# An Analysis of Rumor Spreading Fundamentals with a Case Study on Facebook

Burcu SAYİN[1] and Serap ŞAHİN[1]

[1]İzmir Institute of Technology, İzmir/Turkey, burcusayin@iyte.edu.tr
[1]İzmir Institute of Technology, İzmir/Turkey, serapsahin@iyte.edu.tr

***Abstract* - Nowadays, the effect of social networks on people's lives is quite high. This situation gives rise to the density of information exist over social networks. That is why, analyzing the spreading pattern of information on social networks is an important issue today. The aim of this study is to technically review the background of information spreading, especially the fundamentals of rumor spreading and analyze the well-known methods on SNs. As a result, this article provides an important background for those, who works on the information spreading over SNs.**

***Keywords* – rumor spreading, epidemics, social networks**

## I. INTRODUCTION

TODAY'S most popular virtual environment is Social Networks (SNs), with a 2.7 billion user [1]. SNs are mostly represented by graphs. Graphs are generally used to represent relationship among SN users. This relationship dynamically changes by (i) adding or removing new friendships and (ii) changing the profile settings. Some questions on this graph representation are listed below:

- How does a post or personal information spread on SN graph?
- Which model does better explain the information spreading?
- How the speed of information spreading is defined and measured?
- Which methods are more efficient to spread any information on SNs?

When answering these questions, an analogy is used between rumor and information spreading because information spreads like a rumor from one friend to another in SNs. That is why, the objective of this study is to comprehend the fundamentals of rumor spreading, including its mathematical background and state-of-art methods. Furthermore, we analyzed the efficiency of these methods in SNs. As a case study, we selected Facebook as our research domain.

The rest of this paper is organized as follows. In Section 2, we present the fundamentals of rumor spreading. Section 3 covers our experimental work on Facebook. We conclude the paper in Section 4.

## II. FUNDAMENTALS OF RUMOR SPREADING

### *A. Epidemic Models*

Previously, rumors and epidemics were likened to each other. Epidemics spread for a time and then lose its effect; rumor also has the same behavior but works with a threshold theorem. The theorem models the population by three types; susceptible ($s$), infected ($i$), and removed ($r$) which is called as SIR model. "Susceptible" ones are ignorant. After a susceptible one gets a rumor, it becomes "Infected". "Removed" means it stopped spreading.

In epidemics, time evolution of a disease according to this theorem is governed by a threshold that is defined based on the number of susceptible, infected and removed rates respectively. For example; in one hand, the probability of infecting the disease in an area that consists of a small number of people is low. On the other hand, if there are many people in the environment, this probability becomes bigger. Thus, population size is an important effect in epidemics [2].

Mainly, epidemics exist with two models: simple epidemics and complex epidemics [3]. Simple epidemics infect the whole population proportionally to the log of the population size. Let the population size be n, then epidemics spread with $\log n$. Additionally, individuals are always either susceptible or infected in this model. On the other hand, in complex epidemics, individuals are either susceptible, infected, or removed. The only complication is to decide when to be removed. In this instance, rumor spreading is deterministically modelled according to equations (1) and (2), given that $s + r + i = 1$.

$$\frac{ds}{dt} = -si \tag{1}$$

$$\frac{di}{dt} = +si - \frac{1}{ctr}(1-s)i \tag{2}$$

The equation (1) shows that susceptible ones will be infected according to the product of number of susceptible and infected populations denoted as "$si$". The equation (2) shows an interest lost for any infected one in time, which means it has recovered from disease or the individual lost interest to spread this rumor. To simulate these equations, an additional term counter value ($ctr$) is added, where $\frac{1}{ctr}$ shows the probability of interest loss. In equation (3), the infection function $i(s)$ denotes $i$ as a function of $s$ with the elimination of time and defined the number of infected population:

$$i(s) = \frac{ctr+1}{ctr}(1-s) + \frac{1}{ctr}\log s \tag{3}$$

Here the value of $i(s)$ goes to zero when the value of $s$ decreases exponentially with the value of $ctr$ according to $s = e^{-(ctr+1)(1-s)}$ [3,4]. This means that if $ctr$ is increased, rumor reaches a bigger portion of network, but it requires more rounds to succeed this. Hence, $ctr$ provides control over the termination and the size of rumor spreading area in a network.
Furthermore, complex epidemics models are divided into two sub-models: complex epidemics with static rumor and complex epidemics with dynamic rumor. During the whole spreading process, if rumor does not have any revision, then it belongs to static rumor sub-model. Otherwise, it becomes a dynamic rumor which is revised by nodes during the process [5].
Overall, rumor spreading algorithms can focus on one of these model structures. Because of the relation between rumors and epidemics, rumor spreading algorithms are also established on these two models. In the following section, the implemented model for our experiments on Facebook SNAP dataset will be explained.

## B. *Rumor Spreading Models*

Rumor spreading can be utilized in a network context to keep all the nodes up-to-date. Here, each node represents an individual. Demers et al. [3] define three common methods for performing this propagation update. These methods are:

- The *direct mail method*, where each update is mailed to all other nodes. Although it is efficient, it is not considered reliable because mails can be lost.
- The *anti-entropy method* is a simple epidemic technique where each node randomly chooses another adjacent node to resolve their differences and, then, exchange a rumor. All nodes perform this action at each round and therefore the communication cost increases. This method is less efficient from the direct mail due to the extra processing cost of difference resolving. Investigations on the cost of anti-entropy gave rise to three update distribution mechanisms: *push*, *pull*, and *push-pull* methods.
    - o According to *push method*, in each round one node randomly chooses another one, if the node which made the selection has more updated rumor than the selected node, it pushes the update.
    - o *Pull method* does the reverse; having the more updated rumor, the selected node pulls the update.
    - o As for *push-pull method*, both push and pull methods are applied in each round.
- The rumor mongering method for update propagation adopts complex epidemics mechanisms to distinguish it from anti-entropy techniques. This method also uses the same update distribution mechanisms (push, pull and push-pull). The scheme considers all nodes as susceptible at the beginning. When a node takes a rumor, it becomes infected and only these nodes start to spread it by doing random choices till it loses interest in spreading the rumor. Due that reason, the communication cost decreases. The main issues of rumor mongering are:
    - i. Decide when to stop spreading (the number of susceptible nodes should be close to 0, which is measured by the count of uninformed or residue nodes ($rsd$))
    - ii. The spreading speed of rumor to all population should be minimized. The system should converge to an inactive state (a state that there is no infection, which means spreading is terminated) by the least number of rounds which is defined by $S_n$. $S_n$ is also related to the speed of rumour spreading.
    - iii. If $ctr$ value is increased (as mentioned in equation 3), $S_n$ increases. In this case, rsd value decreases because number of uninformed nodes becomes smaller when the number of rounds increase.

According to the study of Demers et al. [3]; randomized algorithms are used to propagate updates for database maintenance in a distributed environment. The algorithms in this study propose some methods which are listed below. They manage how the system converges according to different ways of interest loss in spreading process.

- *feedback:* a sender loses interest only if the recipient already knows the rumor, this feedback depends on the probability of neighbor node's state,
- *blind:* sender loses interest with probability $\frac{1}{ctr}$ regardless of the recipient state,
- *counter:*
    - o counter with feedback: sender loses interest only after $ctr$ unnecessary contacts,
    - o counter with blind: sender loses interest after any $ctr$ contacts.

In the proposed rumor spreading model:

i. The main performance variables are $rsd$ (the number of residue uninformed nodes) and $S_n$ (the number of rounds to converge whole network); $rsd$ should be close to 0 and $S_n$ should be optimized.
ii. The input variables are defined by $ctr$ (counter) as mentioned above and the communication traffic m. Communication traffic is measured by the number of update messages sent in each round without regarding to the topology of the network.
    a. The deterministic solutions prove that increasing $ctr$ value with feedback is an

effective way of minimizing the values of $rsd$ and $S_n$.

b. The average value of traffic for an infected node in each round is formulated as $m = \frac{Total\ rumor\ update\ count}{number\ of\ node}$

iii. $S_n$ is proportional to the value of m. Increasing the value of m, on the other hand, decreases the residue according to $rsd = e^{-m}$.

A relation among the performance variables ($ctr$, $rsd$, m and $S_n$) can, also, be seen in the original article [3].

Anti-entropy and rumor mongering methods both use the same push, pull and push-pull algorithms as mentioned above. But depending on the nature of the network, the advantages, and disadvantages between pull and push algorithms vary. For instance; if a network has very frequent multiple updates simultaneously, then the pull algorithm has advantages in spreading the rumor very fast because there is a high probability that the selected node is already infected. But if a network has very rare updates, then the pull algorithm creates an unnecessary traffic and in that case push algorithm is preferable. Karp et al. [6] compared the push and pull algorithms under the same assumptions such as similar update rate, under uniform distribution and a perfect interconnection without failures. The result of their performance is presented in Figure 1.a and 1.b.

- Push algorithm forwards the rumor to the nodes and the set of infected nodes grows exponentially until reaching to $n/2$ of the population. After this point as shown in Figure 1.a, the set of susceptible populations shrinks with a constant factor at each round. This factor is about $(1 - 1/e)$ since the fraction of players that do not receive a call in a round is approximately $1/e$. Thus, this shrinking phase takes $\theta(lnn)$ rounds and the push algorithm sends $\theta(n)$ messages.

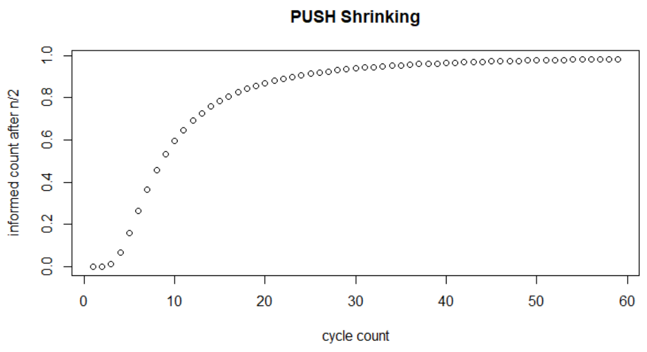

Figure 1.a. PUSH Algorithm Shrinking Phase

- In the implementation of pull algorithm; the infected node must wait for a connection request to start spreading the rumor. Therefore, propagation time can be unpredictable for the first round. After the set of infective node count reaches to $n/2$ of the population as shown in Figure 1.b, pull algorithm has an advantage against the push algorithm due to the fraction of susceptible nodes roughly squares from round to round. This is because in a round starting with $\epsilon$.n susceptible nodes, each node has probability $1 - \epsilon$ to receive the rumor, so that the probability of staying susceptible is $\epsilon$ and $\epsilon > 0$. At the end of the round $\epsilon.n(1 - \epsilon) \approx \epsilon^2.n$ susceptible nodes will exist. Thus, shrinking phase takes $\theta(\ln \ln n)$ rounds and $\theta(n \ln \ln n)$ messages. "n" factor in the message count comes from the total node number because according to the assumption behind these methods is that each node transmits a message in each round.

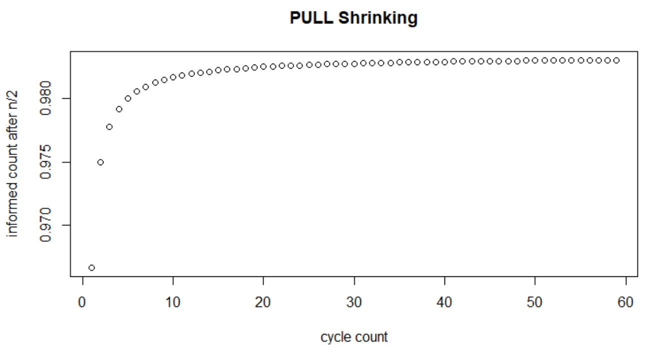

Figure 1.b. PULL Algorithm Shrinking Phase

The goal of R. Karp et al. [6] is to spread the rumor among all nodes with minimized number of rounds and update transmission. They impress that aforementioned algorithms are commonly used for lazy transmission of updates. They investigate that this large communication overhead is coming from the nature of epidemic algorithms and it can be reduced significantly when the rumor is sent in both directions. Combining the benefits of push and pull algorithm provides a more efficient algorithm. Push algorithm works efficiently until $n/2$ of the population becomes informed and pull algorithms works efficiently only after $n/2$ becomes informed. Thus, by using those two algorithms at the same time, we can get a more efficient result. With a simple push-pull algorithm, spreading a rumor to all nodes requires $S_n = O(\ln n)$ rounds and $O(n \ln \ln n)$ transmissions. For instance; the random phone call model uses uniform distribution model and push-pull algorithm [6]. This randomized communication of n players in parallel rounds increase robustness of the rumor spreading model. At each round, a node u randomly picks another node v. Then, u makes a phone call to v and they exchange rumors. At this model, they state that in a round t, rumor can be exchanged in both directions (push-pull). Thus, they claim that the number of transmissions can be reduced significantly by using a simple push-pull algorithm. This algorithm informs all nodes in time as the maximum number of rounds $S_{n_{max}} = log_3 n + O(\ln \ln n)$ using $O(n \ln \ln n)$ messages.
A problem of this algorithm is determining the optimal termination time. Furthermore, it is very sensitive to any error among nodes which affects the rumor spreading. In order to improve their solution, they devise a distribution termination scheme which is called as median-counter algorithm [6]. According to this algorithm, rumor is defined as r and there are four types of states for each node; A, B, C and D. Nodes that has not taken any rumor yet are in state A. For nodes, which are

in state B a counter is held, which is shown as $ctr$ (v,r) where v denotes the node. Each time a node in state B takes a new rumor r, it increases its counter. If $ctr(v,r) = x$, then node v is in B-x state. If x is equal to the maximum counter value, state is changed to C. If a node v in state A receives r only from nodes in state B, then it switches to state B-1. If a node in state A receives r from a node in state C, then it switches to state C. For nodes in state C; every node stays in this phase for at most $ctr_{max} = O(\ln\ln n)$ rounds, and then switches to state D which terminates the rumor spreading. Hence, the median-counter algorithm uses only $S_n = O(\ln n)$ rounds with $O(\ln\ln n)$ message transmission. How the rsd value is affected and why median-counter algorithm is used are explained in the article thoroughly.

Furthermore, Karp et al. [6] give a general lower bound for the random phone call model using uniform distribution model that any address-oblivious algorithm requires $S_n = \Omega(\ln n)$ rounds in order to inform all nodes and requires $\Omega(n \ln\ln n)$ transmission independent from the number of rounds.

## III. Experiments

We performed our experiments on SNAP Facebook dataset [7]. This dataset contains real data from Facebook such as users, profile features, friendship relations etc. However, all these data are kept as anonymized to protect user's privacy. Our dataset contains 4039 real Facebook users and 88234 relations between them. We used NetworkX library [8] and created a graph from this dataset.

Before describing our experiments, we first provide some information about Facebook. Facebook is a social network which allow users to add other users as friend (friendship relation), follow the friends' updates, share photo/status (post), like/comment on shared posts, join specific groups etc. Moreover, when a user creates a post, s/he can select a privacy setting for it. There are some specific terms like Friends, Friends of Friends, Public etc. These terms define the visibility area of the posts on social graph. For example, if someone selects "Friends" as privacy setting for his/her post, then only his/her friends can see or access it. Thus, we can observe some subsets i.e. friendship graph over the whole social graph based on specific users. Friendship graph of a user only contains nodes and the corresponding edges which are directly connected with this user via friendship relation.

In Facebook, user's privacy preferences for the posts on their profile directly affect the visibility of those posts in SNs. The user has, also, the rights to put some additional restrictions on the visibility of specific posts, such as direct addressing to a specific user or group to view this post. Apart from the case of restricting the privacy setting to only a specific person or group, the post can be visible by all users in the friendship graph. However, the visibility of the post can be beyond of this friendship graph if the members of it also share this post with or without comments on their friendship graphs. According to these specifications of Facebook we assume that:

i. Each post can be generated randomly by any user and this post spreads on whole Facebook graph through connected users.
ii. Spreading process stops when all connected users in network receive the post and there is no requirement for a counter ($ctr$) value to define removal times.

In our experiments, we would like to observe theoretical outcomes by the measurement of $S_n$ value for the spreading of a specific rumor on whole network. Hence, we used "complex epidemic model with static rumor, without a counter value" as presented in previous section. The following sections present the observed, analyzed and evaluated experiments.

We refer to Karp et. al. [6] and perform an experiment to observe push, pull and push-pull methods on Snap Facebook dataset.

We propose a detailed explanation of our experiment together with the algorithms in the following part:

**Push method:** Algorithm 1 shows the steps of this method. We kept and continuously monitored two lists. One for susceptible nodes and other one is for infected (informed) nodes. Initially, susceptible list contains all nodes in the graph and infected list is empty. First round starts after we randomly select a node from susceptible list and give the rumor to that node. The selected node becomes informed, so it is removed from susceptible list and added to our infected node list. One round is completed when all nodes in infected list randomly select a node among its neighbor nodes and pushes the rumor. Each new informed node is removed from susceptible list and added to infected list so list sizes changes dynamically. Push method runs until the whole nodes in the network become informed.

***Algorithm 1***

```
Input: val [int] – any information/rumor, network – social graph
1. SET infectedNodeList to an empty list
2. SET nodeList to all nodes in network
3. SET randomNodeID to a randomly selected node
4. Find the node whose ID is equal to randomNodeID and
   inform it about val
5. SET numberofRounds to 0
6. SET infectedNodeCount to 1
7. WHILE all nodes in network are not informed:
       a. FOR all nodes in infectedNodeList:
              i.  Select a neighborNode
              ii. If the neighborNode does not have the
                  rumor, inform it and add neighborNode
                  to infectedNodeList. Increment
                  InfectedNodeCount by 1.
       b. Increment numberOfRounds by 1
Output: numberOfRounds
```

**Pull method:** Algorithm 2 explains this method. We have susceptible and infected lists again but this time susceptible nodes randomly selects a neighbor and try to pull the rumor. All nodes in the network are susceptible at the beginning so susceptible list holds the whole network initially and the infected list is empty. After we randomly select a node and give the rumor, that node becomes informed and we remove it from susceptible list so add to infected list. Then, first round starts.

One round is completed when all nodes in susceptible list randomly selects a neighbor node in network and pulls the rumor. Each new informed node is removed from the susceptible list. Pull method runs until the whole nodes in the network become informed.

***Algorithm 2***

```
Input: val [int] – any information/rumor, network – social graph
1. SET susceptibleNodeList to all nodes in network
2. SET nodeList to all nodes in network
3. SET randomNodeID to a randomly selected node
4. Find the node whose ID is equal to randomNodeID,
   inform it about val and remove it from
   susceptibleNodeList
5. SET numberofRounds to 0
6. SET infectedNodeCount to 1
7. WHILE all nodes in network are not informed:
     a. SET currentInfectedNodes to empty list
     b. FOR all nodes in susceptibleNodeList:
          i. Select a neighborNode
          ii. If the neighborNode have the val, pull it
              and add this node to infectedNodeList.
              Increment InfectedNodeCount by 1.
     c. FOR all nodes in currentInfectedNodes:
          i. Remove from susceptibleList
     d. Increment numberOfRounds by 1
Output: numberOfRounds
```

**Push-pull method:** Algorithm 3 depicts the steps of this method. Whole network is kept in a list. First round starts after we randomly select a node and give the rumor to that node. One round is completed when all nodes in the list randomly choose a neighbor node and either pushes/pulls the rumor or does not do any operation. If the current node does not have the rumor but randomly selected neighbor node has the rumor, current node pulls the rumor from it. Otherwise, it pushes the rumor to randomly selected neighbor node. If both the current and randomly selected nodes have the rumor or neither of them have it, no operation is needed. Push-pull method runs until the whole nodes in the network become informed.

***Algorithm 3***

```
Input: val [int] – any information/rumor, network – social graph
1. SET nodeList to all nodes in network
2. SET randomNodeID to a randomly selected node
3. Find the node whose ID is equal to randomNodeID and
   inform it about val
4. SET numberofRounds to 0
5. SET infectedNodeCount to 1
6. WHILE all nodes in network are not informed:
     a. FOR all nodes in network:
          i. Select a neighborNode
          ii. IF the node has the val but neighborNode
              does not have; inform it and then
              increment InfectedNodeCount by 1.
          iii. ELSE IF the node does not have the val
              but neighborNode has; pull the rumor
              and then increment InfectedNodeCount
              by 1.
     b. Increment numberOfRounds by 1
Output: numberOfRounds
```

Our results support the study of Karp. et. al. [6]. Let the population size be n. After $n/2$ of the population being informed, pull method spreads faster than push method. Below two figures shows the differences.

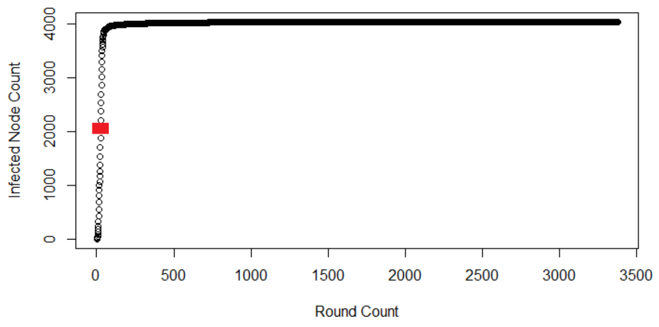

Figure 3.a PUSH Algorithm Shrinking Phase
Population Size: 4039

For the push algorithm shrinking phase, 2042 nodes become informed after 26 rounds (red point in Figure 3.a).After that, 3354 more rounds are performed to inform whole population (4039 nodes). Figure 3.a shows this situation, after $n/2$ of the population becomes informed, it takes too much round to inform whole population because the probability that a randomly chosen neighbor node being uninformed (susceptible) is too low.

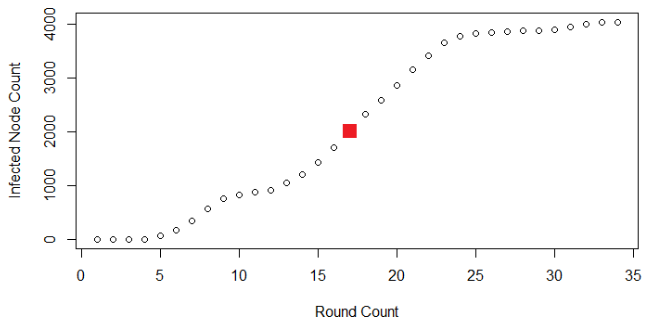

Figure 3.b PULL Algorithm Shrinking Phase
Population Size: 4039

For the pull algorithm shrinking phase, 2017 nodes become informed after 17 rounds (red point in Figure 3.b).After that, 17 more rounds are performed to inform whole population. Thus, pull algorithm informs whole population in only 34 rounds. It can also be seen in Figure 3.b that after $n/2$ of the population becomes informed, whole population becomes informed in a fast way because the probability that a randomly chosen neighbor node being uninformed (susceptible) is very high.

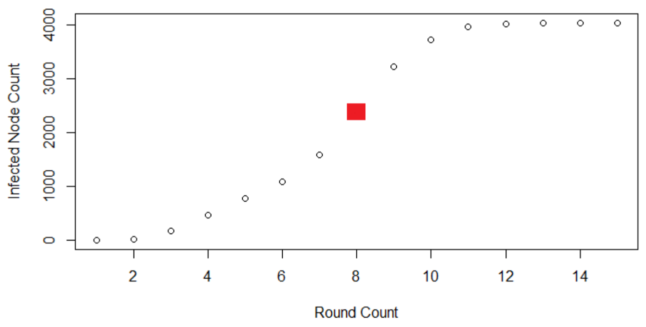

Figure 3.c PUSH-PULL Algorithm Shrinking Phase
Population Size: 4039

For the pull algorithm shrinking phase, 2396 nodes become informed after 8 rounds (red point in Figure 3.b).Then, only 7 more rounds are performed to inform whole population. It can be seen in Figure 3.c that push-pull algorithm over performs both push and pull algorithms and spreads the rumor through all nodes in a fast way. It takes 15 rounds to inform 4039 nodes. This experiment supports the claim of the study [9] which says that "if an n-node connected graph has conductance $\phi$ then rumor spreading successfully broadcasts a message within $S_n = O\left(\frac{log^4 n}{\phi(G)^6}\right)$ steps, with high probability, using the push-pull strategy."

## IV. CURRENT APPROACHES TO RUMOR SPREADING

In this section, we will firstly introduce the studies, which focus on the modified version of SIR model as an adaptation to information spreading, and then demonstrate an information spreading model based on cascades, that allow us to predict how well the information will be spread.

Bao et al. [10] revises SIR model and proposes SPNR model by dividing the Infected state into two: (i) Positive Infected (P: nodes that have been infected, and they support the information) and (ii) Negative Infected (N: nodes that have been infected, but they oppose the information).

Serrano et al. [11] propose an agent-based information spreading model, considering four states: (i) Neutral (initial state), (ii) Infected (believe the information), (iii) Vaccinated (believe the anti-information before being infected) and (iv) Cured (believe the anti-information after being infected).

Cordasco et al. [12] propose that a user may not immediately start spreading when he/she becomes an Infected; so they only become an "Aware". This model consists only these three states: (i) Ignorant, (ii) Aware and (iii) Spreading. Instead of defining a Removed state, they propose a termination rule in the original paper.

Sumith et. al. [13] claims that the assumptions made in SIR model fail in real world. They propose that the whole population do not mix with each other equally, so the distribution is not homogeneous. Moreover, all individuals are not Susceptible initially, and most of them will restrain themselves from interactions. With this approach, authors proposed $R_n$ SIR model, as an extension to SIR model. Hence, they added a new state to SIR model, which is called as $R_n$. $R_n$ state represents nodes, who restrain themselves from any interaction with other nodes.

Tong et al. [14] describes an information cascade model in SNs. First, they provide an extensive study on cascade scales, the scope of the cascade subgraphs, and topological attribute of spread tree. Then, based on the evaluation results, they analyze the spread of the user's decisions for city-wide activities. Decisions include "want to take part in the activity" and "be interested in the activity". This study introduces three mechanisms to use for making a decision: (i) equal probability, (ii) similarity of nodes, and (iii) popularity of nodes.

Overall, the main aspect of the current approaches for modeling the information spreading is to propose a realistic model that matches with the complex and dynamic mechanism of human behavior. Hence, researchers try to adopt their models with new parameters, such as popularity of nodes, similarity between them, etc.

## V. CONCLUSION

The aim of this study is to propose an enhanced technical review on rumor spreading models and analyze the success of core update distribution mechanisms (push, pull, push-pull) on a popular social network, Facebook.

In this paper, we highlighted how an epidemic model and rumor spreading model resembles each other. Then, we analyzed related mathematical models, rumor spreading algorithms and their success on Facebook data. Results show that complex epidemic model can be adapted to social networks and push-pull algorithm is a realistic approach to observe the actual spreading of information.

Many other implementation and research domains exist. Most of them requires efficient data aggregation algorithms and modelling for dynamic SNs [15,16]. Another application domain could be source detection of any information exist on SNs, which is a valuable evidence to verify any received information and to protect SN user from malware attacks.

## REFERENCES

[1] https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/ "Number of social media users worldwide from 2010 to 2020 (in billions)"

[2] D. J. Daley and D. G. Kendall (1965), Stochastic Rumours, IMA J Appl Math (1965) 1 (1): 42-55. doi: 10.1093/imamat/1.1.42

[3] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In Proceedings of the sixth annual ACM Symposium on Principles of distributed computing (PODC '87), Fred B. Schneider (Ed.). ACM, New York, NY, USA, 1-12. 1987. DOI=10.1145/41840.41841 http://doi.acm.org/10.1145/41840.41841.

[4] N. T. J. Bailey. The Mathematical Theory of Infectious Diseases and its Applications. Hafner Press, Second Edition, 1975.

[5] Y. Zhang, S. Zhou, Z. Zhang, J. Guan and S. Zhou. Rumor Evolution in Social Networks. Physical Review E, vol. 87, no. 3, Article ID 032133, 2013.

[6] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS '00). IEEE Computer Society, Washington, DC, USA, 565-574, 2000. DOI: 10.1109/SFCS.2000.892324.

[7] https://snap.stanford.edu/data/egonets-Facebook.html

[8] https://networkx.github.io/

[9] F. Chierichetti, S. Lattanzi, and A. Panconesi. Rumour spreading and graph conductance. In Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms (SODA '10). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1657-1663.9. 2010.

[10] Y. Bao, C. Yi, Y. Xue, and Y. Dong. A new rumor propagation model and control strategy on social networks. In Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '13). ACM, New York, NY, USA, 1472-1473, 2013. DOI=http://dx.doi.org/10.1145/2492517.2492599.

[11] E. Serrano, C. Ángel Iglesias, and M. Garijo. A Novel Agent-Based Rumor Spreading Model in Twitter. In Proceedings of the 24th International Conference on World Wide Web (WWW '15 Companion). ACM, New York, NY, USA, 811-814, 2015. DOI: http://dx.doi.org/10.1145/2740908.2742466.

[12] G. Cordasco, L. Gargano, A. A. Rescigno, and U. Vaccaro. Brief Announcement: Active Information Spread in Networks. In Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing (PODC '16). ACM, New York, NY, USA, 435-437, 2016. DOI: http://dx.doi.org/10.1145/2933057.2933069.

[13] N. Sumith, B. Annappa, and S. Bhattacharya. Rnsir: A new model of information spread in online social networks. In 2016 IEEE Region 10 Conference (TENCON), pages 2224–2227, Nov 2016.

[14] C. Tong, W. He, J. Niu, and Z. Xie. A novel information cascade model in online social networks. Physica A: Statistical Mechanics and its Applications, 444(Supplement C):297 – 310, 2016.

[15] D. Kempe, A. Dobra, and J. Gehrke. 2003. Gossip-Based Computation of Aggregate Information. In Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS '03). IEEE Computer Society, Washington, DC, USA, 482-491, 2003.

[16] C. Liu, Z.-K. Zhang, Information spreading on dynamic social networks, Communications in Nonlinear Science and Numerical Simulation, Volume 19, Issue 4, April 2014, Pages 896-904, ISSN 1007-5704, http://dx.doi.org/10.1016/j.cnsns.2013.08.028.