REPUBLIC OF TURKEY

AKDENIZ UNIVERSITY

IMPROVING GOSSIP-BASED DECENTRALIZED DEEP LEARNING

Yunus BÜTÜN

INSTITUTE OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

JUNE 2022

ANTALYA

REPUBLIC OF TURKEY

AKDENIZ UNIVERSITY

IMPROVING GOSSIP-BASED DECENTRALIZED DEEP LEARNING

Yunus BÜTÜN

INSTITUTE OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

JUNE 2022

ANTALYA

REPUBLIC OF TURKEY
AKDENIZ UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

IMPROVING GOSSIP-BASED DECENTRALIZED DEEP LEARNING

Yunus BÜTÜN

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

This thesis was unanimously accepted by the jury on 27/06/2022

Asst. Prof. Dr. Mustafa Berkay YILMAZ (Supervisor)

Asst. Prof. Dr. Murat AK

Prof. Dr. Cafer ÇALIŞKAN

ÖZET

FISILTI TABANLI DAĞITIK DERİN ÖĞRENME YÖNTEMİNİN GELİŞTİRİLMESİ

Yunus BÜTÜN

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Mustafa Berkay YILMAZ

Haziran 2022; 50 sayfa

Bu çalışmada fısıltı protokolleriyle yerel olarak eğitilen modellerin düğümler arasında rastgele değişimine dayananan tam dağıtık öğrenmeyi araştırdık. Diğer parçalı dağıtık öğrenme yöntemlerinden farklı olarak bu metotta bir lider yada koordinator düğüm bulunmuyor. Her düğümde aynı algoritma çalışır. Her düğüm aldığı modellerin ortalamasını alır ve kendi yerel verisiyle eğitir. Teorik olarak ispatlanmasa da pratikte kullanılışlı durumdadır. Bu algoritmayı anlamak için algoritmanın modelleri dağıtmak için kullandığı iletişim protokollerinin anlaşılması gerekiyor. Fısıltı öğrenmesi ilk olarak itme tipi fısıltı protokolü temelli olarak öne sürüldü. Bu alanda çok az çalışmanın olduğunu gözlemledik. Dağıtık öğrenmenin gelişmesi üzerine bina edildiği fısıltı protokollerinin doğru şekilde uygulanması ve model birleştirme stratejilerine dayanıyor. Bu çalışmanın ilk bölümünde var olan bütün fısıltı protokolleri SI ve SIR modunda teorik ve deneysel olarak ayrıntılı şekilde incelendi. İtme protokolünün SIR modunda enfekte olan düğüm sayısının en yoğun olduğu zamanı tahmini gibi birçok denklem türetildi. Var olan çalışmaların çoğunda fısıltı öğrenmesi SVM ve lojistik regresyon gibi doğrusal öğrenme algoritmalarıyla yapılıyor. Bu çalışmanın ikinci bölümünde yapay sinir ağları kullanarak dağıtık veri üzerinde öğrenme işlemini değişik parçalı model aktarma stratejileriyle değişik ağ büyüklüklerinde gerçekleştirdik.

ANAHTAR KELİMELER: Epidemik Protokol, Fısıltı Protokolü, Tam Dağıtık Derin Öğrenme, Yapay Sinir Ağları

JÜRİ: Dr. Öğr. Üyesi Mustafa Berkay YILMAZ

Prof. Dr. Cafer ÇALIŞKAN

Dr. Öğr. Üyesi Murat AK

ABSTRACT

IMPROVING GOSSIP-BASED DECENTRALIZED DEEP LEARNING

Yunus BÜTÜN

MSc Thesis in Computer Engineering

Supervisor: Asst. Prof. Dr. Mustafa Berkay YILMAZ

June 2022; 50 pages

In this study we have studied fully-decentralized learning which is based on exchanging trained local models randomly using gossip protocols between nodes. Unlike other partially decentralized learning methods, in this approach there is no leader or coordinator node. The same algorithm executes in each node. Every node averages received models and trains the averaged model with its local data. Although it is theoretically not proved, the method works quite well in practice. In order to understand the algorithm, it is necessary to understand the underlying communication protocol used to distribute trained models. Gossip learning originally proposed based on push type gossip protocol. We have observed that there is little amount of study in this field. Since improvement of decentralized learning strongly depends on application of underlying protocols in an appropriate way and model merging strategies. In the first part of this research, we have studied all gossip protocols extensively both in SI and SIR models theoretically and empirically. We have derived numerous equations such as predicting peak time when network load reaches its maximum value in push type gossip in the SIR model. In most existing research, gossip algorithms are used to train linear models such as SVM and logistic regression. In the second part of this study, we have used neural networks for training on distributed data using gossip learning algorithm with various partial model transfer strategies for different network sizes.

KEYWORDS: Gossip Protocol, Epidemic Protocol, Fully Distributed Deep Learning, Artificial Neural Networks

COMMITTEE: Asst. Prof. Dr. Mustafa Berkay YILMAZ

Prof. Dr. Cafer ÇALIŞKAN

Asst. Prof. Dr. Murat AK

ACKNOWLEDGEMENTS

Firstly, I want to thank to my advisor Asst. Prof. Dr. Mustafa Berkay YILMAZ who gave me the opportunity to explore this interesting topic on my own, and helped me whenever necessary. I would also thank to Asst. Prof. Dr. Murat AK, the idea was originated from a course project that we have worked on.

I dedicate this study to all good people and animals that do not build their body with life of innocents.

LIST OF CONTENTS

TEXT OF OATH

I declare that this study “IMPROVING GOSSIP-BASED DECENTRALIZED DEEP LEARNING”, which I present as master thesis, is in accordance with the academic rules and ethical conduct. I also declare that I cited and referenced all material and results that are not original to this work.

27/06/2022
Yunus BÜTÜN

LIST OF SYMBOLS AND ABBREVIATIONS

Symbols

β : Probability of successful delivery of payload message

Δ : The time delay between 2 successive rounds

s : Fraction of susceptible nodes

S : Number of susceptible nodes

i : Fraction of infected nodes

I : Number of infected nodes

γ : Probability of switching to removed state for infected nodes

M_C : Number of communication messages

M_P : Number of payload messages

N : Total number of active nodes in the network

p_{ss} : Probability that a newly infected node can infect another node within same round

r : Fraction of removed nodes

τ : Time delay

T_{end} : Time when infection ends

t_i : Time when an update injected into to the system

T_{peak} : Time when number of infected nodes reach its maximum value

Period (.) was used as the decimal separator.

Abbriviations

SGD : Stochastic Gradient Descent

SI : Susceptible-Infected

SIR : Susceptible-Infected-Removed

SVM : Support Vector Machine

UAV : Unmanned Aerial Vehicle

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Gossip is a way of natural information dissemination among human individuals. Even in absence of any central coordinator, information can spread very fast. One key property of gossip is robustness. Once a few humans start to gossip about a piece of information, it becomes very hard to stop the dissemination process since information flows through random paths with exponentially increasing speed. Starting gossip is easy and doesn't require much effort but stopping gossip in a well-connected society is very hard if not impossible.

Spread of gossip is very similar to epidemic spreading. Where instead of information, virus is being spread. And indeed, theoretical research in gossip protocols mostly rely on results from epidemiology. Thus, many terminologies are borrowed from epidemiology. In literature, sometimes gossip and epidemic words are being used interchangeably to describe these types of protocols.

However, despite many similarities, as we will have shown, gossip protocols are quite different and more complex. However, spread of biological viruses with epidemics has become an inspiration to computer science for designing large scale, fail tolerant and self-organizing decentralized systems.

Following the bitcoin revolution, fully distributed systems are increasingly gaining popularity in recent years. Most blockchain applications rely on gossip protocols for distributing transactions. Gossip protocols are the only leaderless and truly decentralized communication protocols.

The basic idea of gossip style communication is each peer periodically communicates with other peers and exchanges information. In the case of randomized gossip, the selected peers are chosen randomly from all networks or from a subset of nodes. On the other hand, in spatial gossip each peer chose its neighbors only. These two approaches can be combined where a peer chose a random node from networks with a probability function, such as choosing nodes in near proximity with higher probability.

Gossip protocols are a collection of protocols which consist of push, pull and push-pull protocols. Push and push-pull protocols can be operated in both SI and SIR modes, but pull is SI only protocol. In this thesis, we have extensively analyzed these protocols in different conditions.

In machine learning, training large models require utilization of multiple machines since the amount of training data exceeds computation power of a single machine. And in neural network like algorithms required sample size increases exponentially with number of parameters (Verbraeken et al. 2020).

Distributing training workload is done in two different ways. In the data parallel approach, training dataset is split among each node with the same training algorithm.

In the model parallel approach, each node has a different part of a global model. The same dataset is being used by each node. This approach can be used when the model

size is very large to be stored on a single machine. But cannot be applied if model parameters are not separable.

Depending on communication constraints, this parallelism can be categorized as synchronous or asynchronous. In synchronous approach each node waits for all of the other nodes to complete their work for each gradient step. Thus, the slowest machine is poses constraint on training time.

In asynchronous approach a parameter server is used to eliminate latency introduced by slow nodes but in this case parameter server itself form communication bottleneck since it has to handle all incoming communication from worker nodes and send them back the aggregated model (Dean et al. 2012).

There are three popular topologies in real-life distributed machine learning applications. Tree topologies have lower communication overhead, since every node needs to communicate only with its parent and children. Local gradients are passed from children to parents toward the root of tree. Then aggregated gradient broadcasted back from top to down. Tree topologies are easy to control and scalable. But they are not failure resilient. Failure of a node can render a subtree rooted at that node useless. If root fails the system fails (Agarwal et al. 2014).

Ring topologies like tree topologies have low communication overhead. Each node needs two communicate with its neighbor nodes only. But failure resilience is the main problem here. Failure of a single node will render the system useless, and the probability of failure increases with the number of nodes.

Parameter Server topology consists of a set of master nodes, each with their own set of worker nodes. The topology has low communication overhead as each worker only need to communicate with their master and each master communicate with other master nodes and its worker nodes. The system has better failure resilience than tree and ring topologies, but it has scalability issue. As the number of nodes increases, parameter servers will form bottleneck since they have to handle all communication (Wei et al. 2015).

Another alternative topology is peer-to-peer topology, which is still under development and not implemented in real-life applications yet. In this fully distributed method, each node has a portion of the dataset. Training is done locally on each node and model parameters are exchanged through gossip protocols. The topology is failure resilient and highly scalable, but convergence is slow compared to centralized training (Orm´andi et al. 2013).

2. LITERATURE REVIEW

Frieze and Grimmett (1985) have shown that the expected number of rounds required to disseminate information from a single node to all nodes in a fully connected network is $O(logN)$ in terms of number of nodes in the network.

Pittel (1987), in probability, proved that expected number of rounds to fully inform all nodes can be calculated as shown in equation (3.3).

Gossip protocols initially introduced by research in Xerox. In the original paper, SI type gossip protocols was called anti-entropy and SIR type push protocol was called rumor mongering. Protocols used to replicate a distributed key value dataset across the globe on a network of approximately 1000 nodes. Each node has a full view of the network and the process is synchronous. The network was assumed to be almost static without churn (Demers et al. 1987).

In order to decrease load on transatlantic links, they have also studied spatial gossip in which nodes chose geographically close peers with higher probability. For theoretical analysis of the SIR push protocol, they have used a modified version of the epidemic SIR model to show the fraction of uninfected nodes at the end of the dissemination process. Although it can be used for this particular purpose as we have shown in Section 4.1.2 this approach cannot be used to model the discrete process of synchronous push protocol in SIR model. We have addressed the problem with introducing a delay which is measured empirically for a wide range of node sizes.

Karp et al. (2000) have provided an upper bound for required number of rounds to inform all nodes in a fully connected network for push-pull gossip protocol and proposed an algorithm to decrease the number of communication messages.

In the initial proposed gossip protocols, nodes have a list of all other nodes in the network. In dynamic networks nodes continuously join or leave the network which is called churn. In presence of churn or failures, maintaining a complete list of other nodes will not be feasible for large networks. Even same nodes can have different network address when they rejoin because of DHCP or position change. If not maintained properly, the list will grow continuously with invalid node addresses and randomly chosen peer address from the list will be invalid with increasing probability by time.

A naive approach to solve this problem would be periodically sending heartbeat messages to each node to check their state. This requires sending $O(N^2)$ heartbeat messages at each round, which is obviously not scalable. Jelasity et al. (2004) have addressed this problem by using a gossip-based peer sampling service. Each node has only a partial view of the full node list. Nodes keep their partial-view up-to-date by periodically exchanging a portion of it with other nodes using gossip protocol again. Nodes use gossip protocol to mix their view with the view of other nodes. Each node address has a counter, and nodes reset their own counter and increase the counter of other node addresses when exchanging views with other nodes. Only half of the local view is being exchanged, which is selected mostly from addresses with low counter value. Thus, if a node remains passive for a long time, it will automatically disappear from views of

all nodes. Many other peer sampling services proposed with similar ideas (Voulgaris et al. 2005; Johansen et al. 2015).

Gossip learning was proposed by Orm´andi et al. (2013) to solve binary classification problems using Pegasos and Adaline algorithms on fully distributed data in which each node has only one data sample. It was assumed the network is fully connected and processing speed of nodes is homogenous.

The idea was based on continuous averaging of models using push gossip protocol. Each node caches recently received models and predictions are done using weighted voting on the cached models. Nodes continuously exchange models and merge two most recently received models by simple averaging. Then averaged models are sent to a random peer again. No stopping criteria is considered. Thus, the process repeats forever.

Giaretta and Girdzijauskas (2019) simulated gossip learning algorithm using SVM and linear regression algorithms for the case each node has multiple data points. The algorithm was also tested on not fully-connected networks such as random, Barabasi-Albert and ring topologies with non-homogeneous processing speeds. It was shown that the algorithm still converges to a correct model if processing speed of nodes and data are not correlated. Otherwise, training results in a less accurate global model. The study also show that the algorithm still converges in not fully-connected networks, although convergence speed is slower.

Hegedűs et al. (2019) have used gossip learning to train logistic regression algorithm on spambase dataset. And Gossip learning was compared to federated learning. Authors claim that gossip learning outperforms federated learning if data are uniformly distributed among nodes.

Bagoly and Danescu (2020) used gossip learning with to train neural networks on MNIST dataset. Different model merging strategies were proposed in this paper such as keeping a large cache of received models and merging the best ones or averaging over all of them instead. In the study it was shown that this approach increases accuracy of global model but convergence speed is similar to generic gossip learning algorithm. This approach requires nodes to have larger amount of memory and extra amount of processing.

Alkathiri et al. (2021) have used gossip learning to train word2vec algorithm in a decentralized setting. Result show that the algorithm is comparable to centralized training with only 4% drop in accuracy.

And recently, Zecchin et al. (2022) proposed using UAVs to speed up convergence speed in decentralized training over sparsely connected networks. In which UAVs follow an optimum path to convey models between disconnected communities.

3. MATERIAL AND METHOD

3.1 Gossip Protocols

3.1.1 Push protocol

Pseudocode of generic SI push algorithm is given in Figure 3.1. This algorithm is independently executed in each node. In order to join the network, a node needs to know the address of an active node. Node gets a list of addresses of all nodes by as a reply of join request. Each node initially joins the network in susceptible state for a fixed update. Since multiple updates can propagate in the network at the same time, a node can be in susceptible state for an arbitrary update and at the same time it can be in infected state for another update.

Algorithm 1: Push Gossip Algorithm in SI Model

```
1  viewList ← JoinRequest(knownNode)
2  state ← susceptible
3  while state==infected do
4  |  Wait(Δ)
5  |  p = SelectPeer(viewList)
6  |  Send(data, p)
7  end while
8  procedure OnUpdateReceived(receivedUpdate)
9  |  state ← infected
10 |  data ← receivedUpdate
11 end procedure
12 procedure OnJoinRequestReceived(sender)
13 |  Send(sender, viewList)
14 end procedure
```

Figure 3.1. Push gossip algorithm in SI model

Discrete deterministic equation for push protocol is given in (Demers et al. 1987) as shown below

$$s(t+1) = s(t)\left(1 - \frac{1}{N}\right)^{(1-s(t))N} \tag{3.1}$$

$s(t)$ is the fraction of susceptible nodes at round t. Assuming update is injected into the network from a single node, $s(0)$ can be calculated as $1 - 1/N$ where N denotes number of nodes in the network. Assuming that each node independently choses another node, and independent of previous decisions. Expected number of susceptible nodes, at round $t + 1$ can be calculated using equation (3.1).

Using Taylor series expansion of e^{-x} around 0 equation (3.1) can be simplified further as

$$\mathrm{s(t+1)} \approx s(t)e^{-\left(\frac{1}{N}\right)(1-s(t))N} \tag{3.2}$$

Pittel (1987), in probability, estimated that minimum number of rounds required to infect all nodes can be calculated as in equation (3.3) assuming all messages are delivered successfully with probability of 1.

$$T(N,f) = \log_{f+1} N + \frac{\log_e N}{f} \pm O(1) \text{ as } N \to \infty \tag{3.3}$$

f denotes gossip fanout which is defined as number of push messages sent by each node during one round. In equation (3.1) gossip fanout is 1. Toward the end of dissemination $s(t)$ becomes negligible and during this phase number of susceptible nodes shrinks exponentially as shown in (Demers et al. 1987) as

$$s(t+1) \approx s(t)e^{-1} \tag{3.4}$$

A closer look at equation (3.1) reveals similarity with balls and bins problem in which probability that a bin will stay empty when throwing m balls into N bins, which can be calculated as in following equation.

$$p = \left(1 - \frac{1}{N}\right)^m \tag{3.5}$$

Exponent $\big(1 - s(t)\big)N$ in equation (3.1) is equal to number of infected nodes at round t. Which is actually corresponds to number of push messages sent during round t. Using equation (3.5) we can rewrite equation (3.1) in a more general form by including fanout factor, f, and probability that a payload message successfully will be delivered which is denoted by β.

$$s(t+1) = s(t)\left(1 - \frac{1}{N}\right)^{i(t)N\beta f} \tag{3.6}$$

In the following sections, we will refer to equation (3.6) as balls and bins equation in SI model. Another approach to model push protocol is using a modified form of compartmental epidemic model proposed by Kermack and McKendrick (1927). The SIR model for push protocol is given as

$$\frac{ds}{dt} = -\beta s(t)i(t) \tag{3.7}$$

$$\frac{di}{dt} = \beta s(t)i(t) - \gamma(1 - s(t))i(t) \tag{3.8}$$

$$\frac{dr}{dt} = \gamma(1 - s(t))i(t) \tag{3.9}$$

$$s_0 = 1 - \frac{1}{N}, i_0 = \frac{1}{N}, r_0 = 0 \tag{3.10}$$

s_0, i_0 and r_0 are initial values at time t_0. In the original SIR epidemic model, equation (3.7) and equation (3.8) are written as

$$\frac{di}{dt} = \beta s(t)i(t) - \gamma i(t) \tag{3.11}$$

$$\frac{dr}{dt} = \gamma i(t) \tag{3.12}$$

In gossip SIR model, second term in equations (3.11) and right-hand side of equation (3.12) multiplied by $(1 - s(t))$ which denotes probability that an infected node comes in contact with a non-susceptible node and γ denotes probability that an infected node switches to removed state when it encounters an already infected node.

β denotes probability that an infection occurs when an infected and a susceptible person comes in contact. r denotes fraction of removed nodes. In SI model γ =0. Thus, the system of nonlinear differential equations given by equations (3.7, 3.8, 3.9) reduces to a single separable non-linear differential equation as

$$\frac{ds}{dt} = -\beta s(t)(1 - s(t)) \tag{3.13}$$

Exact solution to equation (3.13) is given in (Bailey 1975) as

$$s(t) = \frac{1}{\left(\frac{1 - s_0}{s_0}\right) e^{\beta(t - t_0)} + 1} \tag{3.14}$$

An exact solution to SIR epidemic model was obtained by Harko et al. (2014) in integral form, which can be solved using numerical integration techniques. But to the best of our knowledge for SIR push gossip model a solution is not obtained yet. We were able to obtain an exact solution in integral form as given in equations (3.15, 3.16, 3.17). Detailed derivation is given in Section 7.1.

$$\int_{s_0}^{s} \frac{ds}{-\beta s\left(-\left(1 + \frac{\gamma}{\beta}\right)s + \frac{\gamma}{\beta}\ln\left(\frac{s}{s_0}\right) + \left(1 + \frac{\gamma}{\beta}\right)s_0 + i_0\right)} = t - t_0 \tag{3.15}$$

$$i = -\left(1 + \frac{\gamma}{\beta}\right)s + \frac{\gamma}{\beta}\ln\left(\frac{s}{s_0}\right) + \left(1 + \frac{\gamma}{\beta}\right)s_0 + i_0 \tag{3.16}$$

$$r = 1 - i - s \tag{3.17}$$

By approximating logarithmic term in the denominator of equation (3.15) using first 2 terms of Taylor series expansion around s_0 we obtained an approximate equation for peak time when number of infected nodes reach its maximum value.

$$t_{peak} = t_0 - \frac{1}{\beta\left(i_0 + \left(\frac{\gamma}{\beta} + 1\right)s_0 - \frac{\gamma}{\beta}\right)}\left(\ln\left(\frac{s_p}{s_0}\right) + \ln\left(\frac{\left(\gamma/\beta s_0 - 1 - \frac{\gamma}{\beta}\right)s_0 + \left(i_0 + \left(\frac{\gamma}{\beta} + 1\right)s_0 - \frac{\gamma}{\beta}\right)}{\left(\gamma/\beta s_0 - 1 - \frac{\gamma}{\beta}\right)s_p + \left(i_0 + \left(\frac{\gamma}{\beta} + 1\right)s_0 - \frac{\gamma}{\beta}\right)}\right)\right) \tag{3.18}$$

Where s_p is fraction of susceptible nodes at the time t_{peak}. And can be calculated as

$$s_p = \frac{\gamma}{\gamma + \beta} \tag{3.19}$$

Detailed derivation is given in Section 7.2. We have also obtained a discrete equation for push protocol in SIR model by combining equation (3.6) and equation (3.16) for fanout value of 1 as

$$s(t+1) = s(t)\left(1 - \frac{1}{N}\right)^{\left(-\left(1+\frac{\gamma}{\beta}\right)s + \frac{\gamma}{\beta}\ln\left(\frac{s}{s_0}\right) + \left(1+\frac{\gamma}{\beta}\right)s_0 + i_0\right)N\beta} \tag{3.20}$$

Demers et al. (1987) have shown that some of susceptible nodes in push SIR model will never receive update. The fraction of susceptible nodes will remain uninfected can be calculated using the following implicit equation as

$$s_u = e^{-((\gamma+1)(1-s_u))/\gamma} \tag{3.21}$$

In sections 4.1.1 and 4.1.2 we have extensively analyzed push protocol and compared results obtained from derived equations with agent-based simulations based on implementation of algorithm given in Figure 3.1.

3.1.2 Pull protocol

In the pull protocol unlike push protocol only susceptible nodes are active. At each round each susceptible node selects one random node from its view list and sends a request message, if the selected node has a newer update it will reply with a payload message. Nodes continuously search for newer updates by sending request messages to randomly selected nodes. Pseudocode of pull algorithm is given in Figure 3.2.

Algorithm 2: Pull Gossip Algorithm in SI Model

```
1  viewList ← JOINREQUEST(knownNode)
2  state ← susceptible
3  while state==susceptible do
4  |  WAIT(Δ)
5  |  p = SELECTPEER(viewList)
6  |  SENDREQUEST(p)
7  end while
8  procedure ONUPDATERECEIVED(receivedUpdate)
9  |  state ← infected
10 |  data ← receivedUpdate
11 end procedure
12 procedure ONUPDATEREQUEST(p)
13 |  if state==infected then
14 |  |  SEND(p, data)
15 |  end if
16 end procedure
17 procedure ONJOINREQUESTRECEIVED(sender)
18 |  SEND(sender, viewList)
19 end procedure
```

Figure 3.2. Pseudo code for pull protocol

Formula for pull gossip is given by Demers et al. (1987) as

$$s(t+1) = s^2(t) \tag{3.22}$$

In Section 4.1.3 we compared equation 3.22 with simulation for 100,5000 and 20000 nodes. And in Section 4.1.6 pull protocol was compared to push and push-pull protocols in terms of convergence speed, communication and payload message overhead.

3.1.3 Push-pull protocol

Push-pull protocol is a combination of push and pull protocols. At each round, both susceptible and infected nodes are active. If a node is susceptible, it will search for updates by randomly selecting nodes. If a node is infected, it will try to distribute an update. Pseudocode of push-pull algorithm is given in Figure 3.3.

Algorithm 3: Push-Pull Gossip Algorithm in SI Model

```
1  viewList ← JoinRequest(knownNode)
2  state ← susceptible
3  while true do
4  |  Wait(Δ)
5  |  p = SelectPeer(viewList)
6  |  if state==infected then
7  |  |  Send(data, p)
8  |  else
9  |  |  SendRequest(p)
10 |  end if
11 end while
12 procedure OnUpdateReceived(sender,receivedUpdate)
13 |  state ← infected
14 |  data ← receivedUpdate
15 end procedure
16 procedure OnUpdateRequest(p)
17 |  if state==infected then
18 |  |  Send(p, data)
19 |  end if
20 end procedure
21 procedure OnJoinRequestReceived(sender)
22 |  Send(sender, viewList)
23 end procedure
```

Figure 3.3. Pseudo code of push-pull algorithm

A similar recurrence relation analogous to push and pull protocol can be derived by combining equation (3.1) and equation (3.22) as

$$s(t+1) = s^2(t)\left(1 - \frac{1}{N}\right)^{i(t)N} \tag{3.23}$$

As shown in Figure 4.23 equation (3.23) perfectly matches simulation results for 100, 5000 and 20000 nodes.

Using equation (3.23) we have derived a continuous analytic approximation for push-pull protocol as

$$\frac{ds}{dt} \approx s^3 - s \tag{3.24}$$

Solution of equation (3.24) was obtained as

$$s(t) \approx \frac{1}{\sqrt{1 + \left(\frac{1 - s_0^2}{s_0}\right) e^{2(t - t_0)}}} \tag{3.25}$$

Because of error in approximating discrete process as continuous process, number of susceptible nodes decrease faster. In order to compensate the error, we introduced a time delay, τ, into equation (3.25) as

$$s(t) \approx \frac{1}{\sqrt{1 + \left(\frac{1 - s_0^2}{s_0}\right) e^{2(t - t_0 - \tau)}}} \tag{3.26}$$

Where

$$t \geq t_0 + \tau \tag{3.27}$$

Detailed derivation of equation (3.26) is given at Section 7.4. It is observed that amount of delay for 10000 to 200000 nodes remains only 3 rounds. The order of growth of this delay with respect to number of nodes is $o(logn)$ as shown in Figure 4.27.

As shown in Figure 4.28 equation (3.26) approximates the recurrence relation given in equation (3.16) quite well. Equation (3.19) can be used to find the T_{end} by choosing a s_{end} value such as

$$s_{end} = \frac{N - 0.9}{N} \tag{3.28}$$

Since $s(t)$ in equation (3.26) can never be zero, instead we can choose a s_{end} value such as number of susceptible nodes drops below 1.

Next, we have derived an approximate analytic SIR model for push-pull protocol using equation (3.24) as given in equations (3.29, 3.30, 3.31, 3.32)

$$\frac{ds}{dt} = s^3 - s \tag{3.29}$$

$$\frac{di}{dt} = -s^3 + s - \gamma(1 - s)i \tag{3.30}$$

$$\frac{dr}{dt} = \gamma(1 - s)i \tag{3.31}$$

$$s_0 = 1 - \frac{1}{N}, i_0 = \frac{1}{N}, r_0 = 0 \tag{3.32}$$

The system of differential equations was solved numerically using Runge-Kutta 4th order method and compared to simulation results. With applying same delay values obtained in Figure 4.26 for SI model, it is observed that numerical solution and simulation matches quite well. In Section 4.1.4 for different N, γ and β values, numerical solution of SIR push-pull model compared to simulation. In Section 4.1.5 SIR push-pull model was compared to SIR push model.

3.1.4 Assumptions

If not explicitly stated otherwise, experiments are performed in synchronous mode and values of β, γ and f are 1. Network is fully connected.

Push and push-pull protocols send payload messages only if their push messages replied with a request for payload.

3.2 Gossip Learning

Pseudocode for generic gossip learning algorithm is given in Figure 3.4. This algorithm is based on push method and executed in each node forever. We have implemented the algorithm using agent-based modelling technique. Each node keeps 2 different model. Last model is the model received from one of other nodes in the previous round. Current model is the last model trained with local data after being averaged with previously received model. In another word, last 2 received models are merged and trained with local data. Then the trained model is sent to a random peer at the next round.

In previous research, full models were being transferred between nodes. Which causes high load on network if models are large like image classification problems. We have tried to find an answer to the research question: "Can we transfer models partially in order to decrease bandwidth usage?". 3 different partial model transfer methods were tested on a neural network with 4 layers on a binary classification task using spambase dataset (Anonymous 1).

In the first method, gossip algorithm was tested by transferring 2 out of 4 layers at a time. Selected 2 layers were successive. And at each round, window moves forward by 1 unit. For example, at first round if layers 1 and 2 were transferred, in next round layers 2 and 3 will be transferred.

In second method we have divided each layer into 4 part and in each round from each layer we randomly selected one part. Thus, in 1 round only 25% of weight blocks were transferred.

In the third method, we have extended the second method by selecting 2 successive parts from each layer. Thus, in 1 round 50% of weigh blocks are being transferred.

In order to test the gossip learning approach in the scenario where individual nodes doesn't have enough data for training, we have assigned 10 data samples to each node. Training was performed using both local-only mode, where nodes don't exchange models.

Results were compared with the case where nodes exchange models using push protocol both in full and partial model transfer strategies.

Algorithm 3 Generic Gossip Learning Algorithm

```
1: procedure MAIN
2:     currentModel ← INITMODEL()
3:     lastModel ← currentModel
4:     while true do
5:         WAIT(Δ)  ▷ training and listening for new model
6:         p ← RANDOMPEER()  ▷ Select an random peer
7:         SEND(p, currentModel)  ▷ Then send current
   model
8:     end while
9: end procedure
10: procedure ONMODELRECEIVED(m)
11:     averageModel ← MERGE(m, lastModel)  ▷ new
   model merged with cached model
12:     currentModel ← UPDATE(averageModel)  ▷
   Updating current model
13:     lastmodel ← m  ▷ Caching received model
14:     TRAIN(currentModel)  ▷ Continuing training using
   local data
15: end procedure
```

Figure 3.4. Generic gossip learning algorithm

Next, we have trained a single node with all data available in other nodes for comparing centralized training with decentralized training with same amount of data.

Previous experiments were repeated for 20,40 and 80 nodes in order to observe behavior of the algorithm for different network sizes. The neural network architecture we have used is presented in Figure 3.5.

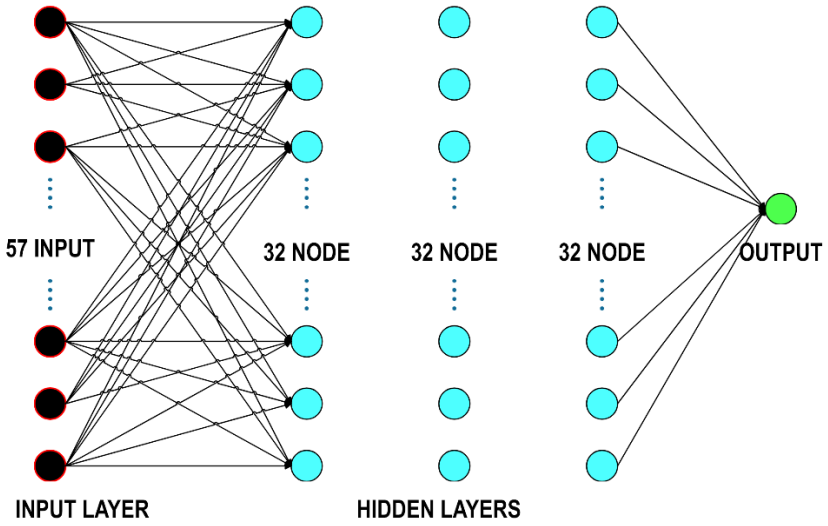

Figure 3.5. The neural network architecture used for training

Training parameters are given in the following table as

Table 3.1 Training Parameters

Training algorithm	SGD
Data samples per node	10
Epoch for each round	1
Batch size	2
Learning rate (constant)	0.1
Regularization parameter	0.5
Weight initialization	Uniform random (-1,1)
Activation Function	Sigmoid

Comparison was done based on test accuracy, which can be defined as measured percentage of correct prediction on test data. Test data consists of 30% of spambase dataset (1380 data samples). Dataset has 4601 instances with 57 attributes and 2 output classes.

4. RESULTS AND DISCUSSIONS

4.1 Gossip Protocol Experiments

4.1.1 Push protocol experiments in SI model

We have verified our implementation of algorithm given in Figure 3.1 with comparing equation (3.1) for different nodes sizes. Simulation results for push protocol in SI mode perfectly matches with results obtained using equation (3.1).

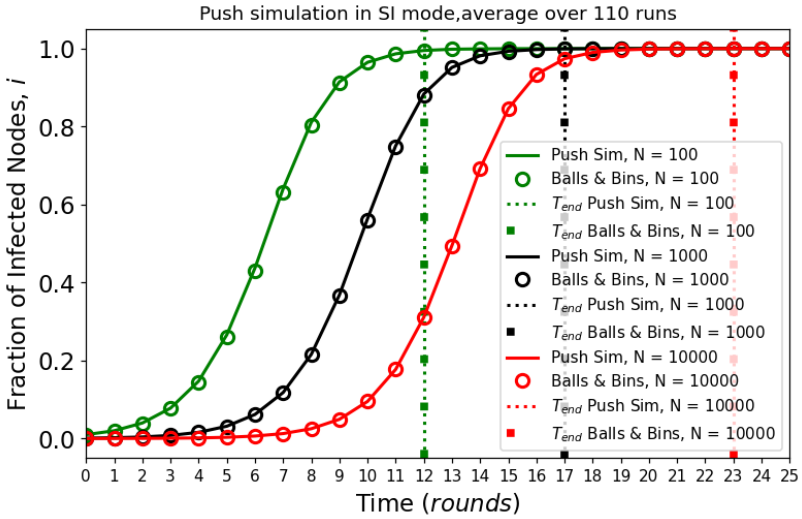

Figure 4.1. Verification of code with comparing simulation results with equation (3.1) for different node sizes

We have validated analytic solution of differential equation of SI push protocol given by equation (3.14) with comparing it with numerical solution obtained using Runge-Kutta 4th order method. Results validate analytic solution.

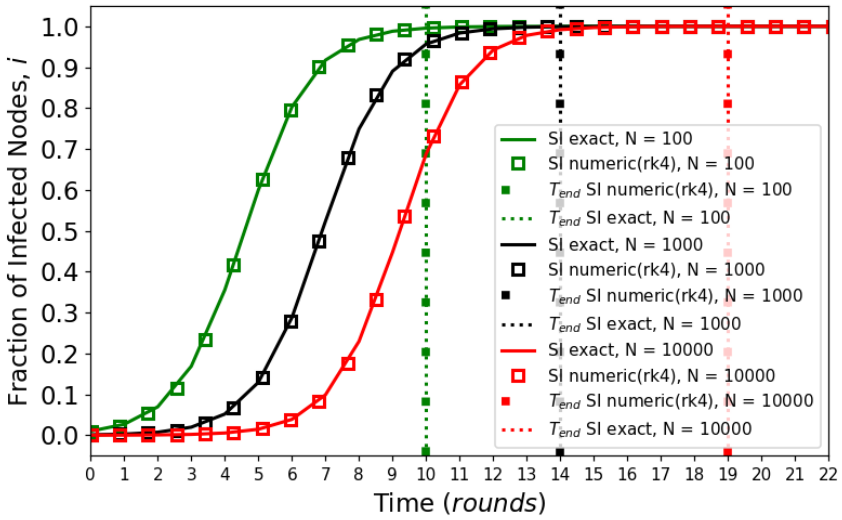

Figure 4.2. Verification of analytic solution of SI gossip model with comparing numerical solution obtained using Runge-Kutta 4th order method

Next, we compared analytic solution of SI model with equation (3.1). Results show that continuous differential equation model of SI push doesn't appropriate to describe discrete fully synchronous push protocol in SI model.

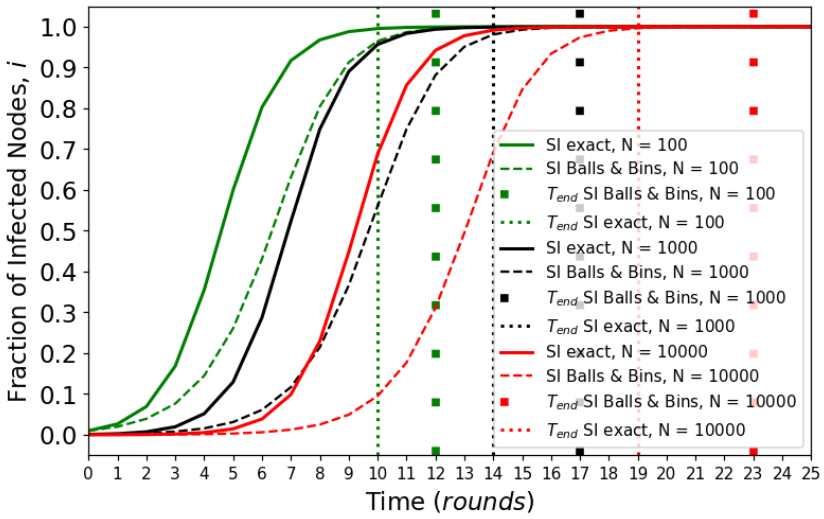

Figure 4.3. The comparison of analytic solution of SI push model with equation (3.1)

In order to discover the relationship between differential equation model and discrete model we have run experiments with extended discrete ball and bins based formula given in equation (4.1) that introduces some asynchronicity to the model by allowing newly infected nodes in a round can also infect other nodes with a constant probability, p_{ss}.

$$s(t+1) = s(t)\left(1 - \frac{1}{N}\right)^{N\beta f((i(t)+(s(t)-s(t+1))*p_{ss})} \tag{4.1}$$

Since in Figure 4.1 we have shown that ball and bins based equation (3.1) perfectly matches synchronous simulation results, we have used the equation instead of simulation to perform the experiment with large node sizes. Otherwise, it is computationally infeasible to simulate billions of nodes. By various p_{ss} values, results of differential equation model and discrete model were compared. Results show that differential equation perfectly describe the semi-synchronous model if p_{ss} values is around 0.688. p_{ss} values greater this values results faster propagation than differential equation model and smaller values results smaller propagation. Figure 4.3 shows that in full-synchronous model dissemination process is slower than differential equation model in which $p_{ss} = 0$

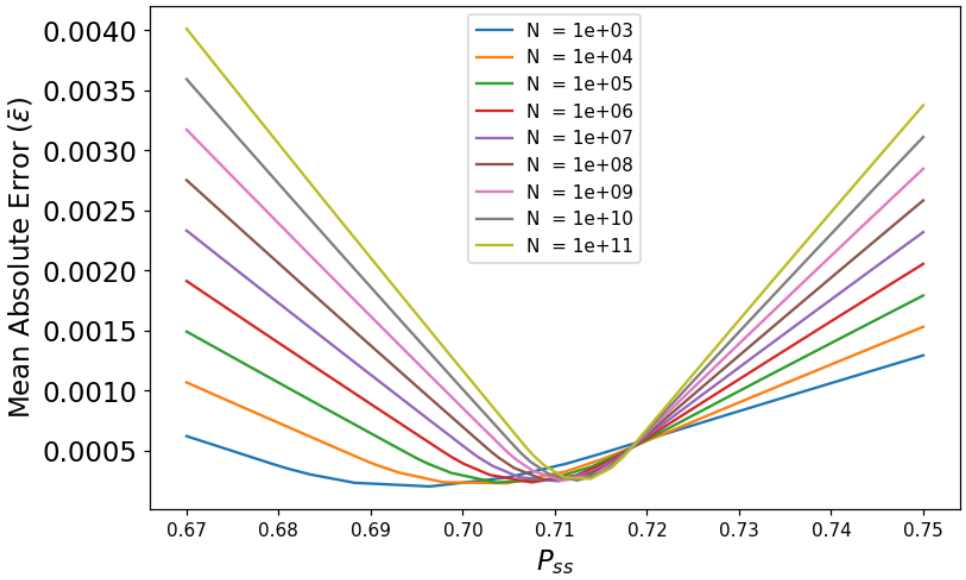

Figure 4.4. Mean absolute error between Infected curves of differential equation model and semi-synchronous discrete model for various p_{ss} values

Results obtained in previous experiment were validated by simulation for 100 and 10000 nodes. In both cases, results validate previous experiment. Thus, the differential equation model can only be used to describe push models only if newly infected nodes in a round also can send update to a random node with a probability p_{ss}= 0.711.

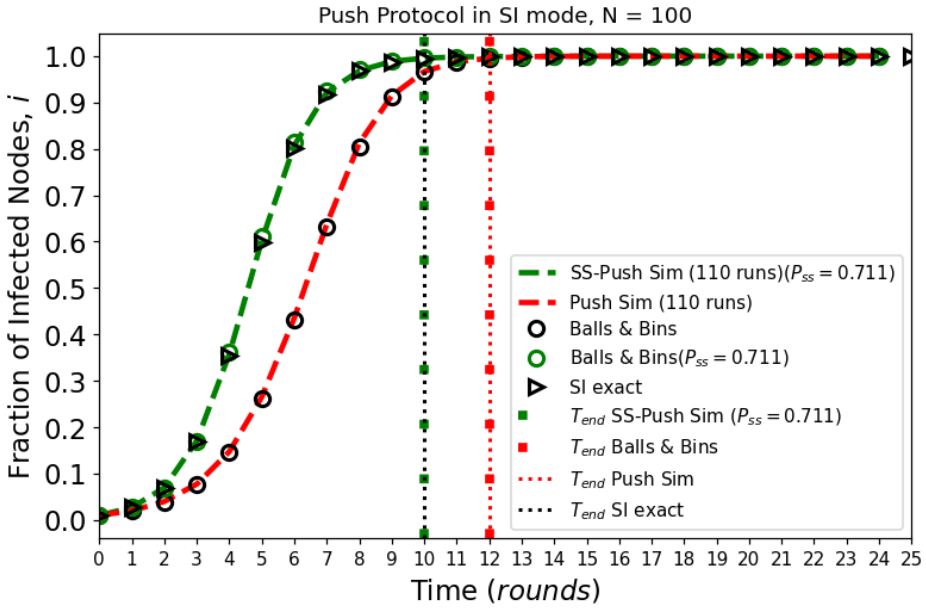

Figure 4.5. The comparsion of differential equation model with semi-synchronous and fully-synchronous push protocol simulation in SI model for N=100

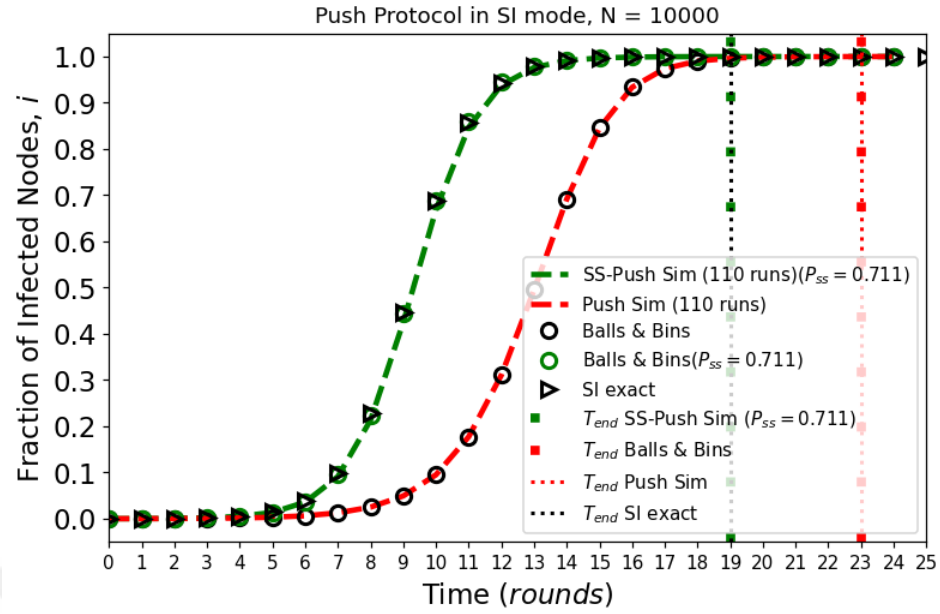

Figure 4.6. The comparison of differential equation model with semi-syncronous and fully-syncrnous push protocol simulation in SI model for N=10000

In the next experiment, we compared push protocol in SI mode with different fanout values. Results show that increasing fanout value almost results faster dissemination and deviation from average becomes smaller. Thus, T_{end} can be more predictable. Experiment repeated 200 runs and curves plotted with small opacity value. Thus, darker color means higher overlap. The histogram in the Figure 4.7 represents T_{end} values for different runs. For fanout value of 2, 121 runs out of 200 have ended at 7th round. For fanout value of 1 distribution is wider.

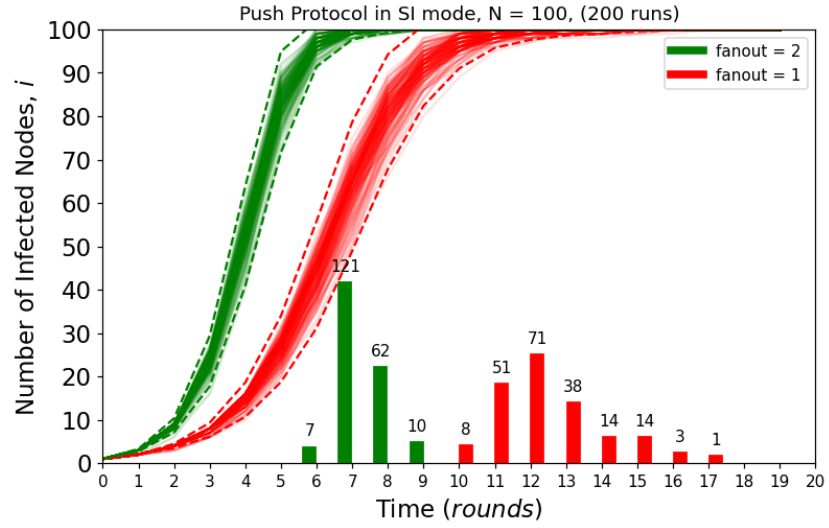

Figure 4.7. The comparsion of push protocol simulations in SI mode with different fanout values for N=100. Dashed lines represent a standard deviation of ±3 σ from mean.

Previous experiment repeated with larger node size. Results given in following figure shows that as node size grow, deviation from average becomes smaller.

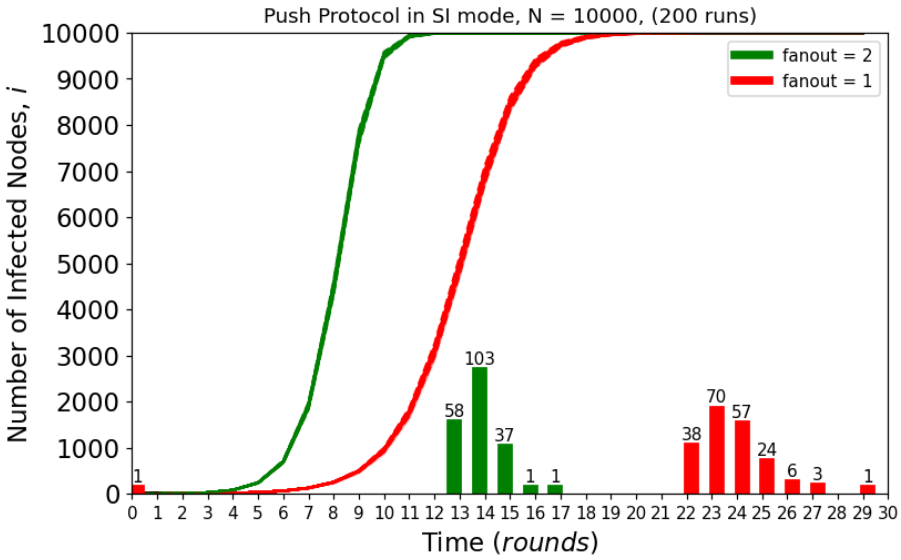

Figure 4.8. The comparison of push protocol simulations in SI mode with different fanout values for N=10000. Dashed lines represent a standard deviation of ±3 σ from mean

In order to see effect of fanout value on number of payload messages, we have simulated push protocol in SI mode with different node size as shown in Figure 4.9. It can be observed that as the number of nodes increase deviation from mean becomes smaller. Thus, the random process becomes more predictable. Results also show that doubling fanout results only 50% of increase in peak value.

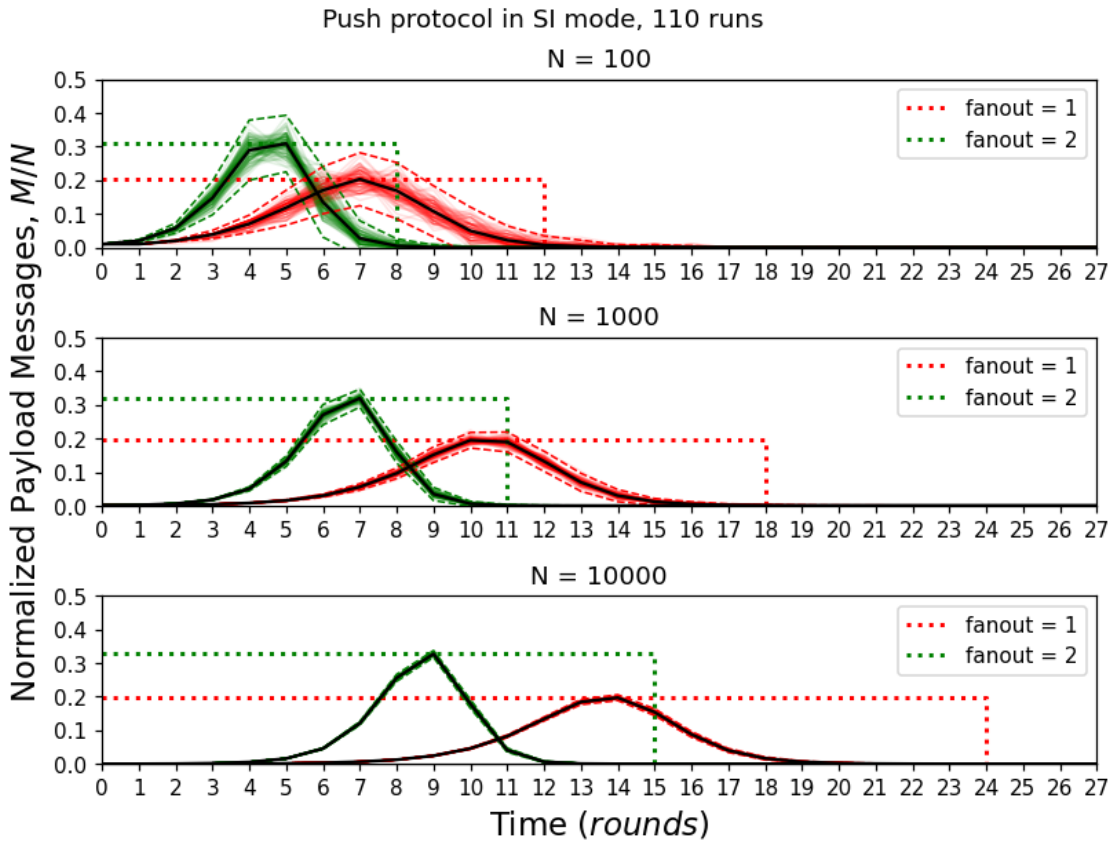

Figure 4.9. The comparison of number of payload messages that are being send for different node sizes. Dashed lines represent a standard deviation of ±3 σ from mean. Vertical dotted lines represent T_{end}

4.1.2 Push protocol experiments in SIR model

We have validated our exact solution to SIR model with results were obtained from numerical solution using Runge-Kutta 4th order method, which is most widely used method to solve nonlinear ordinary differential equation systems. Results presented in Figure 4.10 show perfect matching between the numerical solution and our solution.

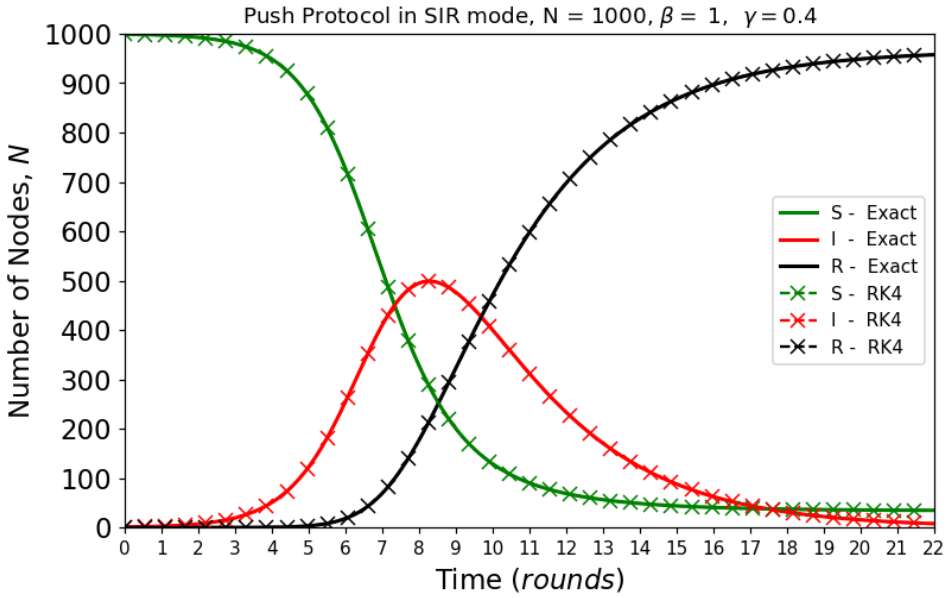

Figure 4.10. The comparison of numerical solution of SIR gossip model with exact solution

Next, we have compared simulation results with differential equation based model. We have observed same shift as in SI case. Thus, continuous differential equation model given by equations (3.7, 3.8, 3.9, 3.10) doesn't describe fully-synchronous push protocol in SIR model.

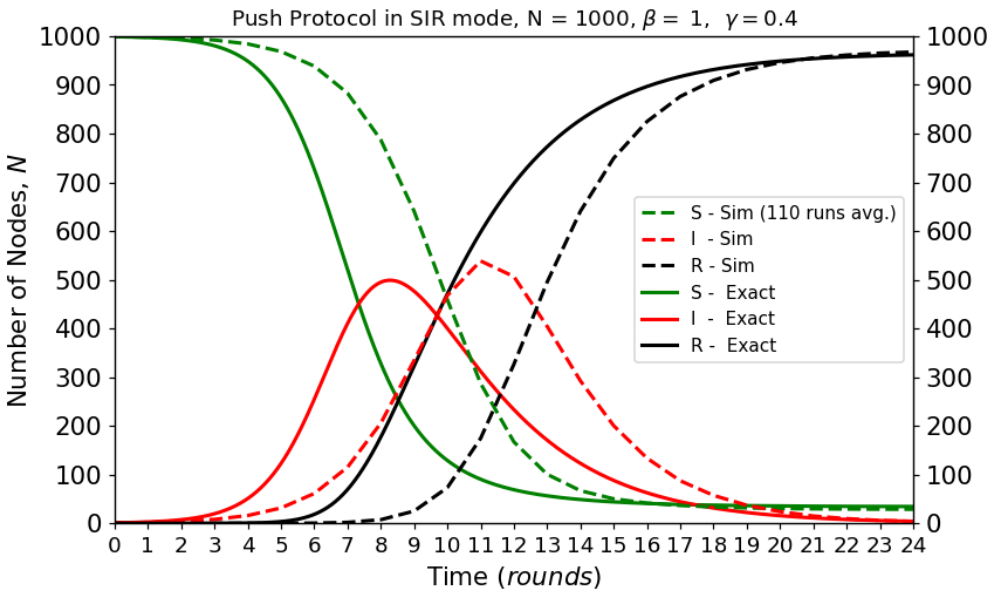

Figure 4.11. The comparison of exact solution of SIR compartmental gossip model with agent based SIR push protocol simulation

Although in previous experiment we have shown the difference between exact solution of differential equations and agent-based push protocol simulation the discrete equation we have derived from differential equation model perfectly matches simulation results for both small and large node sizes. (Figure 4.12, Figure 4.13)

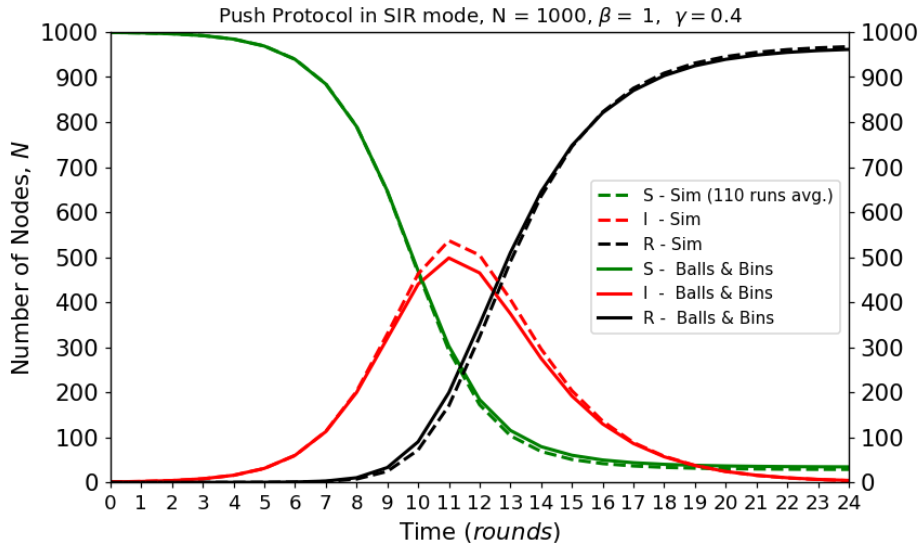

Figure 4.12. The comparison of agent based SIR push protocol simulation with equation (3.20) for N = 1000

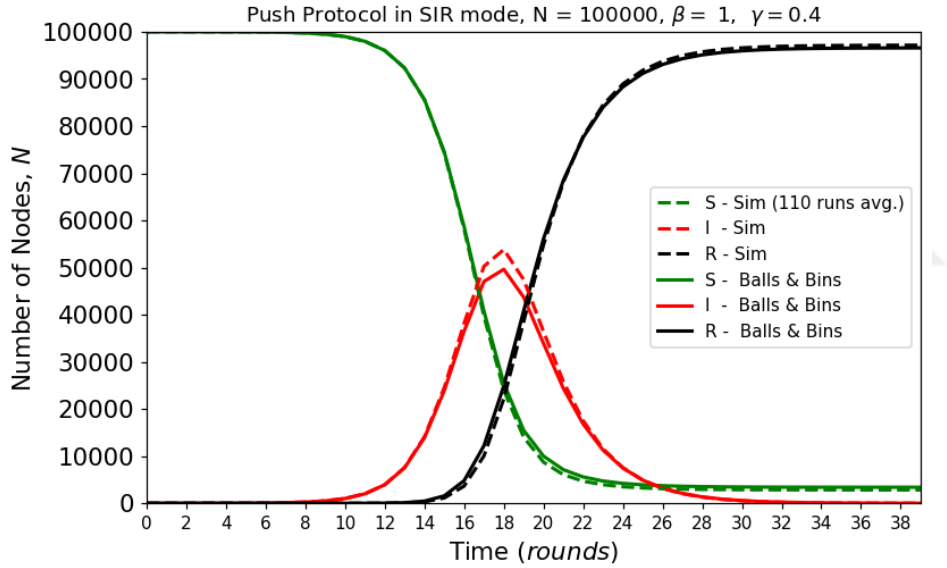

Figure 4.13. The comparison of agent based SIR push protocol simulation with equation (3.20) for N = 100000

We have repeated previous simulation with a wide range of node sizes and different value of γ and β values. Peak time when the number of infected nodes reaches a maximum value recorded and compared to results obtained using equation (3.20). Results show that equation (3.20) can be safely used to find peak time in synchronous push protocol in SIR mode. (Figure 4.14, Figure 4.15, Figure 4.14, Table 4.1)

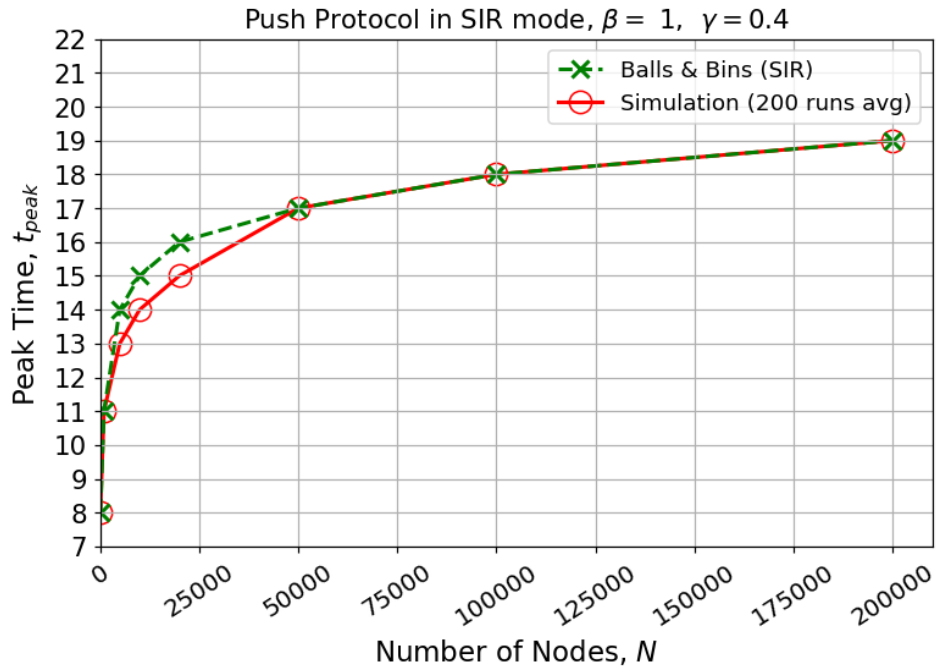

Figure 4.14. The comparison of T_{peak} values obtained from agent based SIR push protocol simulation with values obtained from equation (3.20) for the case of β = 1. Where payload messages reaches to target with no failure.

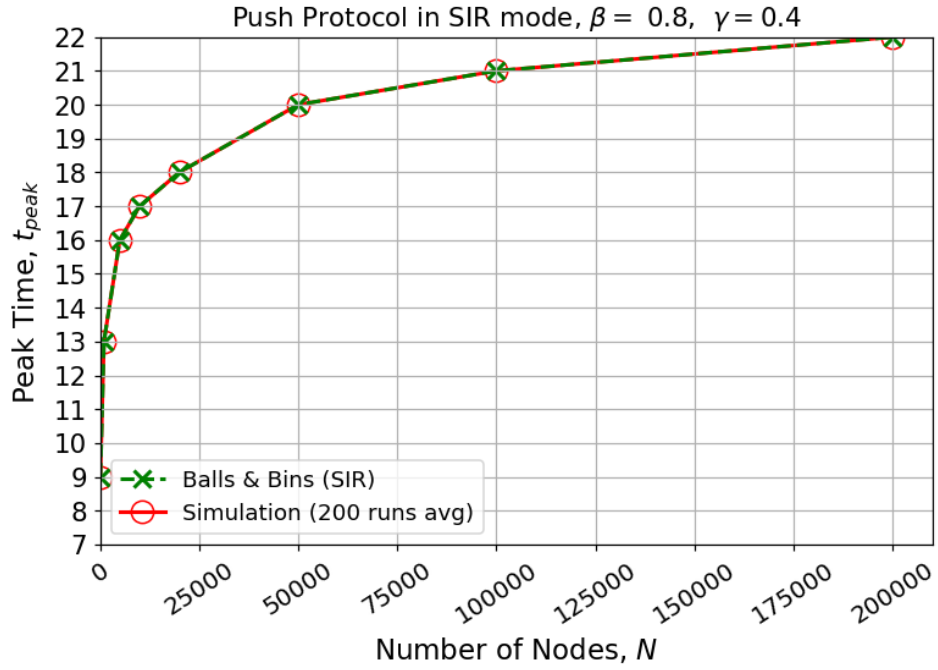

Figure 4.15. The comparison of T_{peak} values obtained from agent based SIR push protocol simulation with values obtained from equation (3.20) for the case of β = 0.8. Where payload messages fails to reach the target peer with 0.2 probability.

Table 4.1. The comparison of peak values obtained using equation (3.20) with simulation results

β = 1, γ = 0.2, 200 runs average						
N	t_{peak_sim}	t_{peak_eq}	$t_{peak\_sim} - t_{peak\_eq}$	i_{peak_sim}	i_{peak_eq}	**error (%)**
100	9	8	1	0.6636	0.63198	4.76
1000	13	12	0	0.66898	0.6406	4.24
5000	16	14	0	0.66901	0.64055	4.25
10000	17	15	0	0.66925	0.64054	4.29
20000	18	16	0	0.66929	0.64054	4.3
50000	20	18	0	0.66146	0.63187	4.47
100000	21	19	0	0.66118	0.63188	4.43
200000	22	20	0	0.66123	0.63188	4.44

In the next experiment, we have repeated previous simulation with a wide range of node sizes and different value of γ and β values. End time, T_{end}, the time when infection process terminates, was measured using simulation and results were compared to results obtained using equation (3.20). Results show that equation (3.20) can be safely used to find T_{end} in synchronous push protocol in SIR mode. (Figure 4.16)

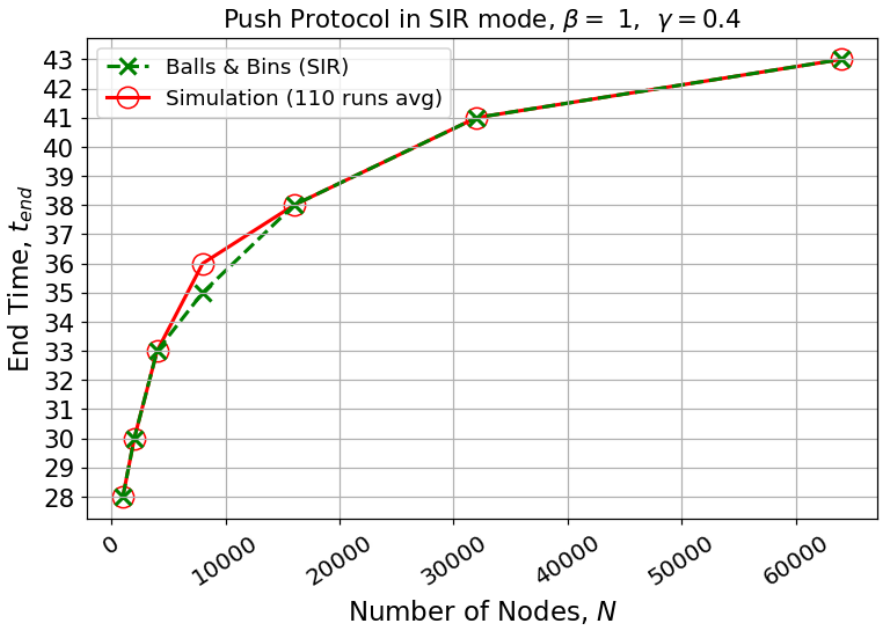

Figure 4.16. The comparison of T_{end} values obtained from agent based SIR push protocol simulation with values obtained from equation (3.20) for the case of β = 1. Where payload messages reaches to target with no failure probability

We have also compared our approximate peak equation (3.18) with peak values obtained using equation (3.20) to measure the amount of shift occurs between peak values. Experiments were repeated for different values of β, γ and node size ranges. Results show that amount of shift nearly constant. Thus, with a delay correction our deterministic peak formula given in equation (3.18) can be used define to location of peak with high accuracy. (Figure 4.17, Figure 4.18, Figure 4.19)

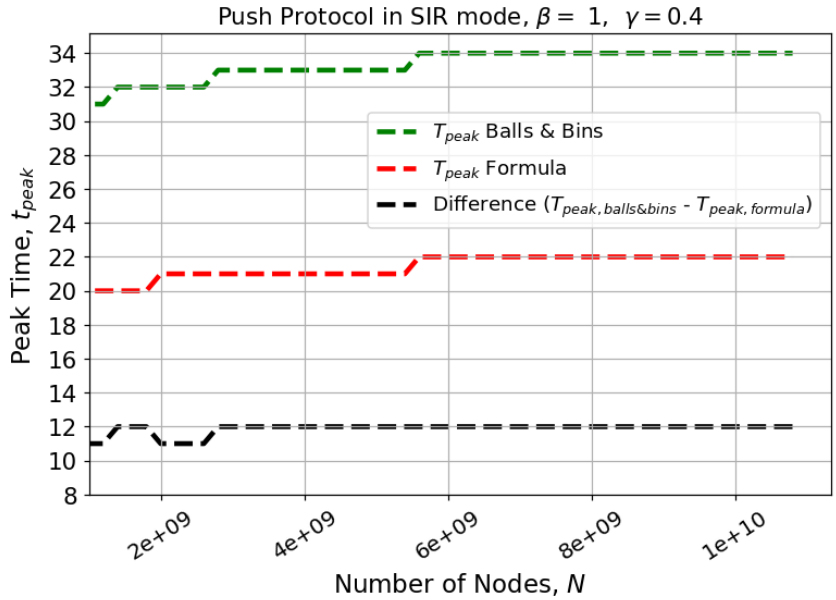

Figure 4.17. The comparison of T_{peak} values obtained from equation (3.18) with values obtained from equation (3.20) for 1 billion to 10 billion nodes. For the case of β = 1

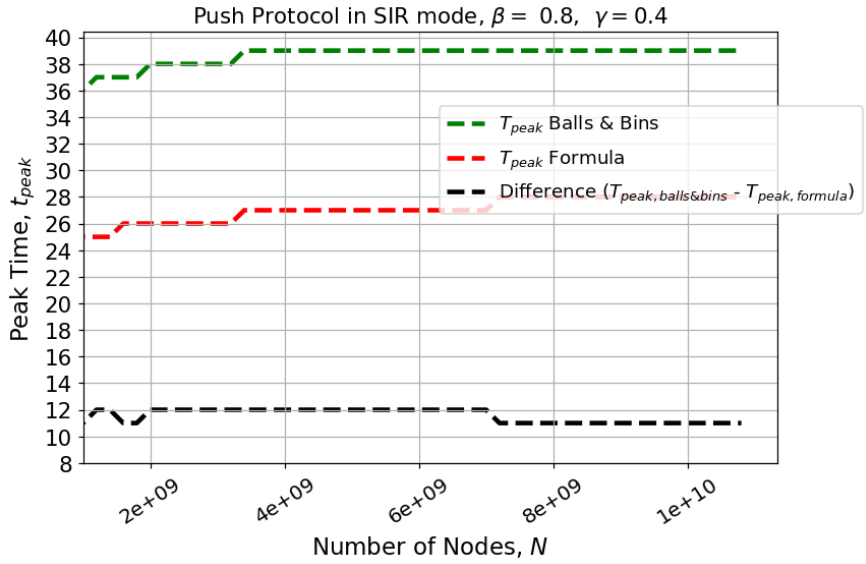

Figure 4.18. The comparison of T_{peak} values obtained from our approximate T_peak formula given in equation (3.18) with values obtained from equation (3.20) for 1 billion to 10 billion nodes. For the case of β = 0.8

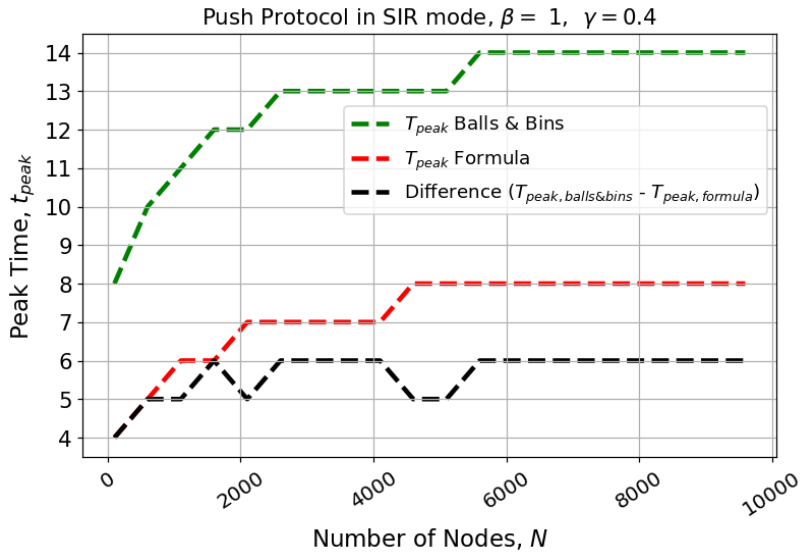

Figure 4.19. The comparison of T_{peak} values obtained from equation (3.18) with values obtained from equation (3.20) for 1000 to 10000 nodes. For the case of β = 1

Until now, we have studied push protocol in SI and SIR mode in single update scenario where a single message is being disseminated throughout the network. One important research question hasn't been answered in literature yet is how protocol behaves when multiple updates injected successively. In the following experiment, we simulated push protocol in SIR mode with multiple successive message each has been injected to the system at time t_i, which denotes injection time. Δt_i is the constant time between two successive updates.

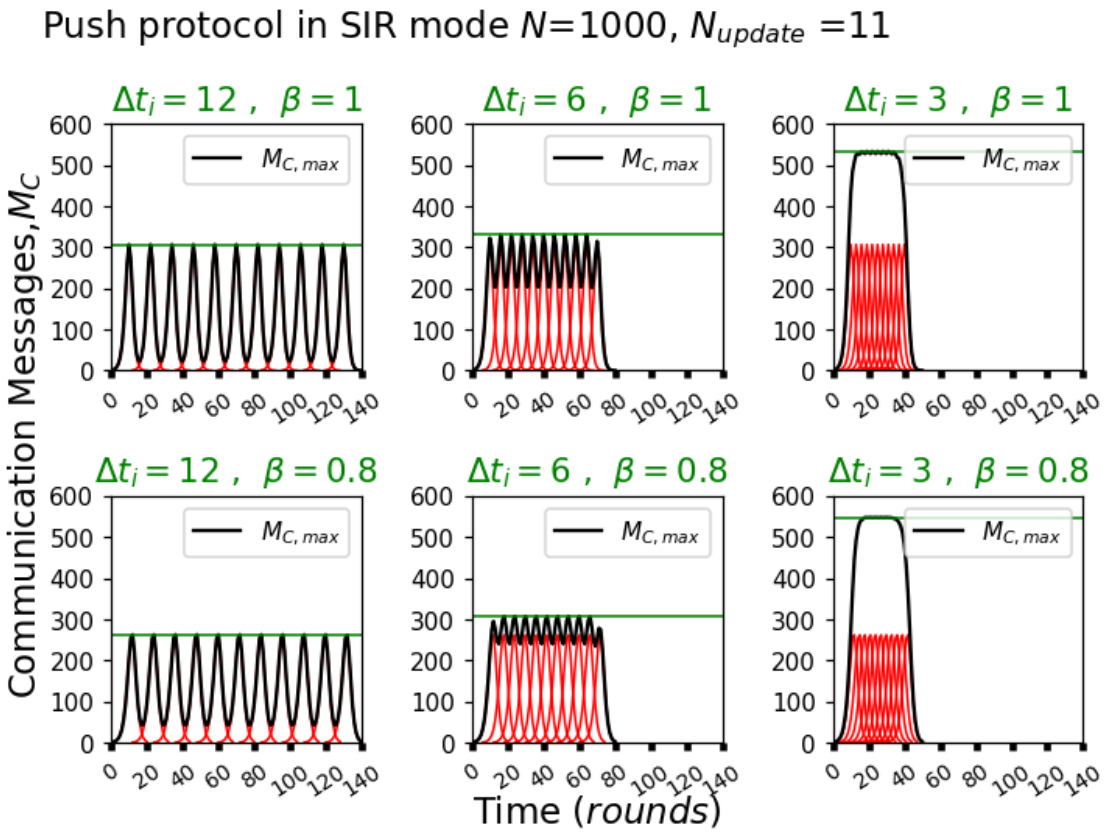

Figure 4.20. The comparison of communication message load on the network for different Δt_i and β values

Results show that if two successive updates injected with enough delay, maximum load on the network will be equal to the single update case, but dissemination will take much longer. As shown in Figure 4.20 when delay between two successive updates decreases because of overlapping maximum load on the network increases. However, this increase strongly depends on the amount of overlap between peaks. Decreasing Δt_i from 12 to 6 time required to disseminate 11 updates decreased from 140 rounds to 80 rounds with only slightly increase in maximum network load. But decreasing Δt_i from 6 rounds to 3 rounds almost doubles maximum amount of network load. If N updates injected into system with $\Delta t_i = 0$, maximum network load will increase N times because of overlapping of peaks. Figure 4.21 illustrates this fact more clearly. Thus, selection of Δt_i have to be done according to available network capacity and desired propagation time. As shown in Figure 4.21 Δt_i values greater than T_{peak} doesn't decrease network load.

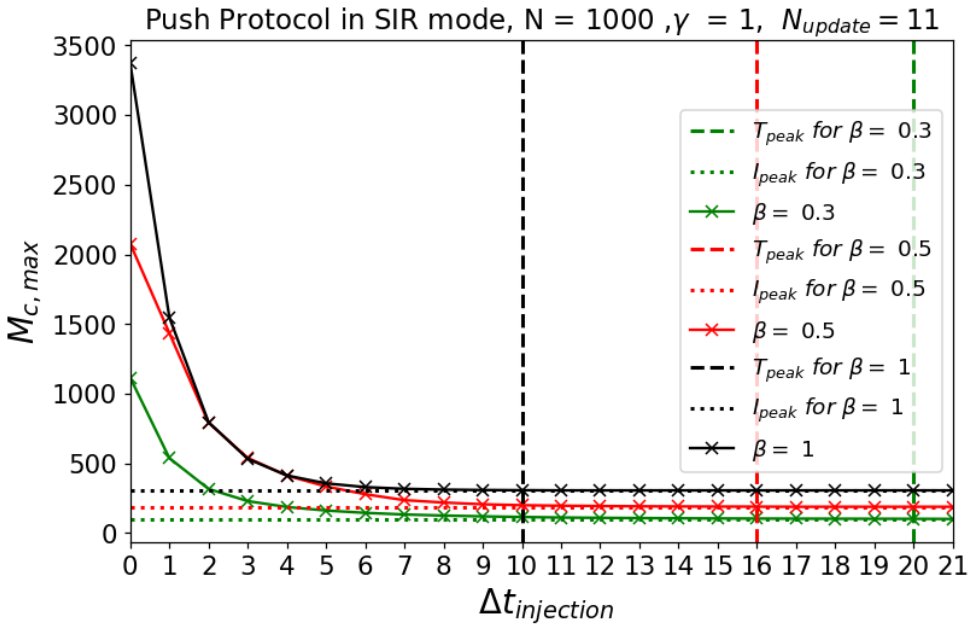

Figure 4.21. The comparison of maximum communication message load on the network for different Δt_i and β values

4.1.3 Pull protocol experiments

In this experiment, we have implemented pull protocol algorithm given in Figure 3.2. For different node sizes, experiments performed and results compared to equation (3.22). It is observed simulation results deviates from equation at the end of dissemination process. And full dissemination time delays by a few rounds.

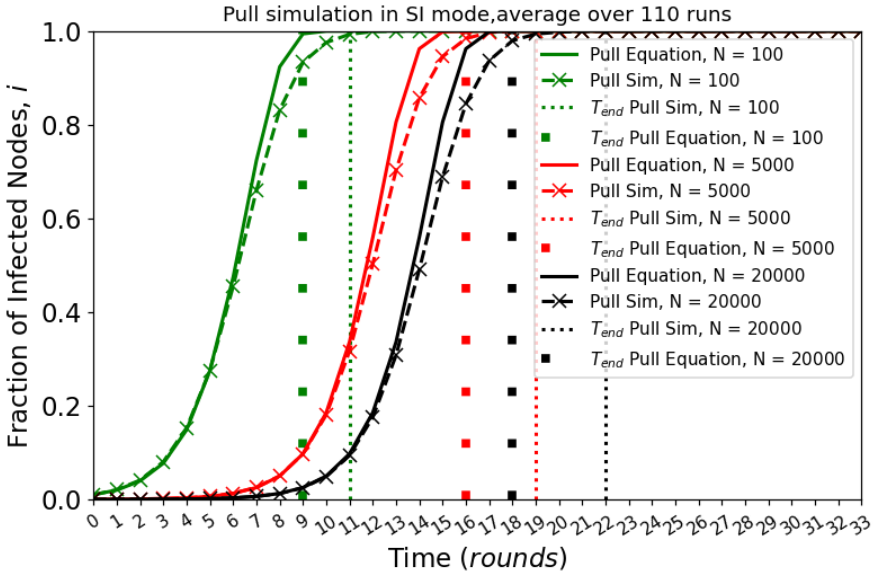

Figure 4.22. The comparison of agent based pull protocol with equation (3.22)

4.1.4 Push-pull protocol experiments in SI model

In this experiment, we have implemented pull protocol algorithm given in Figure 3.3. For different node sizes, experiments performed and results compared to equation (3.23). Results show that the discrete model given by equation (3.23) approximates synchronous push-pull protocol quite well.

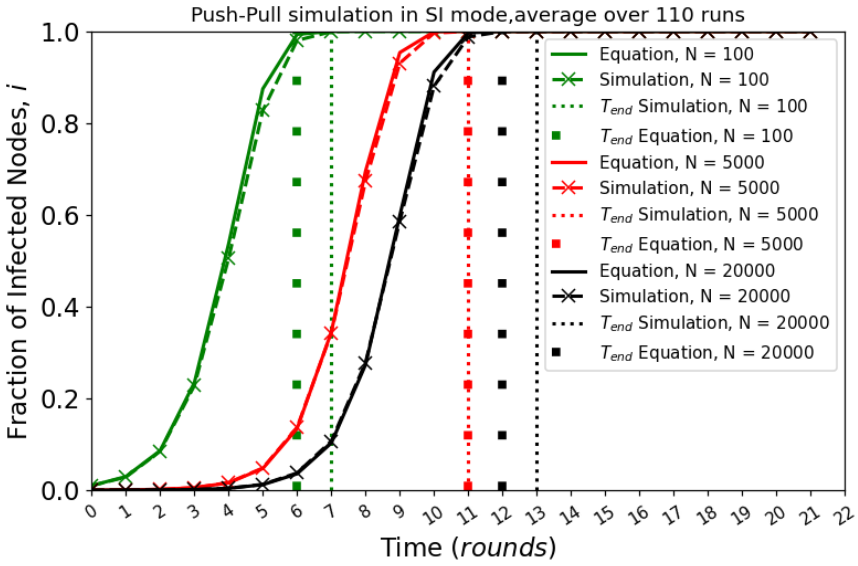

Figure 4.23. The comparison of agent based push-pull protocol in SI mode with equation (3.23)

Next, we compared our differential equation-based model given by equation (3.25) with discrete push-pull equation. Experiment repeated for small and large node sizes. As can be seen in Figure 4.24 and Figure 4.25 the same shift problem is being observed. But interestingly, shift between T_{end} values seem to be constant.

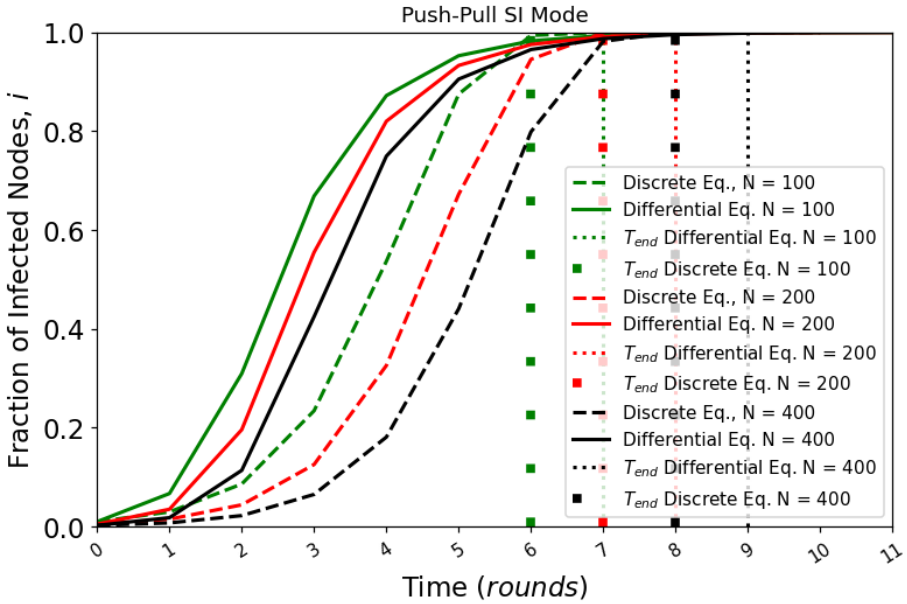

Figure 4.24. The comparison of our differential equation given by equation (3.25) model for push-pull protocol with the discrete push-pull equation for 100,200 and 400 nodes

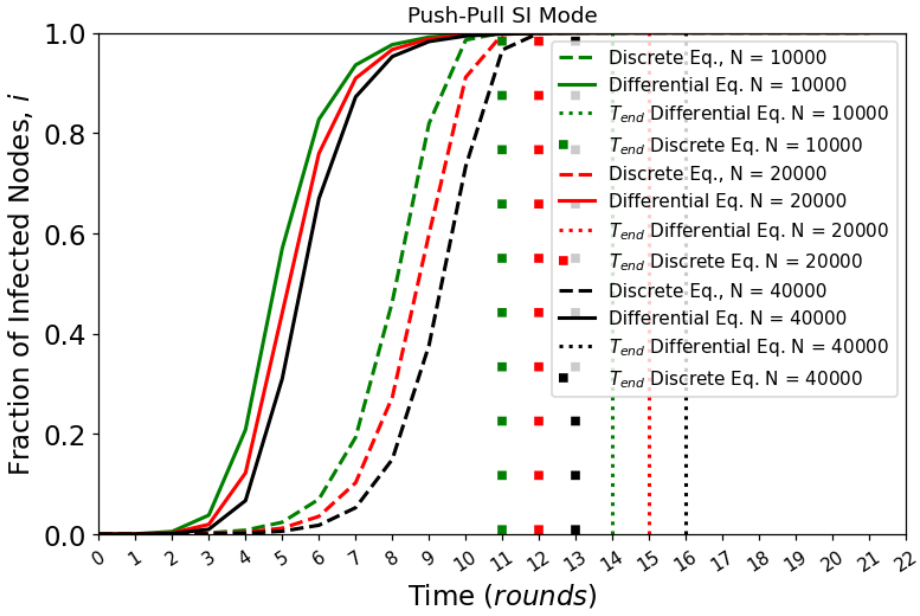

Figure 4.25. The comparison of our differential equation model for push-pull protocol with discrete push-pull equation for 10000, 20000 and 40000 nodes

In the next experiment, we have measured the difference between T_{end} values obtained from differential equation model and discrete equation model. Surprisingly the difference was constant for node sizes of 10000 to 200000. Thus, our differential equation based model can be used instantly predict T_{end} by taking into account the delay for nodes sizes within this range (Figure 4.25). Experiment was repeated for wide range of node sizes and results obtained show that growth of difference is order of $o(logn)$(Figure 4.26).

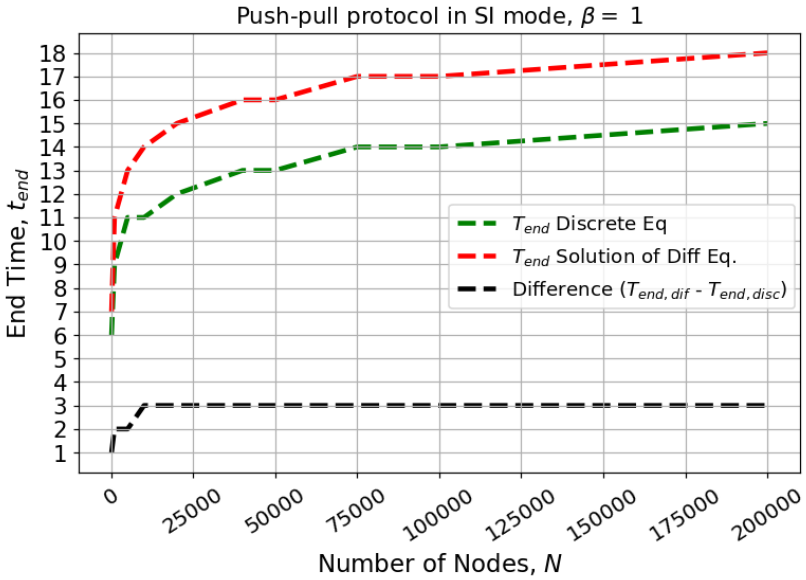

Figure 4.26. The comparison of T_{end} values obtained from differential equation based push-pull model and discrete push-pull equation

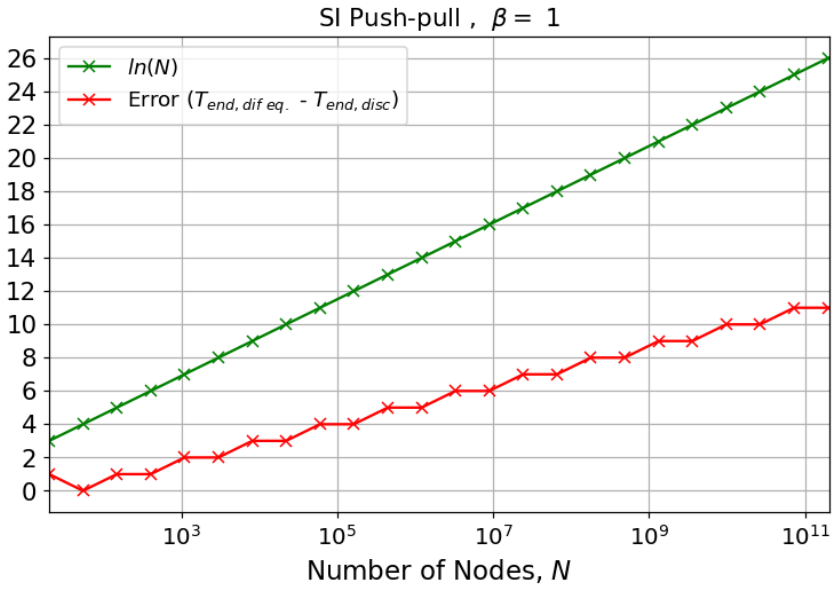

Figure 4.27. The growth of difference between T_{end} values obtained from differential equation based push-pull model and discrete push-pull equation given by equation (3.23)

In the following experiment, it is observed that by delayed differential equation based model with time difference measured in previous experiment quite well approximates discrete push-pull equation as shown in Figure 4.27.

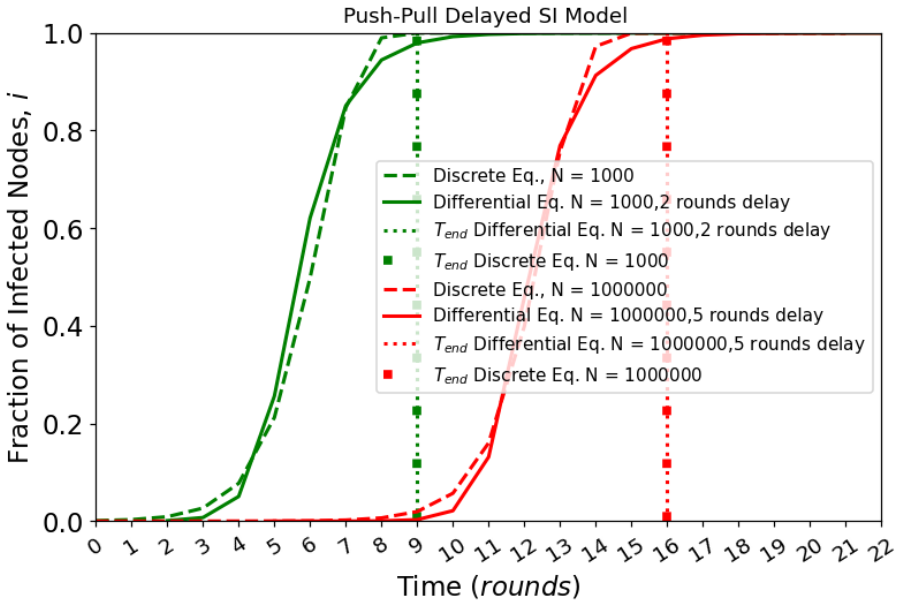

Figure 4.28. The comparison of differential equation based delayed SI model given by equation (3.26) with discrete push-pull equation

4.1.5 Push-pull protocol experiments in SIR model

In this section, we compared numerical solution of our compartmental push-pull model with agent-based push-pull protocol in SIR mode. Numerical solution obtained using Runge-Kutta 4th order method. With applying delay correction values obtained

from Figure 4.25 and Figure 4.26 we observed that the delayed compartmental model can be used to approximate push-pull protocol in SIR model.

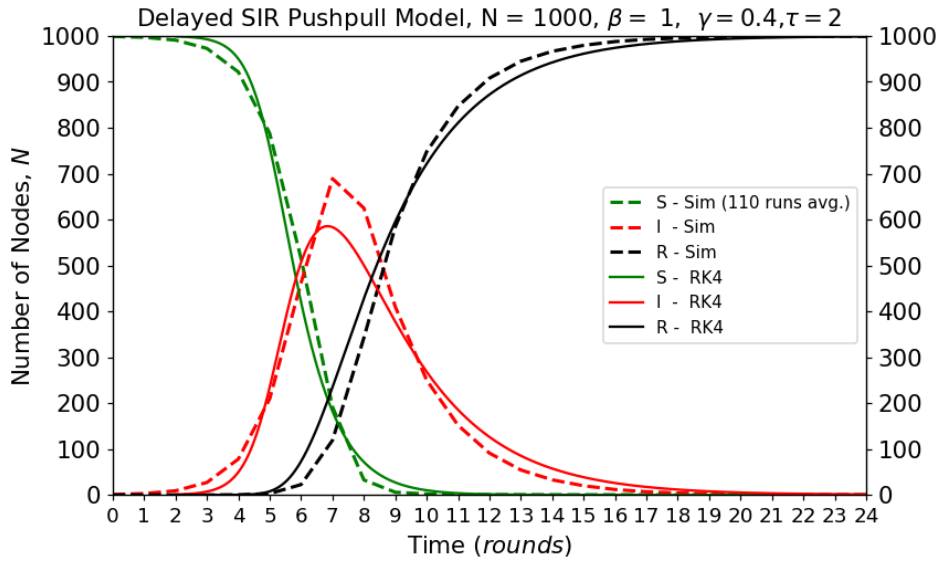

Figure 4.29. The comparison of delayed compartmental push-pull model with agent-based push-pull simulation in SIR model for N=1000. τ denotes amount of delay in terms of rounds

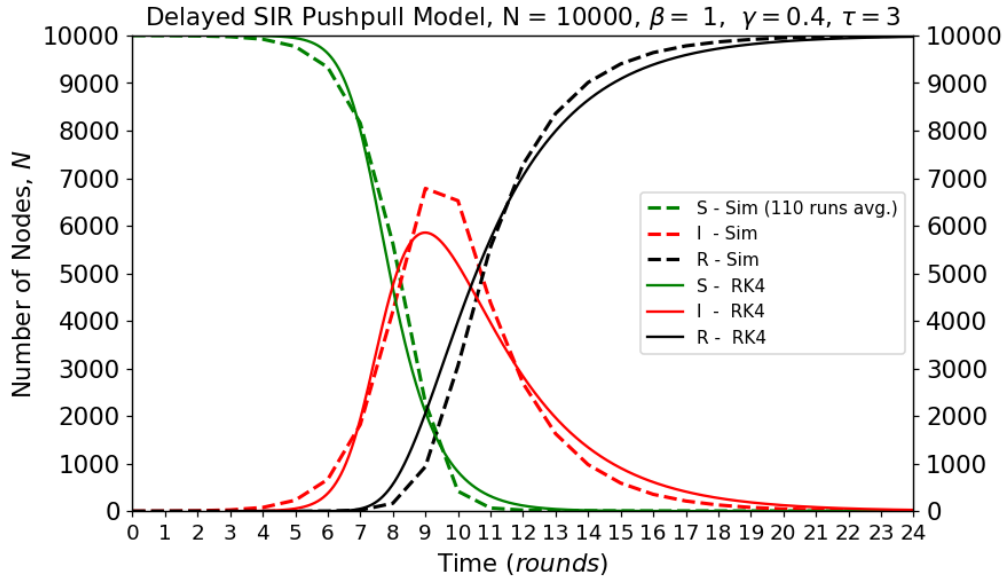

Figure 4.30. The comparison of delayed compartmental push-pull model with agent-based push-pull simulation in SIR model for N=10000. τ denotes amount of delay in terms of rounds

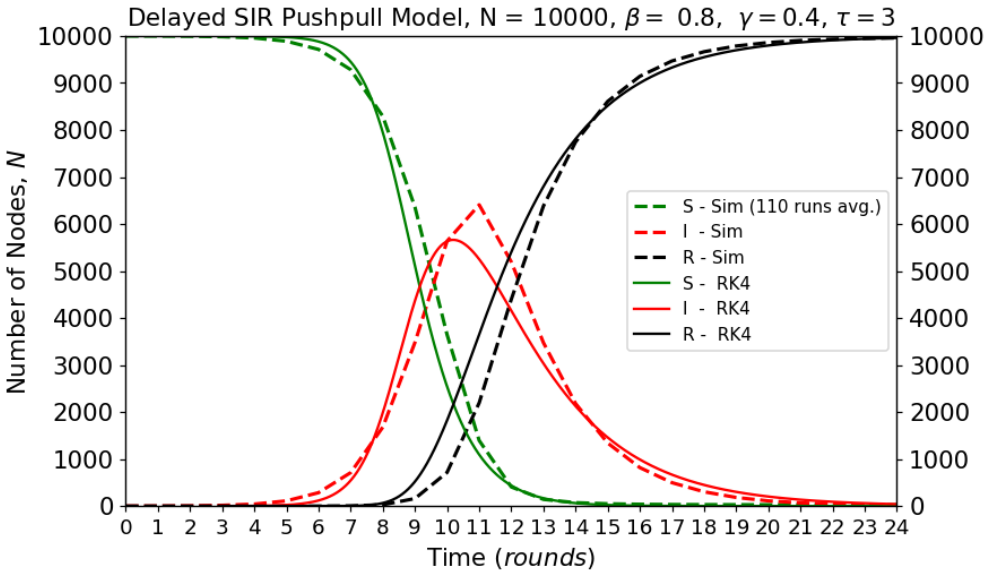

Figure 4.31. The comparison of delayed compartmental push-pull model with agent-based push-pull simulation in SIR model in the case where payload messages fails with %20 probability

4.1.6 The comparison of push, pull and push-pull protocols in SI mode

In this section, we have compared all three gossip protocols in susceptible-infected (SI) model. As shown in Figure 4.31 it is observed that push-pull protocol is fastest. This is expected since all the nodes are active. Pull protocol is slightly faster than push but deviation from average is quite high. Thus, behavior of pull protocol is less predictable. Variance in T_{end} time for push-pull protocol is quite low compared to others. Thus, complete dissemination time can be predicted with higher accuracy, on the other hand variance in T_{end} for push and pull protocols is quite high.

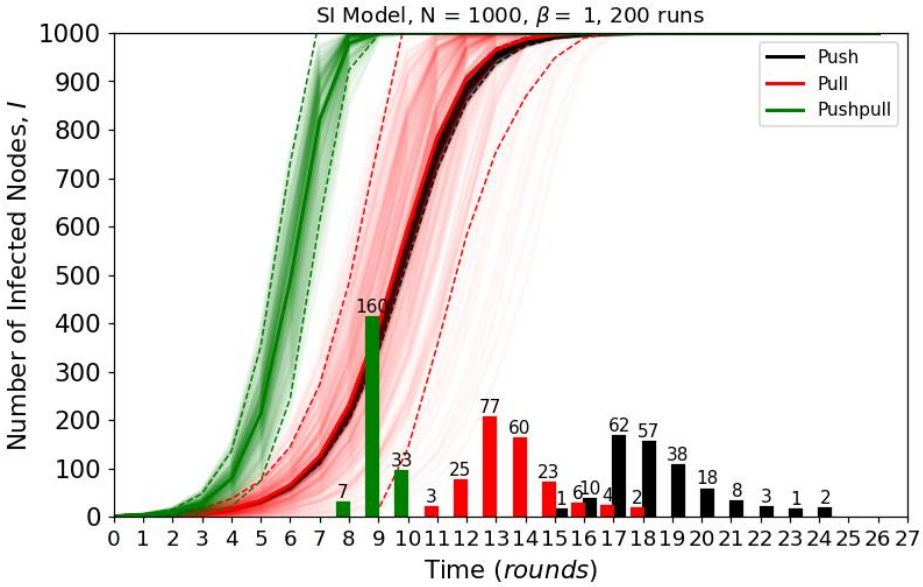

Figure 4.32. The comparison of push, pull and push-pull protocols in SI model

In the next experiment, we have compared push, pull and push-pull protocols in terms of number of communication messages sent in each round. Number of communication messages, M_C, sent in push-pull protocol are constant and equals to node size. Push and pull protocols use approximately the same number of communication messages. In push protocol, number of communication messages increases with time. On the other hand, M_C decreases in pull case since number of susceptible nodes decreases. (Figure 4.32)

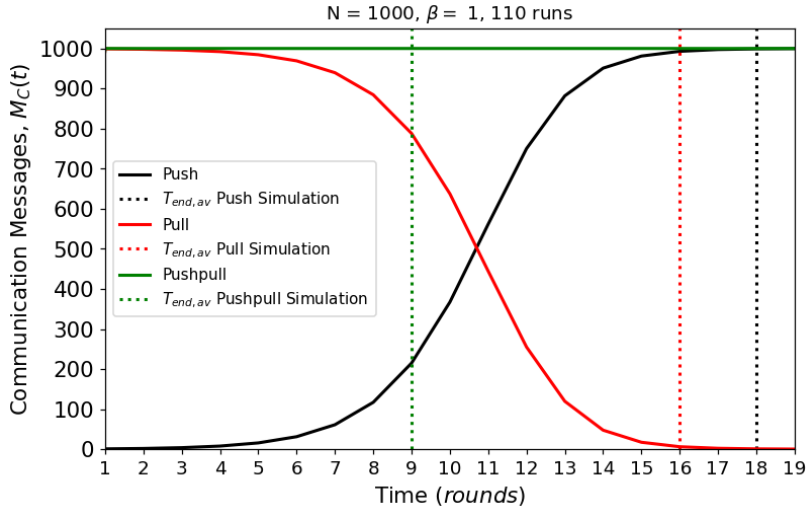

Figure 4.33. Number of communication messages sent in push, pull and push-pull protocols with respect to time

In applications where payload message size is much bigger than communication message size, it is important to compare network load of push, pull and push-pull protocols in terms of payload messages. In this study, we assume that in push and push-pull protocols, infected nodes transfer payload messages only if the randomly chosen node replies with a request for the payload message. As shown in Figure 4.33 push-pull needs more network capacity. Push and pull protocols show the same behavior. Area under the curves is the same for all three protocols and equals to N. Since a node receives a payload message only once.

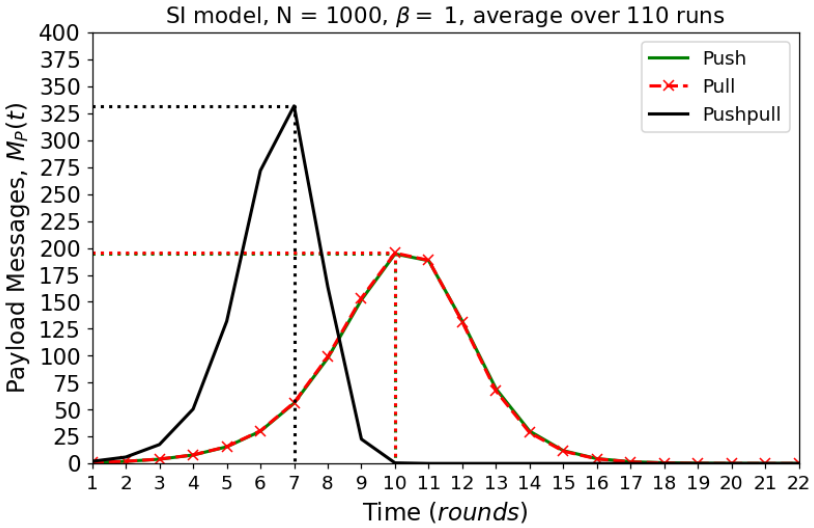

Figure 4.34. The comparison of push, pull and push-pull protocols in terms of payload messages

4.1.7 The comparison of push and push-pull protocols in SIR model

In this section, we have compared push and push-pull protocols in SIR model. It is observed that push-pull is faster and all nodes finally receives the update. On the other hand, push protocol is slower and some fraction of nodes remains susceptible at the end of the process. (Figure 4.34) Despite these important advantages, communication overhead of push-pull protocol is higher.

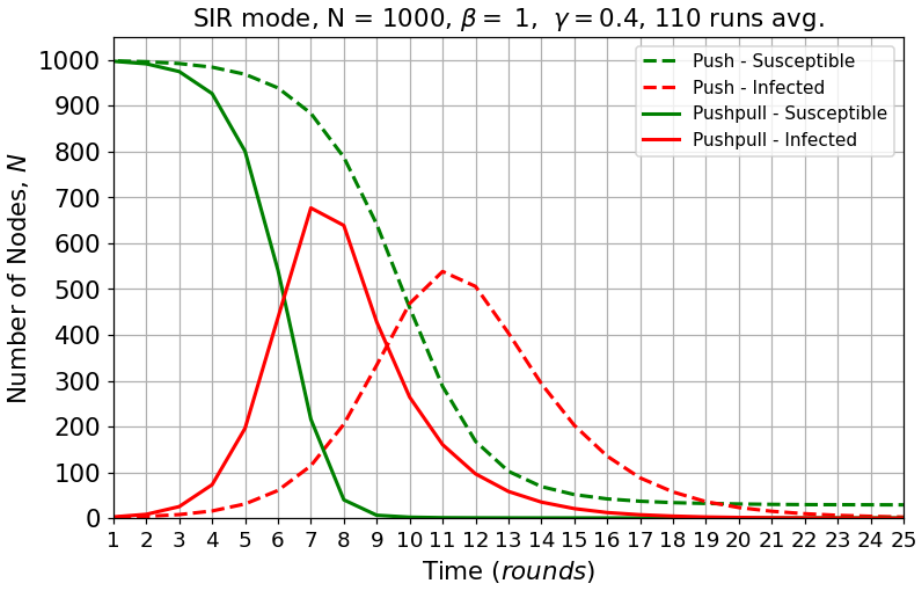

Figure 4.35. The comparison of push and push-pull protocol in SIR mode using agent based simulation

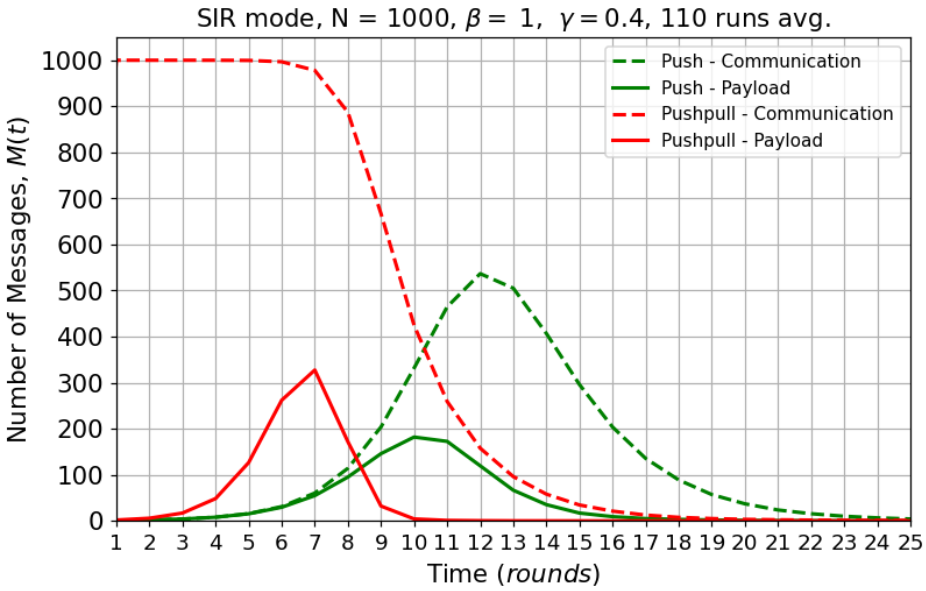

Figure 4.36. The comparison of communication overhead of push and push-pull protocol in SIR mode using agent based simulation

In the final experiment of this section, we compared push and push-pull protocols in multiple update case. Updates were injected to the system at time t_i, which denotes injection time. Δt_i is the constant time difference between two successive updates. Thus, we assumed updates are injected to the system periodically, not at random. Push-pull protocol requires more than 3 times higher bandwidth amount. However, dissemination process is faster than push case. Figure 4.36 shows comparison of push and push-pull protocols with two β values and 3 different injection periods. Figure 4.36 shows comparison considering only payload message case. Results show that if size of payload message is much higher than size of communication message, it is better to use push-pull protocol which disseminates update much faster. Otherwise, it is better to use push protocol.

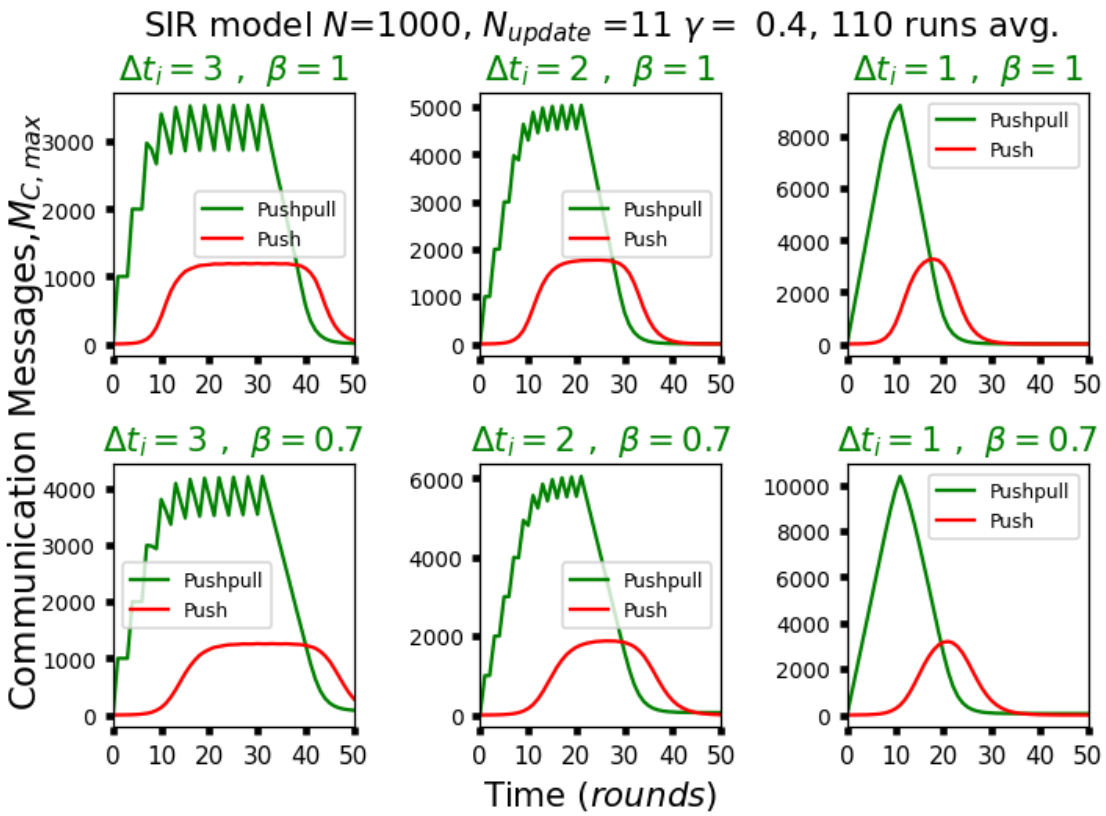

Figure 4.37. The comparison of communication overhead of push and push-pull protocols in multiple update case excluding payload messages

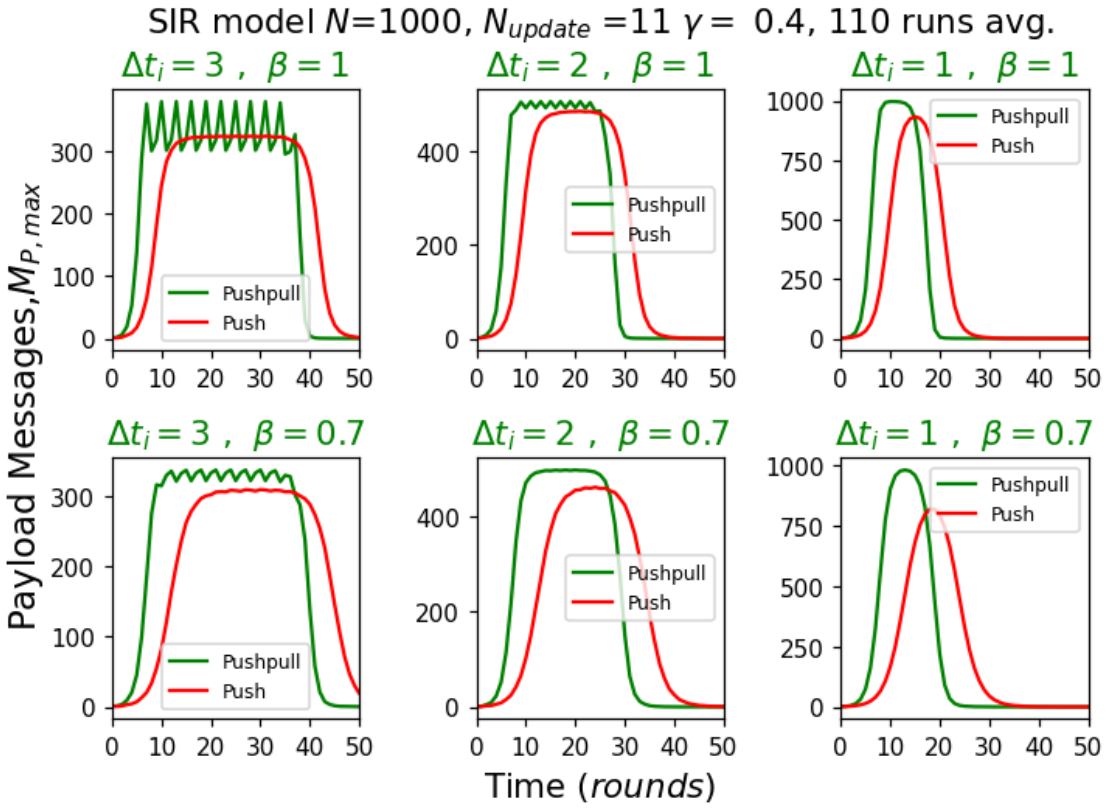

Figure 4.38. The comparison of communication overhead of push and push-pull protocols in multiple update case considering only payload messages

4.2 Gossip Learning Experiments

In the first experiment of this section, gossip learning algorithm given in Figure 3.4 was implemented and compared to partial model transfer and local only training case where nodes do not exchange models. Each node has only 10 data points. Results show that final test accuracy of nodes in local only training varies according to how their local data is close to distribution of test data and their initial weights.

Results show that average final accuracy over all nodes can be improved significantly if nodes collaborate with exchanging models using gossip learning algorithm. (Figure 4.39, Figure 4.40)

Experimental results with partial model transfer (Figure 4.41, Figure 4.42, Figure 4.43) show that model can be transferred by part without loss of final test accuracy but at the expense of longer iterations for convergence, proportional to saved bandwidth or inversely proportional to transferred percentage of model.

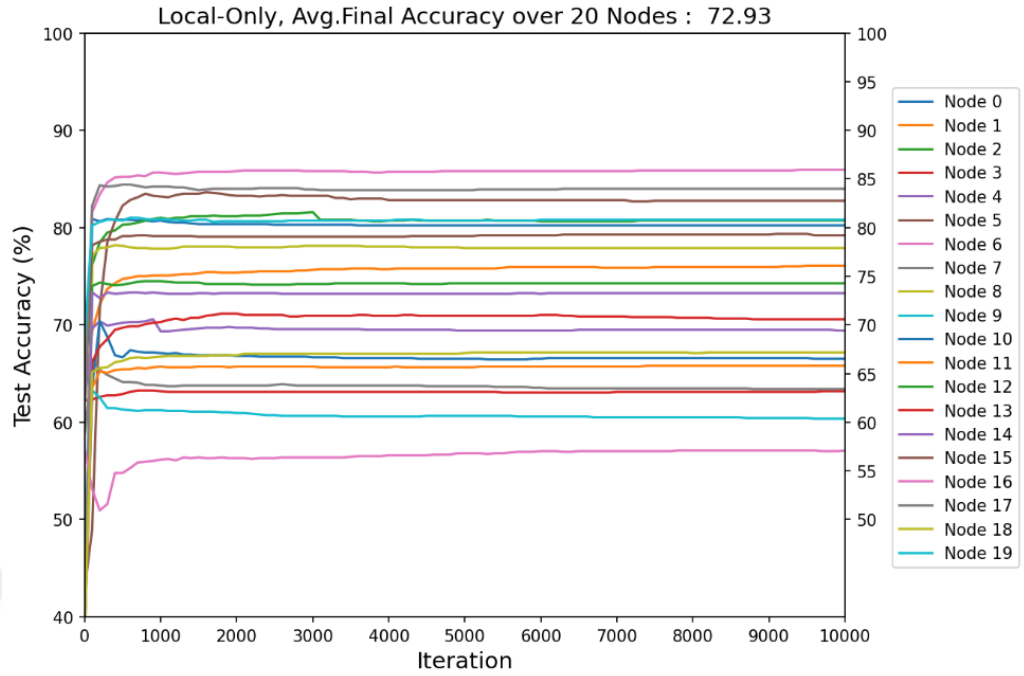

Figure 4.39. Local only training with 20 nodes

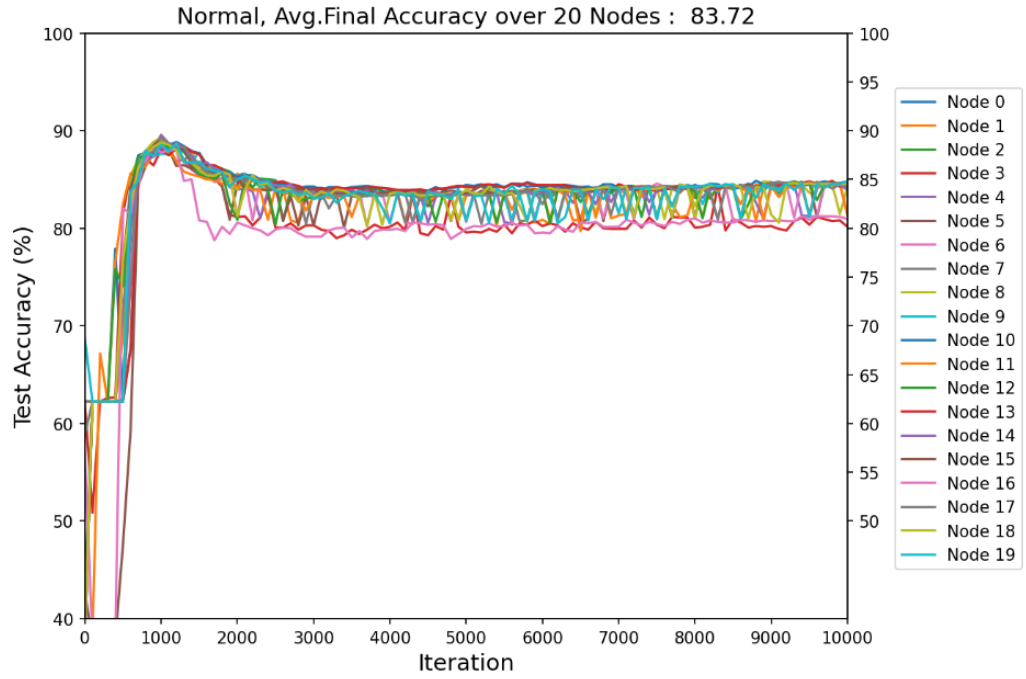

Figure 4.40. Generic gossip learning algorithm with 20 nodes

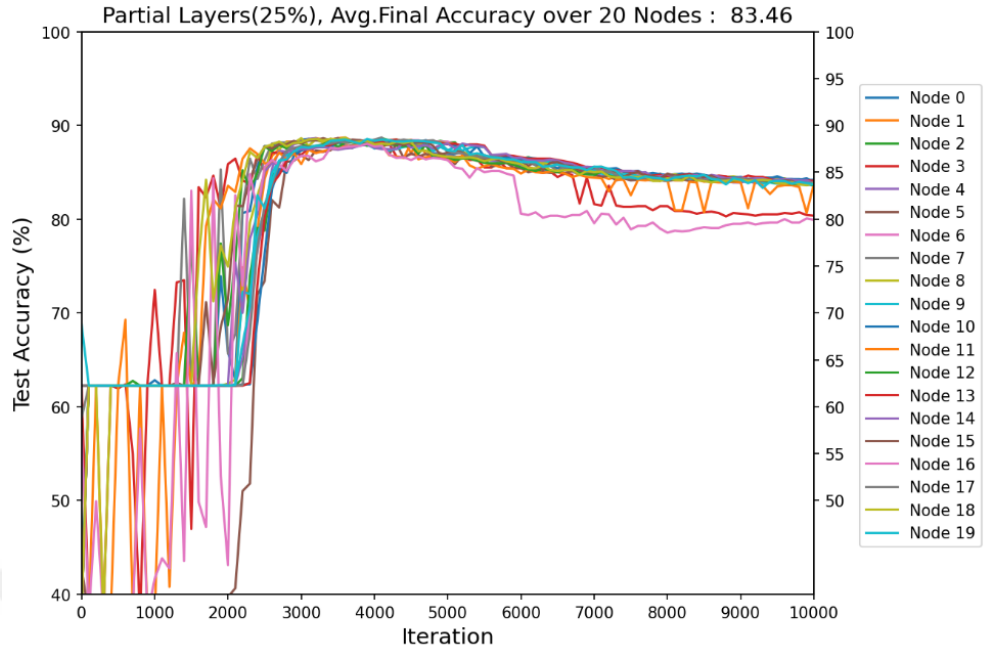

Figure 4.41. Gossip learning algorithm with 25% of random weight transfer with 20 nodes

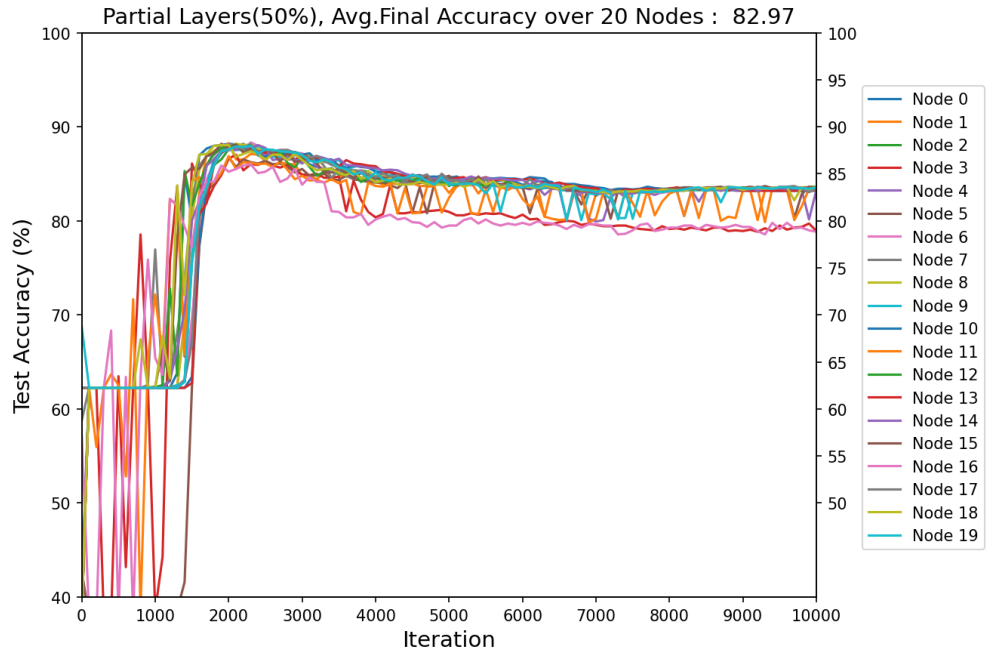

Figure 4.42. Gossip learning algorithm with 50% of random weight transfer with 20 nodes

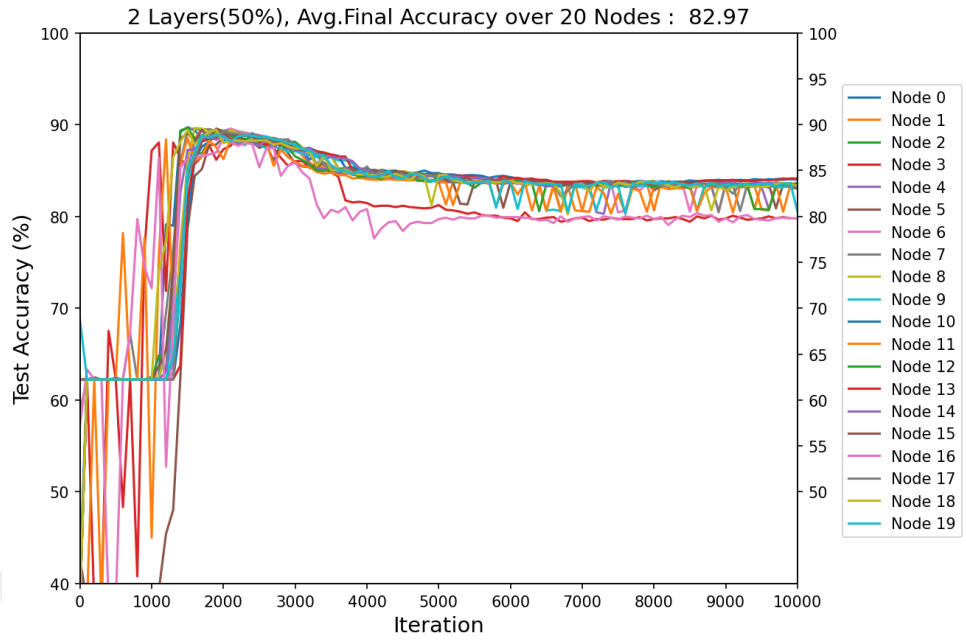

Figure 4.43. Gossip learning algorithm with 2 successive layer tranfer with 20 nodes

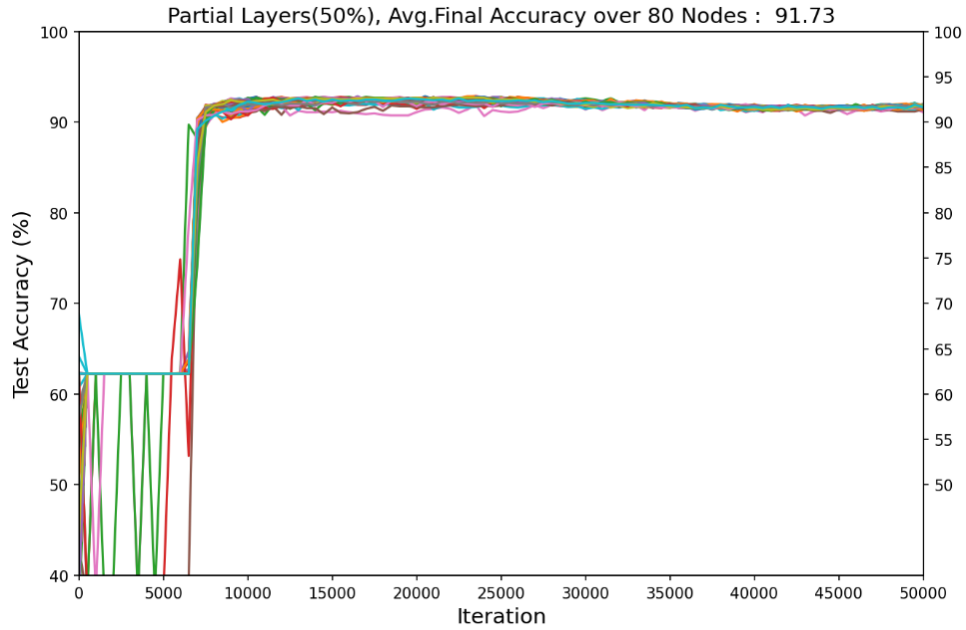

Figure 4.44. Gossip learning algorithm with 50% of random weight transfer with 80 nodes

In order to see effect of node size on the convergence previous experiments repeated for 40 and 80 nodes. Only average results are presented here. In addition, we

have also included single node case where a single centralized node was trained with all data available in other nodes.

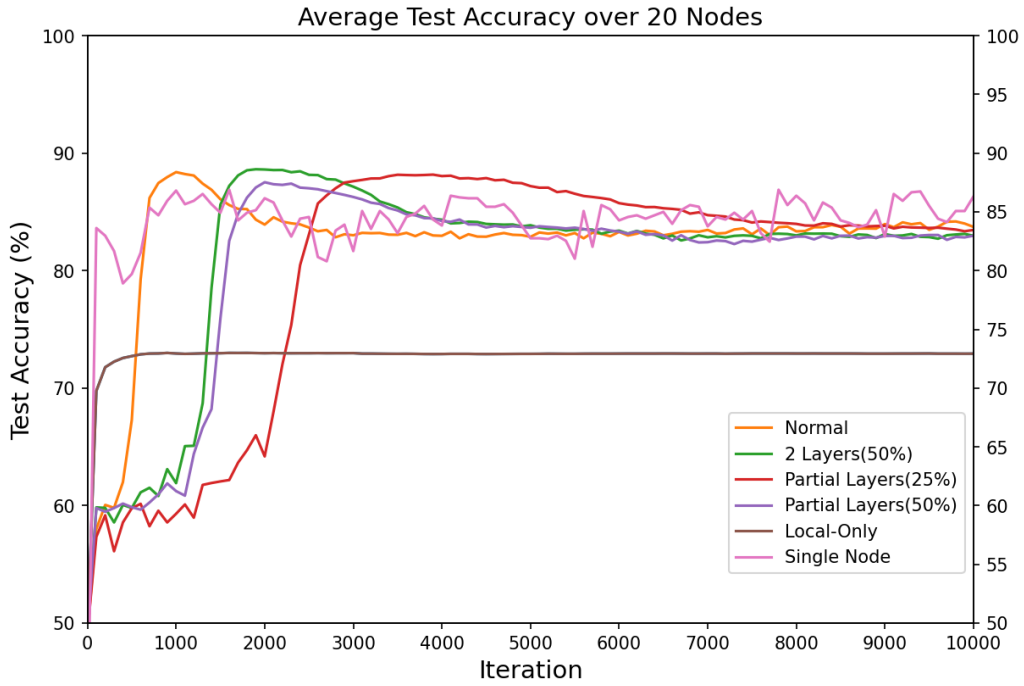

Figure 4.45. The comparison of generic gossip learning algorithm with partial model transfer, centralized and local training for 20 nodes

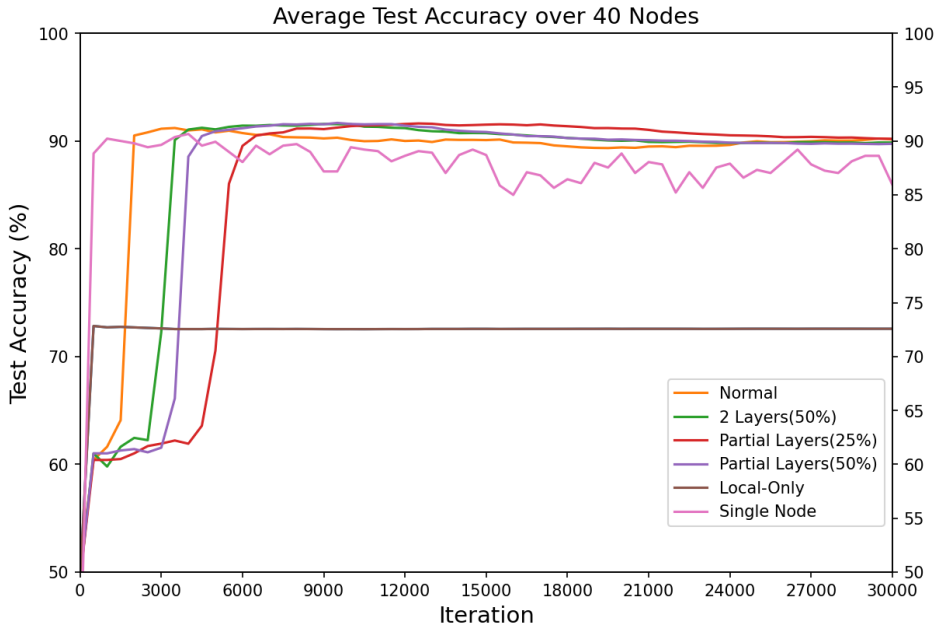

Figure 4.46. The comparison of generic gossip learning algorithm with partial model transfer, centralized and local training for 40 nodes

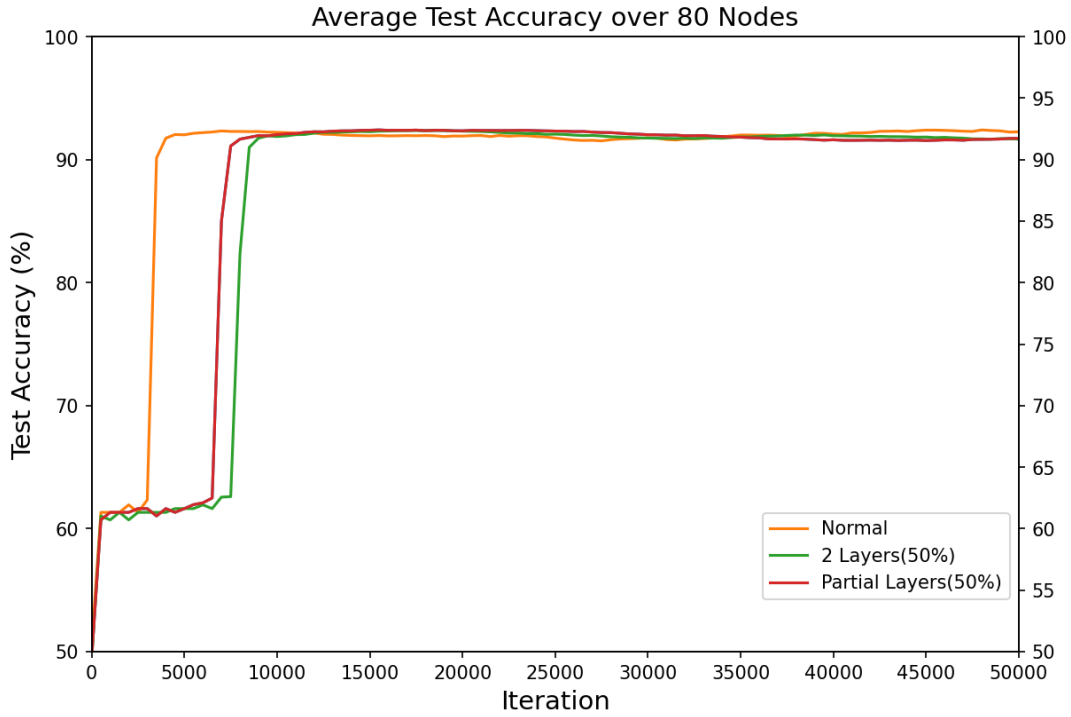

Figure 4.47. The comparison of generic gossip learning algorithm with partial model transfer for 80 nodes

Results presented in above show that doubling network size approximately doubles required iterations for convergence. And larger networks produce better training results with almost no over fitting and stable accuracy (Figure 4.42, Figure 4.44). For 50000 iterations it is observed that accuracy remains almost same as can be seen in Figure 4.44. But for networks with small number of nodes variance of test accuracy is high and over fitting can be observed as neural network over trained. This can be expected since more nodes means more data which improves accuracy. However, gossip learning algorithm with both full model and partial model transfer performs better than centralized training with equal amount of data.

5. CONCLUSION

In this study, we have studied fully-decentralized deep learning and its underlying gossip protocol. We derived an exact solution of SIR model for push protocol and an approximate peak equation that can be used to find the time when number of infected nodes reach its maximum value.

We have implemented and verified an agent-based simulation framework which can be used to simulate push, pull and push-pull protocols, obtain exact and numerical solutions to both gossip and epidemic models in various settings.

We have shown that differential equation-based models used in epidemiology are not appropriate to describe synchronous gossip protocols in their original form. We address this problem by introducing a delay factor. We observed that delayed SI and SIR models match quite well to synchronous gossip protocols.

And we have derived a recurrence relation for push protocol in SIR model which is perfectly matches simulation results. In addition, we have derived an approximate analytic solution for push-pull protocol in SI model. We have derived an approximate analytic model for push-pull protocol in SIR mode and compared its numerical solution with simulation results.

We have trained neural network using push-based gossip learning algorithm, and we have shown that average accuracy of individual nodes increases when they exchange models. We have simulated gossip learning protocol with partial model transfer where nodes exchange only a small fraction of weights. For bandwidth limited networks, models can be transferred partially without loss of accuracy but at expense of number of rounds proportional to saved bandwidth amount.

It is observed that halving transferred weights approximately doubles the required number of rounds for convergence. Increasing network size prevents over fitting and gives better results compared to a single centralized node with the same amount of data.

It is also observed that doubling network size doubles the required number of rounds for convergence. And increasing network size prevents over-fitting and gives better results compared to single centralized node with same amount of data.

6. REFERENCES

Agarwal, A., Chapelle, O., Dudík, M., and Langford, J. 2014. A reliable effective terascale linear learning system. *The Journal of Machine Learning Research, 15(1):* 1111-1133.

Alkathiri, A.A., Giaretta, L., Girdzijauskas, S., and Sahlgren, M. 2021. Decentralized Word2Vec Using Gossip Learning. In *23rd Nordic Conference on Computational Linguistics (NoDaLiDa 2021),* 1-2 June, Online

Anonymous 1: Spambase Data Set https://archive.ics.uci.edu/ml/datasets/spambase [Last access date: 27.05.2022].

Bagoly, S.M., and Dănescu, R.G. 2020. Round Based Extension Algorithm for Gossip Learning. In *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP),* pp. 251-257, 3-5 September, Cluj-Napoca, Romania

Bailey, N. T. 1975. The mathematical theory of infectious diseases and its applications. In The mathematical theory of infectious diseases and its applications, Charles Griffin, London, 35 p.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., and Ng, A. 2012. Large scale distributed deep networks. *Advances in neural information processing systems,* pp. 1232-1240, 3-6 December, Nevada, USA

Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., and Terry, D. 1987. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*: 1-12.

Frieze, A. M., and Grimmett, G. R. 1985. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics,* 10(1): 57-77.

Giaretta, L., and Girdzijauskas, Š. 2019. Gossip learning: Off the beaten path. In *2019 IEEE International Conference on Big Data,* pp. 1117-1124. 9-12 December, Los Angeles, USA

Harko, T., Lobo, F.S., and Mak, M. 2014. Exact analytical solutions of the Susceptible-Infected-Recovered (SIR) epidemic model and of the SIR model with equal death and birth rates. *Applied Mathematics and Computation,* 236, 184-194.

Hegedűs, I., Danner, G., and Jelasity, M. 2019. Gossip learning as a decentralized alternative to federated learning. In *IFIP International Conference on Distributed Applications and Interoperable Systems,* pp. 74-90, 17-21 June, Kongens Lyngby, Denmark

Johansen, H.D., Renesse, R.V., Vigfusson, Y., and Johansen, D. 2015. Fireflies: A secure and scalable membership and gossip service. *ACM Transactions on Computer Systems (TOCS),* 33(2), 1-32.

Karp, R., Schindelhauer, C., Shenker, S., and Vocking, B. 2000. Randomized rumor spreading. In *Proceedings 41st Annual Symposium on Foundations of Computer Science* pp. 565-574, 12-14 November, CA, USA

Kermack, W.O., and McKendrick, A.G. 1927. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing*

papers of a mathematical and physical character, 115(772), pp. 700-721, 1 December, London, United Kingdom

Ormándi, R., Hegedűs, I., and Jelasity, M. 2013. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4): 556-571.

Pittel, B. 1987. On spreading a rumor. SIAM Journal on Applied Mathematics, 47(1): 213-223.

Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., and Rellermeyer, J.S. 2020. A survey on distributed machine learning. *ACM Computing Surveys (CSUR)*, 53(2): 1-33.

Voulgaris, S., Jelasity, M., and Steen, M. V. 2003. A robust and scalable peer-to-peer gossiping protocol. In *International Workshop on Agents and P2P Computing*, pp. 47-58, 14 July, Springer, Berlin, Heidelberg.

Wei, J., Dai, W., Qiao, A., Ho, Q., Cui, H., Ganger, G.R., and Xing, E. P. 2015. Managed communication and consistency for fast data-parallel iterative analytics. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pp. 381-394, 27-29 August, Kohala Coast, Hawaii

Zecchin, M., Gesbert, D., and Kountouris, M. 2022. UAV-Aided Decentralized Learning over Mesh Networks. arXiv preprint arXiv:2203.01008.

7. APPENDIX

7.1. Exact Solution to Push SIR Model

Compartmental model of gossip is given in equations (7.1, 7.2, 7.3) with initial conditions in equation (7.4)

$$\frac{ds}{dt} = -\beta si \tag{7.1}$$

$$\frac{di}{dt} = \beta si - \gamma(1-s)i \tag{7.2}$$

$$\frac{dr}{dt} = \gamma(1-s)i \tag{7.3}$$

$$s_0 = 1 - \frac{1}{N}, i_0 = \frac{1}{N}, r_0 = 0 \tag{7.4}$$

s_0, i_0 and r_0 are initial values at time t_0. By differentiating both side of equation (7.1) we get

$$\frac{d^2s}{dt^2} = -\beta \frac{ds}{dt} i - \beta s \frac{di}{dt} \tag{7.5}$$

Solving equation (7.1) for i we get

$$i = -\frac{1}{\beta s} i \frac{ds}{dt} \tag{7.6}$$

Inserting equation (7.6) into equation (7.5) we get

$$\frac{d^2s}{dt^2} = -\beta \frac{ds}{dt}\left(\frac{1}{\beta s} i \frac{ds}{dt}\right) - \beta s \frac{di}{dt} \tag{7.7}$$

And inserting equation (7.6) into equation (7.2) we get

$$\frac{di}{dt} = -\frac{ds}{dt} + \gamma(1-s)\frac{1}{\beta s}\frac{ds}{dt} \tag{7.8}$$

And inserting equation (7.8) into equation (7.7) we get

$$\frac{d^2s}{dt^2} = \frac{1}{s}\left(\frac{ds}{dt}\right)^2 + \beta s \frac{ds}{dt} - \gamma \frac{ds}{dt} + \gamma s \frac{ds}{dt} \tag{7.9}$$

Equation (7.9) is special non-linear second order differential equation with missing independent variable t which can be reduced to first order as shown in following steps. Let

$$\frac{ds}{dt} = M \tag{7.10}$$

Then left-hand side of equation (7.9) can be rewritten using chain rule as

$$\frac{d^2s}{dt^2} = \frac{dM}{dt} = \frac{dM}{ds}\frac{ds}{dt} \tag{7.11}$$

Substituting equation (7.10) into equation (7.11) we get

$$\frac{d^2s}{dt^2} = M\frac{dM}{ds} \tag{7.12}$$

Then inserting equations (7.10, 7.11) into equation (7.9) we reduce second order differential equation into a first order non-homogeneous differential equation which can be solved using integration factor.

$$\frac{dM}{ds} - \frac{M}{s} = (\beta + \gamma)s - \gamma \tag{7.13}$$

Multiplying both side of equation (7.13) with its integration factor $1/s$ we get

$$\frac{1}{s}\frac{dM}{ds} - \frac{M}{s^2} = (\beta + \gamma) - \frac{\gamma}{s} \tag{7.14}$$

And equation (7.14) can be rewritten as

$$\frac{d}{ds}\left(\frac{M}{s}\right) = (\beta + \gamma) - \frac{\gamma}{s} \tag{7.15}$$

Integration of equation (7.15) results

$$\frac{M}{s} = \beta s + \gamma s - \gamma \ln(s) + c \tag{7.16}$$

Finally using equation (7.10) equation (7.16) can be rewritten as

$$\frac{ds}{dt} = (\beta + \gamma)s^2 - \gamma\, \mathrm{sln}(s) + cs \tag{7.17}$$

Inserting equation (7.17) into equation (7.1) and using initial conditions given in equation (7.4) then solving for c we get

$$c = \gamma \ln(s_0) - (\beta + \gamma)s_0 - \beta i_0 \tag{7.18}$$

And inserting equation (7.18) into equation (7.17) and rearranging terms we get

$$\frac{ds}{dt} = -s\beta\left(-\left(1 + \frac{\gamma}{\beta}\right)s + \frac{\gamma}{\beta}\ln\left(\frac{s}{s_0}\right) + \left(1 + \frac{\gamma}{\beta}\right)s_0 + i_0\right) \tag{7.19}$$

Integration of equation (7.19) results

$$\int_{s_0}^{s} \frac{ds}{-\beta s\left(-\left(1 + \frac{\gamma}{\beta}\right)s + \frac{\gamma}{\beta}\ln\left(\frac{s}{s_0}\right) + \left(1 + \frac{\gamma}{\beta}\right)s_0 + i_0\right)} = t - t_0 \tag{7.20}$$

Comparing right-hand side of equation (7.1) with equation (7.19) we get an equation for i as

$$i = -\left(1 + \frac{\gamma}{\beta}\right)s + \frac{\gamma}{\beta}\ln\left(\frac{s}{s_0}\right) + \left(1 + \frac{\gamma}{\beta}\right)s_0 + i_0 \tag{7.21}$$

And finally fraction of removed nodes can calculated as

$$r = 1 - i - s \tag{7.22}$$

Equations (7.20, 7.21, 7.22) are exact solution to push protocol in SIR model.

7.2. Peak Time Approximation for Push Protocol in SIR Model

The time when number of infected nodes reaches its maximum value can be found using equation (7.20) as

$$t_{peak} = t_0 - \frac{1}{\beta}\int_{s_0}^{s_p} \frac{ds}{s\left(-\left(1+\frac{\gamma}{\beta}\right)s + \frac{\gamma}{\beta}\ln\left(\frac{s}{s_0}\right) + \left(1+\frac{\gamma}{\beta}\right)s_0 + i_0\right)} \tag{7.23}$$

Where s_p is fraction of susceptible nodes when number of infected nodes reaches its maximum value. s_p can be calculated using equation (7.2). At the peak time, left side of equation (7.2) is equals to 0.

$$0 = \beta s_p i_p - \gamma(1 - s_p)i_p \tag{7.24}$$

Solving for s_p we get

$$s_p = \frac{\gamma}{\gamma + \beta} \tag{7.25}$$

Let $z = \gamma/\beta$ to simplify denominator in left-hand side of equation (7.20) as

$$t_{peak} = t_0 - \frac{1}{\beta}\int_{s_0}^{s_p} \frac{ds}{s\left(\left(-(1+z)s + z\ln\left(\frac{s}{s_0}\right) + (1+z)s_0 + i_0\right)\right)} \tag{7.26}$$

And expanding logarithmic term around s_0 using first 2 terms of Taylor series and rearranging we get

$$t_{peak} = t_0 - \frac{1}{\beta}\int_{s_0}^{s_p} \frac{ds}{\left(\frac{z}{s_0} - 1 - z\right)s^2 + (i_0 + (z+1)s_0 - z)s} \tag{7.27}$$

In order to further simplify notation, let

$$M = \left(\frac{z}{s_0} - 1 - z\right) \tag{7.28}$$

And

$$H = (i_0 + (z+1)s_0 - z) \tag{7.29}$$

Then equation (7.27) can be simplified as

$$t_{peak} = t_0 - \frac{1}{\beta}\int_{s_0}^{s_p} \frac{ds}{Ms^2 + Hs} \tag{7.30}$$

Since M and H are not dependent on s integral in equation (7.30) can be evaluated easily using partial fraction technique as

$$t_{peak} = t_0 - \frac{1}{\beta H}(\ln\left(\frac{s_p}{s_0}\right) + \ln\left(\frac{Ms_0 + H}{Ms_p + H}\right) \tag{7.31}$$

Substituting M, H, z and s_p we get final peak time approximation as

$$t_{peak} = t_0 - \frac{1}{\beta\left(i_0 + \left(\frac{\gamma}{\beta} + 1\right)s_0 - \frac{\gamma}{\beta}\right)}(\ln\left(\frac{s_p}{s_0}\right) + \ln\left(\frac{\left(\gamma/\beta s_0 - 1 - \frac{\gamma}{\beta}\right)s_0 + \left(i_0 + \left(\frac{\gamma}{\beta} + 1\right)s_0 - \frac{\gamma}{\beta}\right)}{\left(\gamma/\beta s_0 - 1 - \frac{\gamma}{\beta}\right)s_p + \left(i_0 + \left(\frac{\gamma}{\beta} + 1\right)s_0 - \frac{\gamma}{\beta}\right)}\right) \tag{7.32}$$

7.3. Discrete Equation for Push Protocol in SIR model

By inserting equation (7.21) into equation (3.6) for fanout value of 1 we get

$$s(t+1) = s(t)\left(1 - \frac{1}{N}\right)^{\left(-\left(1+\frac{\gamma}{\beta}\right)s + \frac{\gamma}{\beta}\ln\left(\frac{s}{s_0}\right) + \left(1+\frac{\gamma}{\beta}\right)s_0 + i_0\right)N\beta} \tag{7.33}$$

Which in text we referred as balls and bins based SIR gossip equation.

7.4. Approximate Analytic Solution to Push-pull in SI model

By subtracting $s(t)$ from both side of equation (2.16) we get

$$s(t+1) - s(t) = s^2(t)\left(1 - \frac{1}{N}\right)^{i(t)N} - s(t) \tag{7.34}$$

Approximating $\left(1 - \frac{1}{N}\right)^{i(t)N}$ as $(1 - iN)$ using Taylor series we get

$$s(t+1) - s(t) \approx s^2(t)(1 - iN) - s(t) \tag{7.35}$$

Approximation left hand side of equation (7.35) in differential form we get

$$\frac{ds}{dt} \approx s^2(t)(1 - iN) - s(t) - \epsilon \tag{7.36}$$

Where ϵ is negative error introduced by approximating left hand side in differential form and results faster decrease in number of susceptible nodes. Error was compensated by introducing a delay. Amount of delay was measured empirically in Section 4.1.4. By omitting error term and further simplification we get

$$\frac{ds}{dt} \approx s^3 - s \tag{7.37}$$

Which has initial condition of

$$s(t_0) = s_0 \tag{7.38}$$

Equation (7.37) can easily be integrated using particle fractions as

$$-\ln(s) + \frac{1}{2}\ln(s^2 - 1) + c \approx t - t_0 \tag{7.39}$$

Solving for s^2 we get

$$s^2 \approx \frac{1}{1 - ce^{2(t-t_0)}} \tag{7.40}$$

Using initial condition given in equation (7.38) we get

$$s_0^2 \approx \frac{1}{1 - c} \tag{7.41}$$

By solving equation (7.41) for c and inserting into equation (7.40) we get

$$s(t) \approx \frac{1}{\sqrt{1 + \left(\frac{1 - s_0^2}{s_0}\right) e^{2(t-t_0)}}} \tag{7.42}$$

7.5. Approximate Analytic SIR Model for Push-pull

Using equation (7.37) push-pull protocol can modeled with a nonlinear system of differential equations as shown in following equations.

$$\frac{ds}{dt} \approx s^3 - s \tag{7.43}$$

$$\frac{di}{dt} \approx -s^3 + s - \gamma(1 - s)i \tag{7.44}$$

$$\frac{dr}{dt} \approx \gamma(1 - s)i \tag{7.45}$$

$$s_0 = 1 - \frac{1}{N}, i_0 = \frac{1}{N}, r_0 = 0 \tag{7.46}$$

CURRICULUM VITAE

Yunus BÜTÜN

EDUCATION

Master of Science 2020-2022	Akdeniz University Institute of Natural and Applied Sciences, Department of Computer Engineering, Antalya
Scientific Preparation for Master of Science 2019-2020	Akdeniz University Institute of Natural and Applied Sciences, Department of Computer Engineering, Antalya
Bachelor of Science 2012-2019	Hacettepe University Faculty of Engineering, Nuclear Energy Engineering, Ankara