CS765 HW3
DApp for fact-checking a news article or item

Magham Dipen Anjan - 210050092
Patil Vipul Sudhir - 210050115
Hari Prakash Reddy - 210050119

April 2024

Part A: Design of DApp

In our implementation of DApp we used following methods or ideas:

- Registration requires deposits fees to be paid to avoid Sybil Attack.
- Trustworthiness for all initial users is same and is later updated based on trustworthiness update rule.
- For every news item there should be some service fees should be paid to avoid checking multiple news by changing slight content.
- Incentives are given to users who gave their vote in favour of majority which also motivates the rational user to give correct vote.
- There is an attack possible because of malicious users which we'll see in Part C.

Trustworthiness update rule:

Let T_j be trustworthiness of a voter in j^{th} round (i.e, j news items are checked previously). If a user joins at k^{th} round then-

$$T_k = 0.5$$

This says that every voter half trustwothiness initially. It will also solve problem of Bootstrapping. Later the trustworthiness should be updated based on their behaviour. Let *trustCount* be a variable for voter which measures the count of votes of user which was in favour with majority. and Initially it's zero. Let f be a factor used to update trustworthiness. f is given by-

$$f = \frac{1}{0.125 * j + 1}$$

Value of f lies in (0,1]. Trustworthiness is upadted using following rule for j^{th} round-

$$T_{j+1} = (1 - f) * (\frac{trustCount}{j+1}) + f * (0.5) \tag{1}$$

As j (number of rounds) increases the factor f decreases which implies that trustworthiness becomes more dependent on *trustCount* than fixed 0.5. In Solidity we cannot use floating point arithmetic so we modified the above equations by adding precision bits and removing them at last. Here we assumed trustworthiness to be an integer between [0,255]. Modified equations are as follows-

$$T_k = 128$$

$$f = \frac{8192}{(j/8) + 1}$$

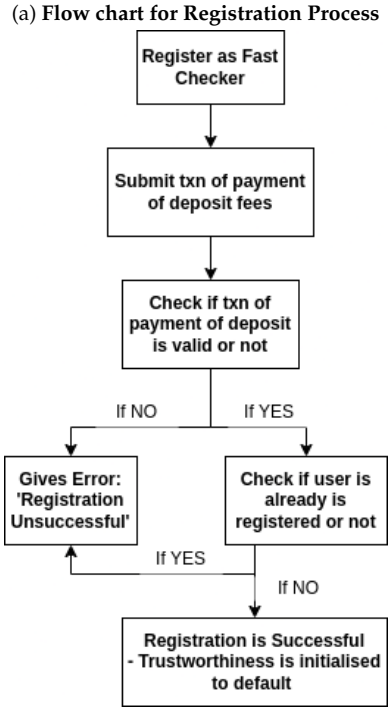
Value of f is integer from range (0,8192].

Trustworthiness is upadted using following rule for j^{th} round-

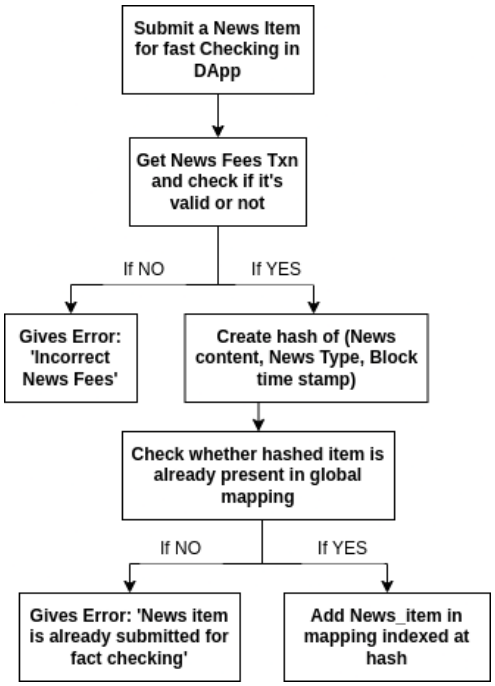
$$T_{j+1} = \left((8192 - f) * (255) * (\frac{trustCount}{j+1}) + f * (128) \right) / 8192$$

Flow Charts for Processes in DApp

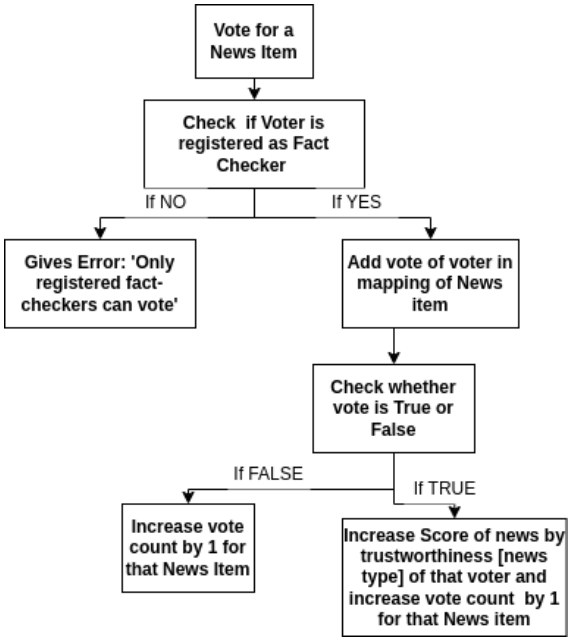
(a) **Flow chart for Registration Process**



(b) **Flow chart for submission of news**



(c) **Flow chart for Voting Process**



(d) **Flow chart for getting result of voting**

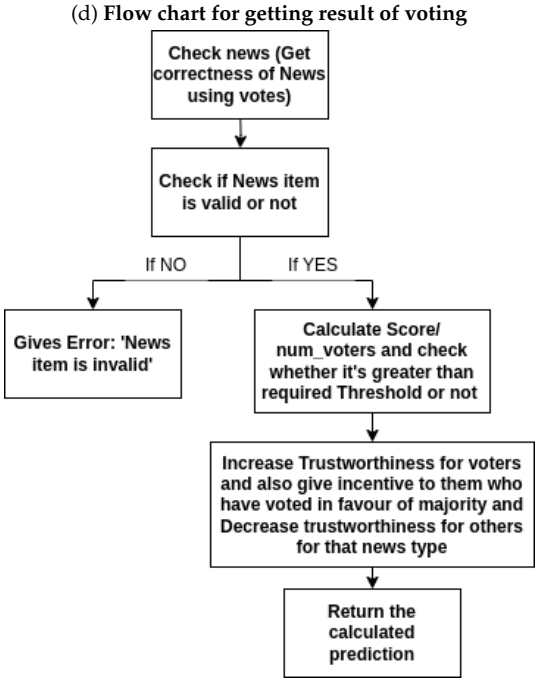


Figure 1: Flow charts for different processes in DApp implementation

Issues in DApp and their Solutions

Sybil attack:

A malicious person can create multiple identities and vote to skew the result in any direction
Solution: To avoid Sybil attack, while registration we can get some deposit from the user who wants to register as fact checker. As deposit gets locked, any rational user will create only one account. We can also use KYC (Know Your Customer) idea but it requires robust implementation as anonymity can be in danger. KYC verification ensures that there is unique account per person and deposit fees ensures that malicious person cannot create many accounts using different persons. But in our implementation we used only deposit fees idea.

Method to evaluate or re-evaluate the trustworthiness of voters:

The Dapp should evaluate how trustworthy different voters are based on how they vote. Note that someone might game the system to get a higher trustworthy rating. A method that is more robust to such gaming of the system, is preferable.
Solution: The voters who casts vote in favour of majority will see increased trustworthiness of theirs and voters who cast opposite in favour of majority will see decreased trustworthiness of theirs. The weight update rule for that is given in equation 1. The updates are done using trust count i.e, calculating fraction of times the peer's vote favoured the majority.

More trustworthy voters should be given more weight:

The opinions of more trustworthy voters should be given more weight. However, we must keep in mind that someone may be more trustworthy for certain types of news and not others. For example, someone may give excellent opinions about news related to Physics but is not so trustworthy on topics related to Politics or Economics.
Solution: The votes of the voters are considered using linear combination of their votes with weights as their trustworthiness. So more the trust worthiness of a voter more the contribution in the fact checking process.

Rational voters are to be incentivised:

Rational voters are to be given incentives to participate and vote truthfully to the best of their ability.
Solution: If a voter gives vote which favours the majority then we can give some money as incentive to that voter. The money which we give as incentives will be collected as Fees for Fact checking a News while submitting a news for Fact Checking.

Uploading a news item:

Some efficient method should be used to identify a news item (which is to be evaluated) in the Dapp.
Solution: We ca store each news in a global mapping indexed with it's hash. That hash can be used to send in messages while communication. When a user with hash wants to get News then he/she can get it from global mapping using hash as an index. This is efficient because there is no need to send news item every time while communicating, we can just send it's hash.

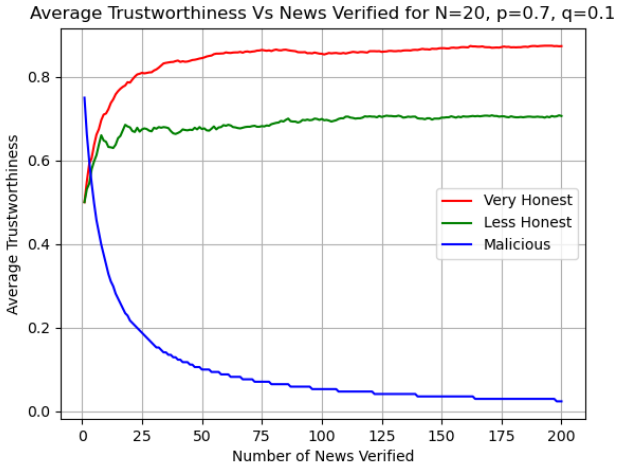
Bootstrapping:

If the Dapp does not have any trustworthy rating of different initial voters, then how to get started with fact-checking news?
Solution: Initially we can assume every voter has equal trustworthiness. Based on the behaviour trustworthiness is updated in every voting. Refer to equation 1. In our implementation trustworthiness is an integer between 0 to 255 (both included). We initialize the trustworthiness to 128 when user registers. Later based on his/her behaviour trustworthiness is updated.

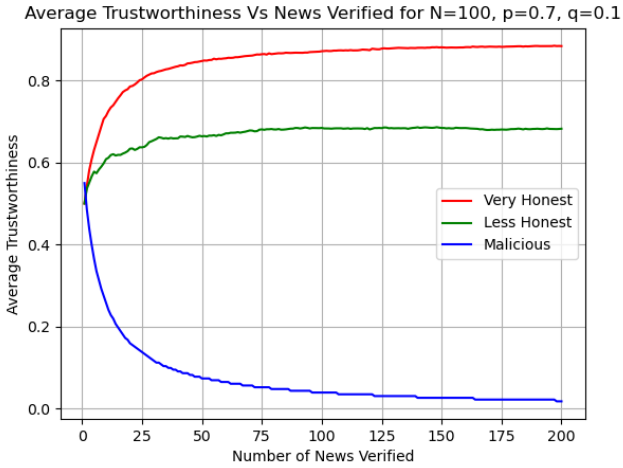
Part C: Results

Varying Number of voters

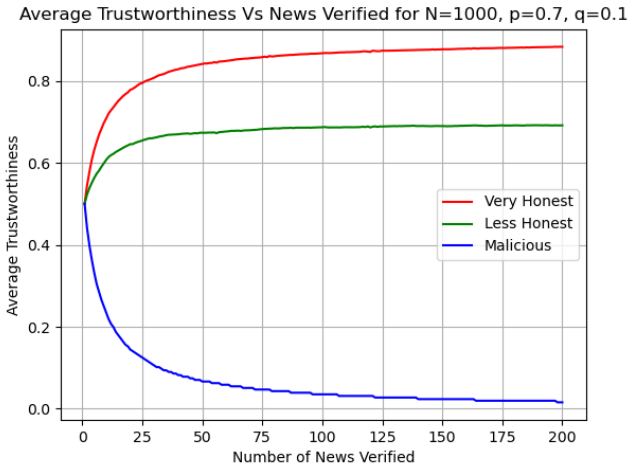
To analyse the behaviour of DApp algorithm on different values of number of voters(N) following plots are generated for N = 20, 100, 1000 by fixing values of p(fraction of less honest peers) = 0.7, q(fraction of malicious peers) = 0.1 and number of news items as 200.



(a) N = 20



(b) N = 100



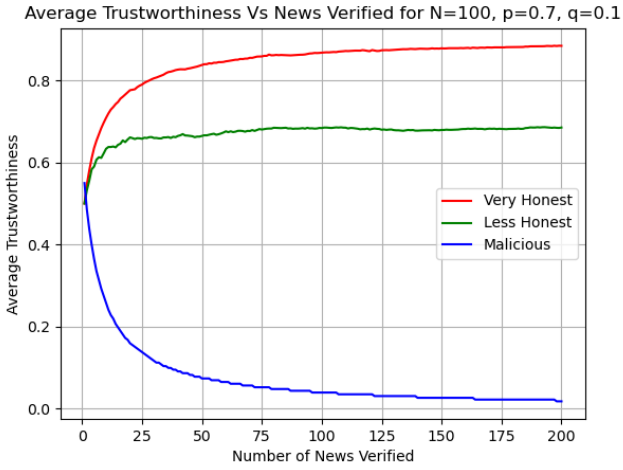
(c) N = 1000

Figure 2: Plots of average Trustworthiness Vs Number of News Items for N = 20, 100, 1000 for 200 news items

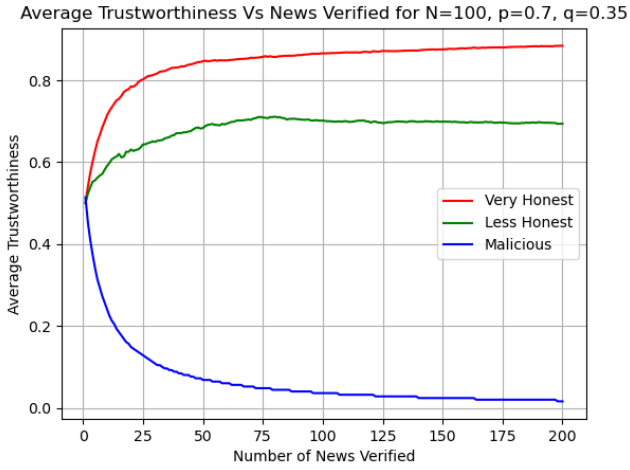
As value of N increases plots become smooth and deviation is less from ideal behaviour. The reason behind smoothness is because of average value which smoothens as N increases.

Varying value q(fraction of malicious peers)

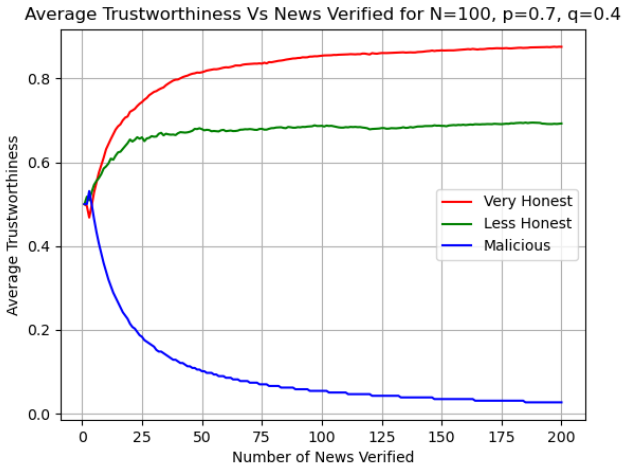
To analyse the behaviour of DApp algorithm on different values of fraction of malicious peers(q) following plots are generated for q= 0.1, 0.35, 0.4, 0.45, 0.5, 0.7 by fixing values of N(No. of voters) = 100, p(fraction of less honest peers) = 0.7 and number of news items as 200.
If fraction of malicious peers is greater than 0.5 then they get trustworthiness 1 eventually. For q values ¿ 0.4 also there are chances that malicious peers can get dominance. So here 51% attack is possible.



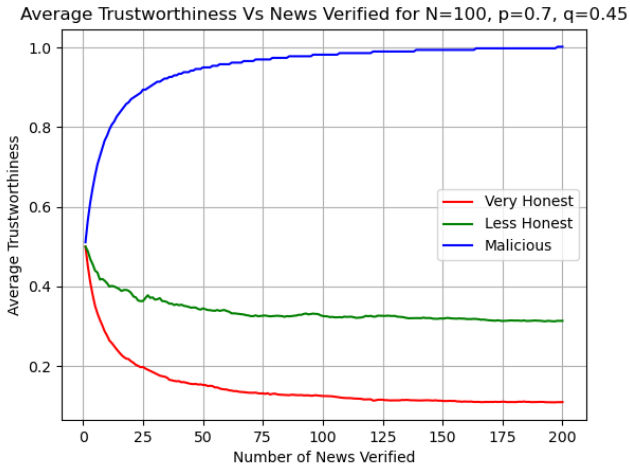
(a) q = 0.1



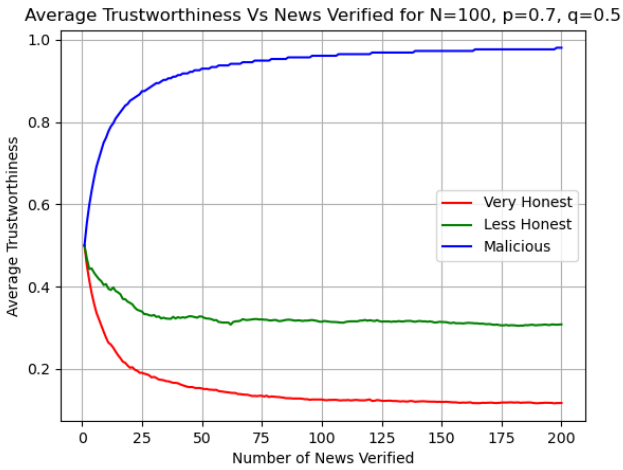
(b) q = 0.35



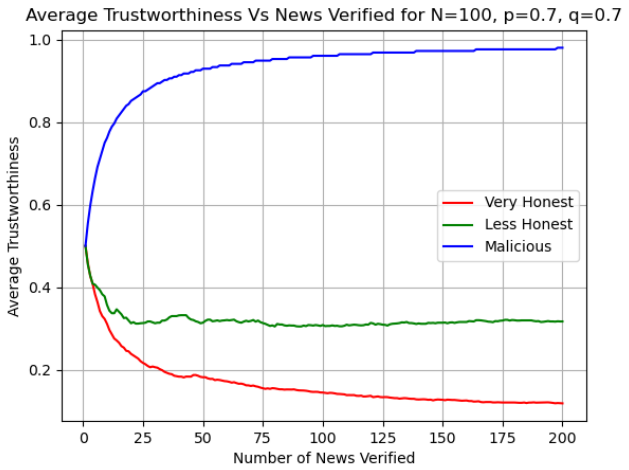
(c) q = 0.4



(d) q = 0.45



(e) q = 0.5

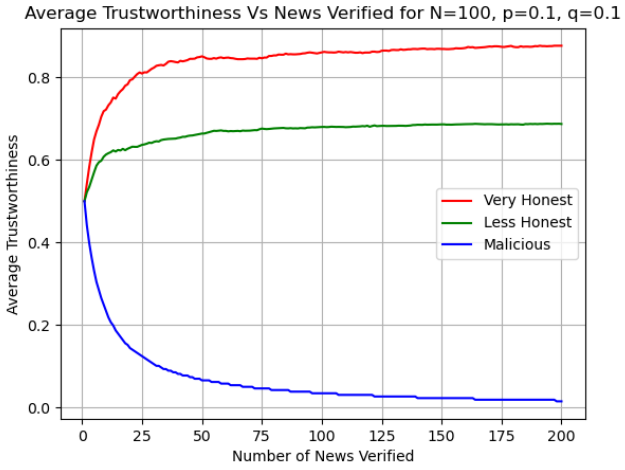


(f) q = 0.7

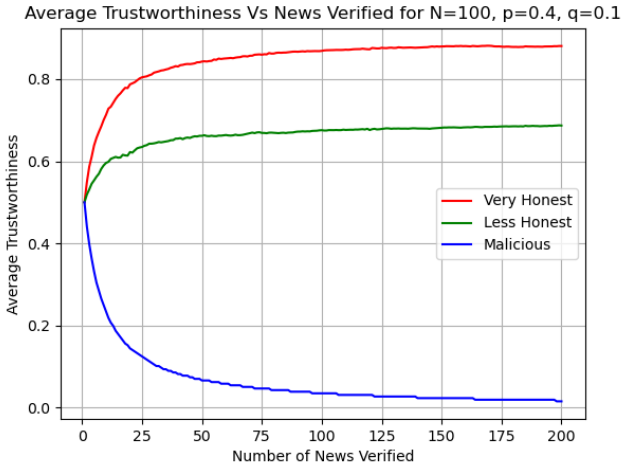
Figure 3: Plots of average Trustworthiness Vs No. of News Items for q= 0.1, 0.35, 0.4, 0.45, 0.5, 0.7 for 200 news items

Varying value p(fraction of less honest peers)

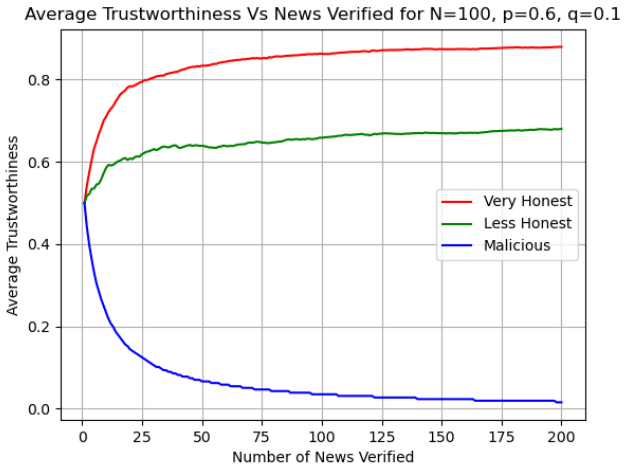
To analyse the behaviour of DApp algorithm on different values of fraction of less honest peers(p) following plots are generated for p= 0.1, 0.4, 0.6, 0.9 by fixing values of N(No. of voters) = 100, q(fraction of malicious peers) = 0.1 and number of news items as 200.



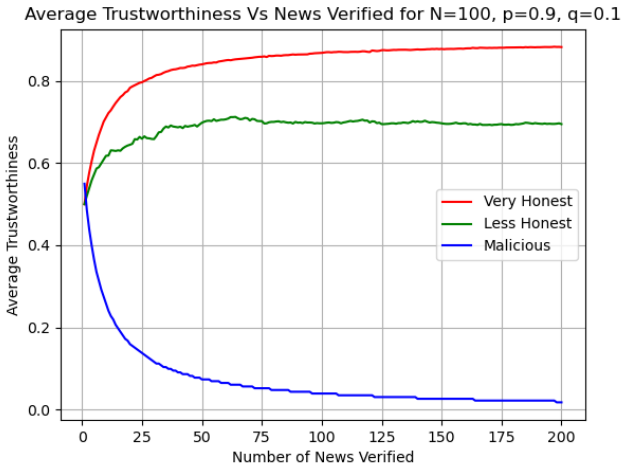
(a) p = 0.1



(b) p = 0.4



(c) p = 0.6



(d) p = 0.9

Figure 4: Plots of average Trustworthiness Vs No. of News Items for p= 0.1, 0.3, 0.5, 0.7, 0.9 for 200 news items