 1,1,0` - Maximum vote allowed for each user for the trusted model
- `var incentive = []int 1,1,0` - This is to refer the incentive allowed or not; 0 - no incentive, 1- incentive; for each function
- `var maxincentive = []int 101,101,0` - Maximum incentive allowed for each user trust function
- `var addincentive = []int 0,0,1` - To include the additional incentives. The number of values are based on the number of functions.
- `var addincentvalue = [][]int 0,0,0,0,0,0,0,0,1,2,1,2` - To include the incentive values for each activity.; 1 - voter, 2 - block, 3 - fact, 4 - positive vote. Index 0 will store the final score of the node.
- `const nsvarraysize = 7` - Number of possible values in Node Service Value (nsv) array
- `var nsv = []int 2,3,4,4,5,2,2` - Number of Node Service Value for each type; for example, if message service or id (mi) is traffic then nsv 0 refers no traffic; 1 refers traffic; 2 may refer too much traffic,..
- `var nodemain[n][n*ne][na]int` - Node Main three dimensional Array
- `var nodeservice[n][mi][nsa]int` - Node Service three dimensional array
- `var nodeservicetable[mi][sta]int` - Node Service table two dimensional array
- `var nodeneighbor[n][n]int` - Node Neighbour two dimensional Array
- `var fact1 [fsize][fa]int` - Fact table two dimensional Array
- `var tscore [n][tsa]float64` - Trust score two dimensional array
- `var tcompute [n][n*n][sca]float64` - Trust compute table three dimensional array to record the opinions or votes.
- `var presentsession = 0` - Refers the presentsession identity. It starts with 0.
- `var sybilvoteoption int` - This is to ensure that all sybil nodes will go for the same activity when go for voting.
- `var blockincent = 1` - whether blockmining required or not; this can be used for other incentive also. 0 - indicates no and 1 - indicates yes.
- `var blockincentive = [] int 0,0,1` - is there facility for the block mining incentive or not; this can be used for other types of incentives; 0 for incentive and 1 for no incentive.
- `var blockincenvalue = [] int 0,0,2` - is there facility for the block mining incentive or not that is

the value for the incentive. If block mining there but no incentive then we can put 0 for that function; this can be used for other types of incentives.

- `var factincentive = [] int 0,1,1` - this is to refer the possibility of fact table in the model. 0 for fact possible as well as incentive and 1 for no incentive.
- `var factincentvalue = [] int 0,1,0` - is there facility for the fact incentive or not. If value for factincentive then give that otherwise it can be zero
- `var ontheflynode = [] int 0,1,0` - this is to introduce the new nodes while running.
- `const maxflynode = 20` - this is to refer the number of on the fly nodes
- `var incentivevalue = []int 0,1,0` - incentive allowed or not; 0 - no incentive, 1- incentive; for each function.
- `const eachsessionfly = 1` - we need to introduce new maxfly nodes in each session. This should be equal to no of sessions or 1
- `var nodeactive[(n+1)+(maxflynode*(eachsession*nos))][2] int` - This is to maintain whether a particular node is active or not.
- `var sessionapplicable = []int 0,0,1` - This is to indicate the session for each trust model; 0 - no session, 1- session; for each function
- `var bloomvote [1000][nodesize] int` - This variable is to control the double voting on user report.
- `var factventry = []int 0,0,1` - This is to indicate whether the trust model go for claiming a negative message received, identified by itself and prove through global voting.
- `var Msecond = time.Duration(300)` - This is to refer how many mill seconds delay required so that all go routines will complete the job - this is in activity function
- `var Dsecond = time.Duration(10)` - This is to refer how many seconds delay required so that all go routines will complete the job - this is in Main function.
- `var Nentry = N * Ne` - Number of entries for each node in the nodemain table
- `const Collude = 1` - This is to enable the colluded attack. 0 - no collusion; 1- collusion.
- `const Colnode = 10` - Number of collusion nodes. This has to be less than the number of malicious node.
- `const Colludeentry = 100` - This refers the number of entries in the collude array table.
- `const Sybilentry = 100` - This refers the number of entries in the sybil array table.
- `var Colludearray[Colludeentry][10] int` - This is to support the collusion attack.
- `var Sybilarray[Sybilentry][10] int` - This is to support the Sybil attack.
- `var Colludeleader[3] int` - This is to store the collusion leaders and colluded nodes will follow the leaders.
- `var cattack = 0 int` - This is to tell the colluded nodes to go for attack. 0 no collision attack and 1 go for collision.
- `var collide = 0 int` - This is to refer now the leaders want to for collision attack. 0 no collision attack and 1 ready for collision.
- `var Sybilindex int` - This is for maintaining the track of entry in the Sybil array.
- `var Colludeindex int` - This is for maintaining the track of entry in the collude array.
- `var Colluderbloom[Colnode][2][Colludeentry] int` - This is to stop more than one time voting in colluder mode; two is given, zero for send message and one for fact.
- `var Sybilbloom[Colnode][2][Colludeentry] int` - This is to stop more than one time voting in sybil mode;two is given, zero for send message and one for fact.
- `var Whitewash = []int {0,1,0}`- This is to mention whether the trust model want the malicious user to go for the whitewash or change his identity.
- `var Whitethreshold = []int {0,0,0}` - This is the threshold value that indicates the node as attacker.
- `var Oscillation = []float64 0.0,0.0,0.4` - This holds the threshold to support the oscillation node to change its behavior after gaining the trust or reputation score.
- `const Osn = 1` - Number of oscillation node
- `var Totalnodes int` - This is to refer the total nodes at present in the network
- `var Introducecond = []int 0,0,0` - This is for if a trust model want to introduce a new node based on condition then set to 1 otherwise 0.
- `const Nonodegroup = 2` -This refers the number of node group like private and public.
- `var Resourceupdate = []int 1,1,1,1` - This is to update the service table entry after all set 1 trust model accept the message.