Controlling Gossip Protocol Infection Pattern Using Adaptive Fanout

Satish Verma
Department of Computer Science
National University of Singapore
satishku@comp.nus.edu.sg

Wei Tsang Ooi
Department of Computer Science
National University of Singapore
ooiwt@comp.nus.edu.sg

Abstract

We propose and evaluate a model for controlling infection patterns defined over rounds or real time in a gossip-based protocol using adaptive fanout. We model three versions of gossip-based protocols: the Synchronous Protocol, the PseudoSynchronous Protocol and the Asynchronous Protocol. Our objective is to ensure that the members of a group receive a desired message within a bounded latency with very high probability. We argue that the most important parameter that controls the latency of message delivery is the fanout used during gossiping, i.e., the number of gossip targets chosen in a particular instance of gossip. We formally analyze the three protocols and provide expressions for fanout. We introduce the idea of using variable fanouts in different rounds in the Synchronous Protocol. We define fanout as a function of time for the Asynchronous Protocol such that an expected infection pattern is observed with high probability. For a better understanding of the theoretical model, we develop a PseudoSynchronous Protocol to highlight the modelling done in order to derive time dependent fanout. We show that our protocols generate $\Theta(n \log n)$ messages, which is optimal for gossip protocols. We aim to use the gossiping mechanism for large-scale group communication with soft real time constraints. This would alleviate the dependence on tree-based deterministic protocols which usually lack scalability.

1. Introduction

Gossip-based algorithms offer a scalable, robust, fault-tolerant and probabilistically reliable protocol design paradigm for large systems. A variety of applications such as reliable multicast [1], [3], [10], [6], membership management [4], consistency of database replicas [2], etc., use randomized gossiping. In this paper, we present a model for fine-grained control of the gossiping process by means of the infection pattern using adaptive fanout. Fanout refers to the number of gossip targets in any instance of gossip and is the critical design parameter that affects the overall latency of message dissemination. Thus, we set our task on quantifying fanout to ensure that data can be disseminated to all members with high probability within a fixed time, T_{max}. Usually, the fanout used is a constant throughout the protocol. In protocols described in [1] and [11], the fanout is 1 while in [3] and [5], it is another constant. In [3], it is shown that using a higher fanout can reduce the rounds needed for gossip although too high a fanout inhibits performance due to excess message overhead. In [9], the authors quantify the fanout needed for gossip to succeed in delivering information to all nodes with high probability in random graphs. They show that a fanout of the order of $(log\, n+c+o(1))$ gives a success probability of $e^{-e^{-c}}$. They also describe inter-cluster and intra-cluster fanouts for their hierarchical gossip. In [8], the authors describe spatial gossip, which bounds propagation time by a poly-logarithmic function in distance by choosing gossip targets with a probability which is an inverse polynomial function of distance. Of interest here is a new performance bound in terms of propagation time instead of number of rounds or message overhead. In [7], it is discussed that a generic gossip protocol needs $\Theta(n \log n)$ messages to spread a rumor. Our goal is also to study the time and message overhead performance of gossip protocols, to make it more predictable by quantifying the fanout.

We experimented with designing variable round-based fanout for a well-known Synchronous Round-Based Gossip implementation. But we noticed that using fixed-interval rounds is not time-efficient. To resolve the problem, we let the nodes gossip as soon as they are infected rather than letting them gossip at fixed intervals. The gossip spreads faster in this Asynchronous Protocol. We develop fanout as a function of time for this protocol. The round-based and time-based fanouts are computed according to user requirements, which are specified as an infection pattern over rounds or real time. An infection pattern measures the number of nodes that are expected to be infected at the end of a round or at a particular instance, depending on the protocol. A node is infected when it receives a gossip message for

the first time. We believe that designing a protocol based on user input will allow the user to control the rate of infection and balance the message overhead over time. In the Asynchronous case, we assume that we are given the delay distribution of the network under consideration. We develop the PseudoSynchronous Gossip Protocol to provide insight into our mathematical model. We show that the number of messages generated during the protocols is $\Theta(n \log n)$, in terms of overhead for gossip protocols as described in [7].

Here, we outline the rest of our paper. In Section 2, we discuss the Synchronous Gossip Protocol. In Section 3, we introduce the notion of real time in the process of gossiping. In Sections 4 and 5, we discuss the PseudoSynchronous and the Asynchronous Protocols respectively. We present our experimental results in Section 6 and conclude in Section 7.

2 Synchronous Gossip Protocol

We begin the discussion on the Synchronous Gossip Protocol by stating our assumptions. We assume that the system has only one sender and we describe the protocol for just one message. The same message is gossiped again and again in the system until the gossip dies and the message is received by all nodes with high probability. We also assume that nodes have a global membership view and gossip targets are picked from this global view uniformly and randomly. Another assumption is that the nodes gossip only when they receive the message for the first time and discard duplicates.

2.1 Synchronous Gossip Protocol Definitions

The Synchronous Gossip Protocol proceeds through fixed-period rounds which are larger than the maximum latency of transmission between any pair of nodes in the system, which we quantify as L_{max}. The rounds proceed as 1, 2, 3,..., R_{max}. We consider the initial state when only the sender has the message as round 0. Here, we emphasize that rounds can also be looked upon as hops. Thus, a node infected in round k is infected by a message that has travelled exactly k hops. We say a node is infected when it receives the gossip message for the first time. We now define the parameters to model the protocol.

Definition h-message: A message that has travelled exactly h hops is called an h-message.

Definition U_r^s: The number of nodes that are not infected in the system at the end of round r. Initially, we have $U_0^s = N - 1$ and we expect $U_{R_{max}}^s = 0$ with high probability.

Definition I_r^s: The number of nodes infected during a round r. $I_0^s = 1$, when only the source has the message.

Definition S_r^s: The number of nodes that are infected in the system at the end of a round r.

Definition F_r^s: The fanout value, i.e., a node that is infected by an r-message will propagate the message to F_{r+1}^s neighbors. Thus, we have fanout as a function of rounds.

2.2 Analysis of Synchronous Gossip Protocol

In this section, we analyze the Synchronous Gossip Protocol. We aim to ensure that nodes receive a message within time T_{max} with high probability. The period of a round, T_{period} is greater than L_{max}. Thus, the number of rounds permissible is $\lfloor \frac{T_{max}}{T_{period}} \rfloor$. In the Synchronous Protocol, rounds can also be viewed as hops. Thus, hops vary as 1, 2,..., H_{max} where $H_{max} = R_{max}$.

The next step in our protocol is to come up with a suitable S_r^s over the R_{max} rounds which we refer to as the infection pattern. We associate values according to some rule, e.g., linear or exponential, with the constraint that $S_{R_{max}}^s$ equals N. Once we have fixed S_r^s, we determine the corresponding fanout values. Based on an analysis of the Synchronous Protocol similar to the one presented in [1] and [3], we can express U_r^s via a recurrence relation given by:

$$U_{r+1}^s = U_r^s \times (1 - \frac{F_{r+1}^s}{N-1})^{I_r^s} \text{ for } 0 \leq r < R_{max} - 1 \quad (1)$$

Equation 1 can be approximated to:

$$U_{r+1}^s = U_r^s \times exp(\frac{-F_{r+1}^s \times I_r^s}{N-1}) \text{ for } 0 \leq r < R_{max} - 1 \quad (2)$$

Equation 2 gives an expression for fanout as follows:

$$F_{r+1}^s = \frac{N-1}{I_r^s} \times ln(\frac{U_r^s}{U_{r+1}^s}) \text{ for } 0 \leq r < R_{max} - 1 \quad (3)$$

We also note that there is a simple relationship between U_r^s and I_r^s which can be expressed as:

$$I_r^s = U_{r-1}^s - U_r^s \text{ for } 1 \leq r \leq R_{max} \quad (4)$$

Since we already know infection pattern S_r^s as a function of rounds, we can compute U_r^s and I_r^s, and using these and Equations 3 and 4, we can estimate the round-based fanout values. The fanout values that are obtained will ensure the infection pattern requirements with high probability. Now, we show that the message overhead in the protocol is $\Theta(n \log n)$.

Lemma 2.1 *The number of messages generated in the Synchronous Protocol is $\Theta(n \log n)$, where n is the number of uninfected nodes in the system at the beginning.*

Proof From the definition of F_r^s, we know that a node infected by an r-message generates F_{r+1}^s new gossip messages. The expected number of nodes newly infected by r-messages is I_r^s. So, the expected number of new messages in round $(r+1)$ is $I_r^s \times F_{r+1}^s$. From Equation 3, we know that $I_r^s \times F_{r+1}^s$ is equal to $(N-1) \times ln(\frac{U_r^s}{U_{r+1}^s})$. Summing over all the rounds, we get:

Total Number of Messages = $(N-1) \times \sum_{r=0}^{r=R_{max}-1} ln(\frac{U_r^s}{U_{r+1}^s})$.

Thus, Total Number of Messages is $(N-1) \times ln(N-1)$, where N is the total number of nodes including the sender. ∎

Next, we look at the Synchronous Protocol from a hop-based point of view.

2.3 Hop Based Analysis of Synchronous Protocol

In this section, we look at the protocol in terms of hops instead of rounds.

Definition HopContribution for a hop h: The number of nodes that are be infected by messages that have travelled exactly h hops. It is the same as I_h^s.

Definition P_h^s: The probability that a node gets a message for the first time in the h-th round, i.e, gets infected by an h-message.

We define the term M_i^s as follows in terms of fanout F_i^s and I_{i-1}^s as follows:

$$M_i^s = (1 - \frac{F_i^s}{N-1})^{I_{i-1}^s} \text{ for } 1 \leq i \leq H_{max} \qquad (5)$$

The probability that a node gets infected by a 1-message is given by $P_1^s = 1 - M_1^s$. Here, $I_0^s = 1$. Similarly, the probability that a node gets infected by a k-message is P_k^s = $M_1^s \times M_2^s \times ... \times M_{k-1}^s \times (1 - M_k^s)$. Thus, in general, we have the following for $1 \leq k \leq H_{max}$:

$$P_k^s = M_1^s \times M_2^s \times ... \times M_{k-1}^s \times (1 - M_k^s) \qquad (6)$$

We can use Equation 6 to compute the expected Hop-Contribution for a particular hop value as described in the following lemma.

Lemma 2.2 *The expected number of nodes that get infected by r-messages I_r^s is $P_r^s \times (N-1)$.*

Proof We have observed that $U_{r+1}^s = U_r^s \times (1 - \frac{F_{r+1}^s}{N-1})^{I_r^s}$, which can also be expressed as $U_{r+1}^s = U_r^s \times M_{r+1}^s$.

Thus, $U_{r+1}^s = U_0^s \times M_1^s \times M_2^s \times ... \times M_{r+1}^s$.

Also, $I_{r+1}^s = U_r^s - U_{r+1}^s = U_0^s \times M_1^s \times M_2^s \times ... \times M_r^s \times (1 - M_{r+1}^s)$.

Hence, $I_{r+1}^s = U_0^s \times P_{r+1}^s = (N-1) \times P_{r+1}^s$. ∎

The above expression is equivalent to the earlier analysis which resulted in Equations 1 and 4, but it presents the result in terms of P_h^s. Thus, we see that we can control infection over rounds by using our round-based fanouts. Next, we describe the notion of real time for the PseudoSynchronous and Asynchronous Protocols.

3 Notion of Real Time in PseudoSynchronous and Asynchronous Protocols

Before we discuss how we interpret time, we define the two protocols we wish to discuss.

3.1 PseudoSynchronous and Asynchronous Gossip Protocols

Definition PseudoSynchronous Gossip Protocol: Nodes gossip as soon as they are infected by a gossip message for the first time. By the property of the network, if a node receives an h_1-message at time t_1 and an h_2-message at time t_2, then $h_1 < h_2$ if and only if $t_1 < t_2$. This is the hop-time ordering property of the network.

Definition Asynchronous Gossip Protocol: Nodes gossip as soon as they are infected for the first time. There is no notion of synchronous rounds here. Messages that have travelled different number of hops exist concurrently in the system. This is referred to as hops running concurrently. However, the hop-time ordering of the PseudoSynchronous case is no longer true. Thus, if a node receives an h_1-message at time t_1, and an h_2-message at time t_2, it is possible that $h_1 < h_2$ but $t_1 > t_2$. Whenever this happens, hop-shift has occurred.

3.2 Hop Progress as a Function of Time

In the Synchronous case, not all nodes that get infected in a particular round get the message at the same time. The time of infection within a round depends on the delay distribution between pairs of nodes which is bounded by L_{max}. But within a round, all nodes getting infected become so due to messages which have travelled the same number

of hops. However, in the PseudoSynchronous and Asynchronous Protocols, at any instance, different nodes may get infected by messages which have travelled different number of hops. Therefore, different hops contribute to infections concurrently. Despite this, it is clear that the first hop surely ends before time L_{max}, the second hop surely ends before time $2 \times L_{max}$, and so on. The H_{max} hop ends in time $\leq H_{max} \times L_{max}$. Thus, a particular hop causes infections from the beginning of the gossip protocol to the end of that particular hop. In our analysis, we use Gamma distribution to model delay. However, any realistic distribution can be used instead. For Gamma distribution, if the delay pdf for 1 hop is $Gamma(r, \lambda)$, then the delay pdf for k hops is $Gamma(k \times r, \lambda)$. In a real network, we interpret this as a k-message taking time anywhere between 0 and $k \times L_{max}$ before it infects. Now, we can visualize the progress of hops as a function of time for the PseudoSynchronous Protocol.

4 PseudoSynchronous Gossip Protocol

In this protocol, nodes gossip as soon as they get infected, but the hop-time ordering property is preserved. We first define the parameters to model node states as a function of time.

Definition $I_h^p(t_1, t_2)$: The number of nodes infected by h-messages between time t_1 and t_2.

Definition $I^p(t_1, t_2)$: The number of nodes infected between time t_1 and t_2.

Definition U_t^p : The number of uninfected nodes at any time t. $U_0^p = N - 1$, and we expect with high probability that $U_{T_{max}}^p = 0$.

Definition S_t^p : The number of infected nodes at any time t. $S_0^p = 1$ and we expect with high probability that $S_{T_{max}}^p = N$.

Definition F_h^p : The fanout value in the PseudoSynchronous Protocol, i.e., a node that is infected by an h-message will propagate the message to F_{h+1}^p nodes in its membership view.

Now, we state a result which relates the fanout values in the Synchronous and PseudoSynchronous cases. We use the same number of hops as in the Synchronous case here. Since we have R_{max} hops, we claim that the gossip will die out within time $R_{max} \times L_{max}$. We denote this time by ∞.

Theorem 4.1 *If $F_h^p = F_h^s$, then $I_h^p(0, \infty) = I_h^s$, for $1 \leq h \leq H_{max}$*

Proof We make use of the hop-time ordering property of the PseudoSynchronous Protocol to prove our assertion. We assume that $F_h^s = F_h^p$ for all h. In the Synchronous Protocol, we know the estimate for the number of nodes that get infected in a particular round I_r^s. However, in the Synchronous case, this also represents the number of nodes infected by r-messages, i.e., messages that have travelled exactly r hops. This is due to the fact that in the r-th round, all messages are travelling their r-th hop. In the PseudoSynchronous case, messages that have travelled different number of hops exist together. Consider a particular hop h such that $1 \leq h \leq H_{max}$. We argue that the number of nodes infected by h-messages is the same as I_h^s in expectation.

Due to hop-time ordering, it is not possible for a node which is expected to be infected by an h-message to be actually infected by a message that has travelled more than h hops. Because, we assume that if a node gets two copies M_h and $M_{h'}$ which travel two different number of hops, h and $h' > h$, then M_h takes less time due to hop-time ordering property. Hence we discard $M_{h'}$. Again, if the message has been received as $M_{h''}$ such that $h'' < h < h'$, then it means that both M_h and $M_{h'}$ are dropped, since it has already been infected by a message which has travelled fewer hops so we need not consider it in the h-th HopContribution analysis.

To show that the HopContribution in the PseudoSynchronous case is indeed the same, we use induction. Consider the HopContribution for hop 1. Initially, only the sender has the message. Consider the messages that travel only one hop. The number of gossip messages that travel one hop is F_1^p, which is the same as F_1^s. The probability that an uninfected node is infected is $\frac{F_1^p}{N-1}$. The expected number of newly infected nodes is F_1^p due to 1-messages. It is not possible that these nodes receive a message that has travelled a larger number of hops in less time. Thus, the number of nodes infected by 1-messages is the same as in the Synchronous case which is obtained by Equation 1 and Equation 4 or F_1^s the round 1. Thus, our assertion is true for hop 1. We assume that if $F_h^p = F_h^s$ for $1 \leq h \leq k$, then $I_h^p(0, \infty) = I_h^s$ for $1 \leq h \leq k$. We assume that $F_{k+1}^p = F_{k+1}^s$. We want to show that $I_{k+1}^p(0, \infty) = I_{k+1}^s$.

Since our assertion is true for hop values $1, 2, ..., k$, the expected number of newly infected nodes for hops $1, 2, ...k$ are $I_1^s, I_2^s, ..., I_k^s$. In the $(k+1)$-th hop, the number of nodes that will gossip with fanout F_{k+1}^p is I_k^s. Also, it is not possible that $(k+1)$-messages infect any of $I_1^s, I_2^s, ..., I_k^s$ in less time. Thus, the effective number of nodes that are susceptible to infection by these messages is $N - I_1^s - I_2^s - ...I_k^s$, which is the same as U_k^s. Also, the probability that an uninfected node gets infected by a $(k+1)$-message is $\frac{F_{k+1}^s}{N-1}$, which is the same as in the Synchronous case. At the end of $(k+1)$-th hop, the number of newly infected nodes be-

comes $I^p_{k+1} = U^s_k - U^s_k \times (1 - \frac{F^s_{k+1}}{N-1})^{I^s_k}$, which is the same as the Synchronous case if we combine Equation 2 and Equation 4. Thus, we see that our assertion is true for $(k+1)$-th hop too, which completes the proof. ∎

4.1 HopContribution Equation for PseudoSynchronous Protocol

We now modify the HopContribution equation from Lemma 2.2 to introduce the notion of time. We want to express the probability that a node gets infected by an h-message for the first time within time t. This time t is bounded by the value $h \times L_{max}$. When we use the Gamma distribution, L_{max} is infinity as Gamma has an infinite tail. However, we can fix a threshold beyond which we can claim that the hop has ended with very high probability. If this threshold's value is L_{max}, then the hop has definitely ended. We define here a term M^p_h in terms of fanout for the PseudoSynchronous Protocol, F^p_h, and $I^p_{h-1}(0,\infty)$, which is the number of nodes which have been infected by $(h-1)$-messages at time ∞:

$$M^p_h = (1 - \frac{F^p_h}{N-1})^{I^p_{h-1}(0,\infty)} \text{ for } 1 \leq h \leq H_{max} \quad (7)$$

Next, we use Gamma delay distribution and Equation 7 to define $P^p_{h,t}$, which is the probability that a node gets a message for the first time after h hops within time t. This is similar to P^s_h but includes the notion of time. Thus, $P^p_{1,t} = (1 - M^p_1) \times Gamma(r, \lambda, t)$. Here, $0 \leq t \leq L_{max}$. In general, for $0 \leq t \leq h \times L_{max}$, $P^p_{h,t}$ is given by:

$$P^p_{h,t} = M^p_1 \times M^p_2 \times .. \times M^p_{h-1} \times (1 - M^p_h) \times Gamma(h \times r, \lambda, t) \quad (8)$$

Equation 8 can be used to compute the expected number of nodes that get infected by h-messages at any time t and at time ∞, as explained in the following results:

Lemma 4.2 *At time* ∞, $I^p_h(0,\infty) = P^p_{h,\infty} \times (N-1)$.

Proof We assume that we are using fanout F^p_h for the PseudoSynchronous Protocol. This gives us a corresponding $I^p_h(0,\infty)$ for each of the hops. We have seen from Theorem 4.1 that a corresponding fanout exists in the Synchronous case, which we call F^s_h, such that $I^s_h = I^p_h(0,\infty)$. We have also seen from Lemma 2.2 that $I^s_h = (N-1) \times P^s_h$.

From Equation 6, we know that $I^s_h = M^s_1 \times M^s_2 \times ... \times M^s_{h-1} \times M^s_h \times (N-1)$. We know from Equation 5 and Equation 7 that $M^p_h = M^s_h$ if $F^p_h = F^s_h$ and $I^s_h = I^p_h(0,\infty)$ both hold at the same time. If that is the case, then we can express $I^p_h(0,\infty) = M^p_1 \times M^p_2 \times ... \times M^p_{h-1} \times M^p_h \times (N-1)$ using the equivalence relation between the two protocols from Theorem 4.1. Thus, $I^p_h(0,\infty) = P^p_{h,\infty} \times (N-1)$. ∎

Corollary 4.3 *At any time* t, $I^p_h(0,t) = P^p_{h,t} \times (N-1)$.

Proof The result follows from Lemma 4.2 and Equation 8. It gives an estimate of the number of nodes infected by h-messages at a time t. ∎

Corollary 4.4 *During a time interval* $[t_1, t_2]$, $I^p_h(t_1,t_2) = (P^p_{h,t_2} - P^p_{h,t_1}) \times (N-1)$.

Proof The result follows directly from Corollary 4.3. ∎

Corollary 4.5 *During any time interval* $[t_1, t_2]$, $I^p(t_1,t_2) = \sum_{h=1}^{H_{max}} I^p_h(t_1,t_2)$.

Proof The result follows from Corollary 4.4 by adding up individual $I^p_h(t_1,t_2)$ over all the possible h. ∎

4.2 Obtaining Fanout Equations for PseudoSynchronous Protocol from User Input

In this section, we describe a technique to compute fanout as a function of hops for the PseudoSynchronous case. We assume that we are given the S^p_t at certain instances by the user. The user inputs h set of pairs $(t_k, S^p_{t_k})$ which means $S^p_{t_k}$ nodes are infected at time t_k for $1 \leq k \leq h$. Then, we get a set of h equations in P^p_{k,t_k} for $1 \leq k \leq h$ as shown below:

$$S^p_{t_k} = \sum_{i=1}^{h} P^p_{i,t_k} \times (N-1) \text{ for } 1 \leq k \leq h. \quad (9)$$

Notice that $P^p_{i,t_k} = M^p_1 \times M^p_2 \times ... \times (1 - M^p_i) \times Gamma(i \times r, \lambda, t_k)$. We get h such equations for k = 1, 2,..., h. Gamma or any other cdf can be easily evaluated. We get h values for $P^p_{1,\infty}$,$P^p_{2,\infty}$,...,$P^p_{h,\infty}$. Since $P^p_{1,\infty} = (1 - M^p_1)$, we can compute M^p_1, which gives us F^p_1 as $I^p_0 = 1$. Using F^s_1 set as F^p_1, we compute U^s_1 using Equation 1 and I^s_1 which is the same as $I^p_1(0,\infty)$, using the equivalence in Theorem 4.1. Then, using $P^p_{2,\infty}$, we can compute M^p_2, F^p_2, U^s_2 and $I^p_2(0,\infty)$. Thus, we can compute the fanout values and corresponding parameters for h hops. If we need more hops than the number of points specified by the user, we interpolate and add points to generate that many equations without changing the user requirements. Once we have decided the value of h, we need exactly that many equations to compute each of the $P^p_{h,t}$. Note that given the constraints, the equations can not be solved for all combinations of user inputs. The inputs must have a well-behaved pattern for use in obtaining PseudoSynchronous Protocol parameters. The issue of acceptable user inputs needs to be investigated further. We present the pseudo-code for our steps in Algorithm PSEUDOSYNC.

Algorithm PSEUDOSYNC($(t_1, S^p_{t_1}), ..., (t_h, S^p_{t_h})$)

1. **for** $k \leftarrow 1$ **to** h
2. do $S^p_{t_k} \leftarrow \sum_{i=1}^{h} P^p_{i,t_k} \times (N-1)$;
3. compute $Gamma(i \times r, \lambda, t_k)$ for $1 \leq i \leq h$;
4. compute individual $P^p_{k,\infty}$ by solving the h equations;
5. **for** $k \leftarrow 1$ **to** h
6. compute $M^p_k \; using \; P^p_{k,\infty}$ using Equation 8;
7. compute $F^p_k \; using \; M^p_k$ using Equation 7;
8. compute U^s_k with $F^s_k = F^p_k$ using Equation 1 and Theorem 4.1;
9. compute $I^p_k(0,\infty)$ using Equation 1, Equation 4 and Theorem 4.1 ;
10. **return**

Lemma 4.6 *The fanouts F^p_h can be computed from knowledge of the infection pattern S^p_t at certain instances as described in Algorithm* PSEUDOSYNC.

Thus, if the user specifies an expected S^p_t at times $t_1, t_2, ..., t_h$, we can derive the h fanout values by evaluating expressions similar to Equation 9. Note that any suitable delay distribution curve instead of Gamma distribution can be used as we only need to compute the cumulative frequency distribution values at those time intervals. If the user specifies fewer values, we can include additional points by interpolation of the curve S^p_t such that it obeys user requirements. In the next section, we extend the ideas from the PseudoSynchronous Protocol to derive the time-based fanout for the Asynchronous Protocol.

5 Asynchronous Gossip Protocol

We consider the Asynchronous Protocol where hop-shift occurs and look at what hop-shift does. Assume that we have computed the fanout values in the way described in the previous section for the PseudoSynchronous Protocol. According to the definitions in the PseudoSynchronous Protocol, we here define U^a_t, $I^a_h(t_1,t_2)$, $I^a(t_1,t_2)$ and S^a_t.

Consider a time interval $[t_1,t_2]$. During this interval, nodes get infected due to the various hops that are running concurrently. The HopContribution equations for the PseudoSynchronous Protocol give the expected number of nodes infected by each hop in the PseudoSynchronous case. Once infected, the nodes continue gossiping with a fixed fanout, depending on which h-message they were infected by. In the Asynchronous case, however, due to hop-shift, nodes might get infected by a different h-message with a larger h value in less time unlike in the PseudoSynchronous case and thus use a different fanout value. This would disturb the infection pattern as the number of messages that are introduced in the system during this time interval would be different from that in the PseudoSynchronous Protocol. We want to ensure that the progress of the Asynchronous Protocol closely follows that of the PseudoSynchronous Protocol. For this, we want to develop a new notion of fanout which is time dependent. We call this $F^a(t_1,t_2)$.

Definition $F^a(t_1,t_2)$: The fanout value used in the Asynchronous Protocol, i.e., a node that is infected by a message between t_1 and t_2 will propagate the message to $F^a(t_1,t_2)$ nodes in its view.

5.1 Time Dependent Fanout for Asynchronous Protocol

Our motivation to define a time dependent fanout is to ensure that the number of messages floated in the system during any time interval is the same as in the PseudoSynchronous case, and thus, the expected number of nodes that are infected at any time is the same as in the PseudoSynchronous Protocol. Hence, if the system state in terms of infected nodes and uninfected nodes is the same in both the protocols at the beginning of a time interval t_1, then we expect probabilistically the system state to be the same at the end of that time interval t_2 for any time interval $[t_1,t_2]$. During a time interval $[t_1,t_2]$, the number of nodes that are infected in the PseudoSynchronous Protocol due to messages with a particular h-message value is given by $I^p_h(t_1,t_2)$. These nodes use fanout values F^p_{h+1}. We observe that in the PseudoSynchronous case, nodes which are infected by H_{max}-message do not gossip further. The source send F^p_1 messages at time 0 to start the protocol which we discount in both the protocols. Thus, the number of messages generated between $[t_1,t_2]$ in the PseudoSynchronous case is given by the expression $Messages^p(t_1,t_2)$:

$$Messages^p(t_1,t_2) = \sum_{h=1}^{H_{max}-1} I^p_h(t_1,t_2) \times F^p_{h+1} \quad (10)$$

Similarly, the number of messages generated during $[t_1,t_2]$ in the Asynchronous case is given by the expression $Messages^a(t_1,t_2)$:

$$Messages^a(t_1,t_2) = I^a(t_1,t_2) \times F^a(t_1,t_2) \quad (11)$$

To ensure that the number of new messages in the two protocols is same in any interval, we propose the following definition for fanout by equating Equations 10 and 11,

$$F^a(t_1,t_2) = \frac{\sum_{h=1}^{H_{max}-1} F^p_{h+1} \times I^p_h(t_1,t_2)}{I^a(t_1,t_2)} \quad (12)$$

Lemma 5.1 *In a time interval $[t_1,t_2]$, if $U^a_{t_1} = U^p_{t_1}$, and $F^a(t_1,t_2)$ is as defined in Equation 12 for the Asynchronous Protocol, then $U^a_{t_2} = U^p_{t_2}$ in expectation.*

Proof We assume that $U^a_{t_1} = U^p_{t_1}$. We have also observed that the number of new gossip messages that are generated during the time interval is the same as in the PseudoSynchronous case if we use $F^a(t_1, t_2)$ from Equation 12 as fanout. Assuming that the gossip targets are picked uniformly and randomly from the entire system , each uninfected node has the same probability of infection in both the protocols.

Here, we prove our assertion. Initially, $U^p_0 = U^a_0 = N - 1$. Thus, t_1 is 0. Considering a time t_2, we show that $U^a_{t_2} = U^p_{t_2}$ in expectation. Since the number of possible gossip targets is the same in the two cases, we can express the probability that a gossip message infects one of them as $\frac{1}{U^a_0}$, which is the same as $\frac{1}{N-1}$. Since targets are randomly but uniformly picked from this entire set, the expected number of new infections will be proportional to this probability and the number of gossip messages generated during the time interval. Since at the beginning $U^p_0 = U^a_0$, the probability of infection is the same. The number of new gossip messages generated is also the same due to our choice of fanout. As a result, the number of nodes which are expected to be infected at the end of this interval is also the same.

When the next interval begins at time t_2, we have $U^a_{t_2} = U^p_{t_2}$ in expectation. Hence the probability of infection by a gossip message will be the same, i.e., $\frac{1}{U^a_{t_2}}$. Since the number of new messages generated will be the same in the next interval, we can expect that the number of nodes that are newly infected in the interval to be the same in the two protocols. Continuing in this manner, we observe that if the number of uninfected nodes is the same in the two protocols at the beginning of an interval of time and the number of new messages generated is also the same, we can expect the number of uninfected nodes to be the same at the end of the interval too.

Note that all the values we discuss in this proof are expected values and differ from run to run. However, on an average based on probabilistic analysis, we can claim that using time dependent fanout will induce the Asynchronous Protocol to behave closely to the PseudoSynchronous Protocol. ∎

Next, we focus on the term $I^a(t_1, t_2)$ and express it in terms of $I^p_h(t_1, t_2)$. With that, we express Equation 12 in terms of PseudoSynchronous parameters.

5.2 Hop Contribution Values in Asynchronous Protocol

We have already pointed out that the HopContribution equations for the PseudoSynchronous case do not work accurately for the Asynchronous Protocol due to hop-shift. Still, we are able to find a relationship between $I^a(t_1, t_2)$ and $I^p(t_1, t_2)$. We claim that, even though the individual HopContribution equations do not work accurately due to hop-shift, the sum of the equations represents the number of nodes infected during a particular time interval, i.e.:

Theorem 5.2 *If $F^a(t_1, t_2)$ is used as defined in Equation 12, then we have:*

$$I^a(t_1, t_2) = \sum_{h=0}^{H_{max}} I^p_h(t_1, t_2) \tag{13}$$

Corollary 5.3 *Equation 13 is true for the Asynchronous case but it is not necessary that $I^a_h(t_1, t_2) = I^p_h(t_1, t_2)$.*

Proof We use the fanout as defined in $F^a(t_1, t_2)$, and given in Equation 12. We identify two cases: when hop-shift occurs and when it does not. Assume that hop-shift does not occur, then the protocol changes to PseudoSynchronous and Theorem 5.2 follows. In this case, we also have $I^a_h(t_1, t_2) = I^p_h(t_1, t_2)$.

Now, assume that hop-shift occurs. We assume that at the beginning of the interval, $U^a_{t_1} = U^p_{t_1}$. Thus, we impose the condition that the number of uninfected nodes is the same in the two protocols at time t_1. In this case, during the interval $[t_1, t_2]$, if a node is expected to be infected by an h-message according to Equation 8 and Corollary 4.4, then either it has been infected by an h-message or if hop-shift has occurred, then it has been infected by an h'-message where $h' > h$. Either way, it has been infected. The equation in Corollary 4.4 is not strictly obeyed any more due to hop-shift but Corollary 4.5 still holds true due to the previous argument that hop-shift means infection, though from a message that has travelled a larger number of hops. Hence, we can claim that in the interval [t_1, t_2], the number of nodes infected remains the same as predicted by the HopContribution equations in the PseudoSynchronous case. This completes the proof. ∎

Thus, the fanout equation can be rewritten by combining Equation 12 and Theorem 5.2 as:

$$F^a(t_1, t_2) = \frac{\sum_{h=1}^{H_{max}-1} F^p_{h+1} \times I^p_h(t_1, t_2)}{\sum_{h=0}^{H_{max}} I^p_h(t_1, t_2)} \tag{14}$$

Thus, the time dependent fanout to be used in the Asynchronous Gossip Protocol can be expressed in terms of parameters of the PseudoSynchronous Protocol as presented in Equation 14. We summarize our algorithm using the following pseudo-code COMPUTEFANOUT with fixed time intervals of duration δ:

Algorithm COMPUTEFANOUT($Pseudo - SyncParams, \delta$)

1. **repeat**
2. do $t_1 \leftarrow 0$, $t_2 \leftarrow \delta$;
3. do $new\_messages \leftarrow \sum_{h=1}^{H_{max}-1} I_h^p(t_1, t_2) \times F_{h+1}^p$ using Equation 10
4. do $new\_nodes \leftarrow I^p(t_1, t_2)$ using Corollary 4.5
5. do fanout(t_1, t_2) $\leftarrow new\_messages/new\_nodes$ using Equation 14
6. do $t_1 \leftarrow \delta$, $t_2 \leftarrow 2 \times \delta$;
7. **until** $new\_messages == 0$
8. **return**

Lemma 5.4 *The message overheads in both the PseudoSynchronous and Asynchronous Protocols are the same as in the Synchronous Protocol, namely,* $\Theta(n \log n)$.

Proof The expected number of messages in the PseudoSynchronous and the Asynchronous Protocols are the same due to the fact that we have designed the fanout by equating the messages. We have also seen the equivalence between the Synchronous and the PseudoSynchronous Protocols in Theorem 4.1, which implies that one can be mapped into the other by equating the fanouts and corresponding $I_h^p(0, \infty)$ and I_h^s. Thus, for every PseudoSynchronous Protocol, there exists a corresponding Synchronous Protocol which has a fixed message overhead of $(N-1) \times ln\,(N-1)$. Thus, we can conclude that the message complexity of the three protocols are exactly the same. ∎

6 Experimental Results and Discussion

In this section, we present our experimental results. We conducted the simulation using NS-2. The topologies were generated using GT-ITM. Our analytical model for the Asynchronous Protocol assumes a Gamma distribution for the delay distribution. However, for our experiments we computed a delay distribution for the actual NS-2 topology under consideration by calculating the actual shortest path delay using static routing for all the node-pairs. This represents the delay distribution for 1 hop. We computed the delay distribution for the higher hops by convolution operation on the delay curve.

For the Synchronous Protocol, we generated a transit-stub topology of 600 nodes. We considered a case with five rounds and a possible infection of 5, 20, 80, 320 and 174 nodes in rounds 1, 2, 3, 4 and 5 respectively. Using Equation 2, we computed the fanout values for each round. These values can be fractional. Thus, if the value is 4.4, then sometimes we use the value 4 and sometimes the value 5, such that over a set of possible runs, the expected fanout was 4.4. Table 1 summarizes the infection pattern and the corresponding fanout values, compares the average simulation results with the analytical values, and presents the standard deviation over a set of 20 experiments.

Table 1. Synchronous Gossip Protocol with Five Rounds

r	U_r^s	I_r^s	F_r^s	$S_{r,anal}^s$	$S_{r,sim}^s$	StdDev
0	599	1	n.a.	1	1	0
1	594	5	5	6	6	0
2	574	20	4.0891	26	26.0	0.55
3	494	80	4.4785	106	105.5	3.76
4	174	320	7.76	426	425.95	9.60
5	0.5	174	10.86	600	599.5	1.86

As we see from Table 1, the use of variable fanouts in various rounds allows the user to control the infection pattern, determine the number of rounds he wants to allow, and accordingly, compute the fanouts. We observed that the individual runs deviated from the analytical values but the average over a set of runs converged to the theoretical values. Table 1 also summarizes the the standard deviation of the observed simulation results. Overall, the expected values do conform faithfully to the theoretical model. We have also seen that the message overhead is a constant independent of fanout values as seen in Lemma 2.1.

For the Asynchronous Protocol, we let the user specify the number of infected nodes he desires at particular instances as ordered pairs (t, S_t). Using this information, we fix the number of hops we want to allow by using the same number of hops as the number of points specified by the user or interpolate accordingly to suit the timing requirements. Using this information and the delay distribution function for various hops, we compute the time dependent fanout using Lemma 4.6 and Equation 14.

For our experiments, we used a 600 node topology generated using GT-ITM in NS-2. We computed the delay distribution for various hops as described before. We considered an example where the user provided an arbitrary infection pattern samples at five time intervals: (5,100), (7.2,200), (10,350), (11,400), (14,510). Using this information and the delay cdfs, we computed the corresponding U_h^s, $I_h^p(0, \infty)$ and F_h^p for the five possible hop values using Lemma 4.6. Figure 1 shows the delay probability distribution curves for the five hops based on NS-2 topology delays. Table 2 summarizes the values of the PseudoSynchronous Protocol parameters. Note that the protocol begins at time 0 when the source gossips to F_1^p nodes, and from there on, the nodes follow the fanout f(t). This is evident in Equation 14 where the number of messages is computed using fanout values from hops 2 to H_{max} while the number of infected nodes is computed over all hops. Whether we include the constant number of messages generated at time 0 is not important as we compute fanout based on incremental values, hence, the constant term gets subtracted. Once we

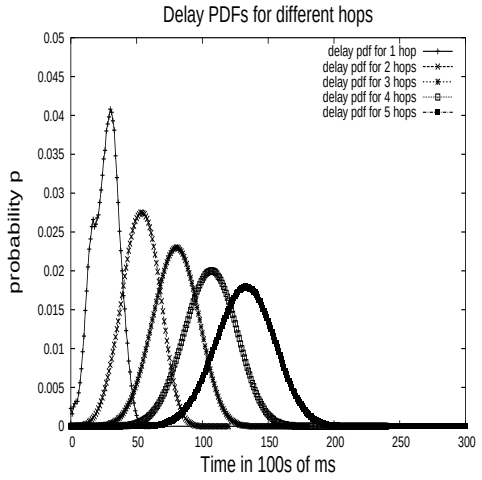

Figure 1. Delay Distribution for 5 hops for a 600-node topology

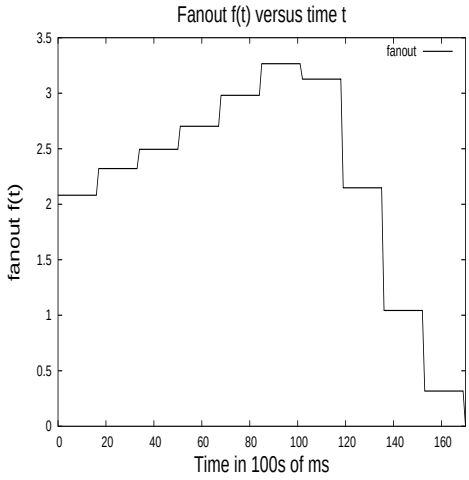

Figure 2. Fanout as a function of time

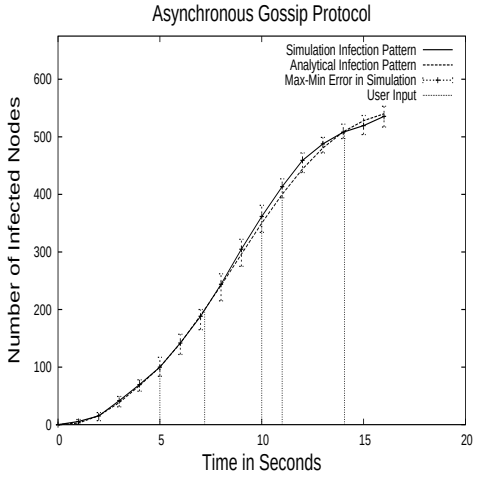

Figure 3. Asynchronous Gossip Protocol Performance

have computed the parameters for the PseudoSynchronous Protocol as shown in Table 2, we migrate to the Asynchronous Protocol. Using Equation 10 and Corollary 4.5, we can compute the number of messages and the number of infected nodes at any instance in the PseudoSynchronous Protocol. Using these two set of values, we finally compute the time dependent fanout. We compute the fanout in the following way: We choose 10 time-intervals to compute 10 fanout values, one for each interval. From the analytical model, we see that all of the messages are generated by 17 seconds. By this time, the first four hops have generated all their messages and the 5-th hop infections do not gossip anymore. Hence, we divide 17 seconds into 10 intervals of 1.7 seconds each and compute the fanout as the number of new messages generated in an interval divided by the number of newly infected nodes in the same interval as described in Algorithm COMPUTEFANOUT. How to choose the interval and whether all the intervals should be of equal length is an interesting question which we wish to investigate in our future research. We believe that we should try to keep the fanout values and interval sizes small so that the theoretical model is faithfully obeyed without much deviation. Table 3 summarizes the steps in computing the fanout f(t) for the ten time intervals in our example.

Figure 2 depicts the time dependent fanout in our example. Figure 3 depicts the average performance of the Asynchronous Protocol over a set of 15 runs compared with the analytical expected infection pattern computed from the PseudoSynchronous model. The vertical impulses in Figure 3 show the user input. We see that the Asynchronous Protocol follows reasonably well the user's requirements in terms of infection pattern at the five points specified. We also depict the maximum deviation from the average that occur in individual runs using yerrorbars. In our experiments, we computed the standard deviation for the data samples at intervals of one second, from $0-15$ seconds, given by [0, 2.2, 4.0, 4.9, 5.9, 8.2, 10.0, 9.98, 12.1, 13.2, 12.3, 10.0, 9.2, 7.7, 8.3, 8.7]. Thus, we can see that, on average the simulation results closely follow the analytical model. Hence, we are able to predict the performance of the gossip protocol with reasonable accuracy. We carried out experiments for five other random user inputs and estimated the error between the actual time taken and the user input values. We found the average error to be around 1.7 %, with the maximum error around 3 %. The protocol follows very closely for most of the time and mostly deviates towards the end when most nodes are infected and in periods following high fanout values. We do believe that we should try to keep the

Table 2. Computation of PseudoSynchronous Parameters Using User Input and Delay PDFs

Input	h	$P^p_{h,\infty}$	M^p_h	U^s_h	$I^p_h(0,\infty)$	F^p_h
5,100	1	0.086	0.91	547.78	51.21	51.21
7.2,200	2	0.162	0.82	450.75	97.04	2.276
10,350	3	0.254	0.66	298.42	152.33	2.54
11,400	4	0.249	0.50	144.39	149.03	2.715
14,510	5	0.166	0.33	49.72	99.43	4.406

Table 3. Computation of Fanout f(t)

$[t_1, t_2]$	new_messages	new_nodes	Fanout f(t)
(0,1.7)	22.76	10.94	2.081
(1.7,3.4)	91.14	39.27	2.321
(3.4,5.1)	133.114	53.347	2.495
(5.1,6.8)	205.92	76.209	2.702
(6.8,8.5)	264.526	88.741	2.981
(8.5,10.2)	299.918	91.823	3.2662
(10.2,11.9)	248.470	79.454	3.127
(11.9,13.6)	126.082	58.681	2.148
(13.6,15.3)	35.248	33.781	1.043
(15.3,17)	4.406	13.920	0.317
(17,18.7)	0.0	3.424	0.0

fanout values within a time interval low. Since our protocol depends on generating the same number of messages and infections within a time interval as the PseudoSynchronous case, using a larger fanout may lead to extra messages and infections within the interval. This phenomenon may cause unexpected deviations from what we expect in the following intervals. Thus, we believe that if the time dependent fanout is designed to have small values, the Asynchronous Protocol will be faithful to the model on an average.

In summary, we believe that fanout is the critical design parameter in designing gossip protocols with a goal of offering a more predictable performance. We have designed and verified a round based fanout for the Synchronous Protocol and a time based fanout for the Asynchronous Protocol, and they follow our analytical model quite closely.

7 Concluding Remarks

We have described three gossip protocols. We see a similarity between the Synchronous and PseudoSynchronous Protocols in terms of fanout equations and HopContribution. This is due to the hop-time ordering property. There is still difference in the two protocols in that the real-time operation of hops in the PseudoSynchronous Protocol is asynchronous. The lack of synchronized rounds makes gossip faster in the PseudoSynchronous case. We have seen that the Asynchronous Protocol is an extension of the PseudoSynchronous Protocol, where the hop-time ordering property is relaxed which is more realistic. We have obtained a time dependent fanout for the Asynchronous case by relating it with the PseudoSynchronous Protocol. We are able to quantify fanout to control gossip infection closely and allow the user to fine-tune this parameter to meet his needs.

We have identified certain issues which need further investigation. The design of good user inputs needs to be considered. We have also noticed that the choice of time interval for computing fanout needs further investigation. This interval need not be fixed but can be chosen appropriately to keep the fanout values small; this is important for the accuracy of the Asynchronous Protocol. We aim to address these issues in our future work.

References

[1] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Trans. on Computer Systems*, vol. 17, May 1999.

[2] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintainance. In *Proc. 7th ACM PODC*, Vancouver, Aug 1987.

[3] P. Eugster, S. Handurukande, R. Guerraoui, A. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. In *Proc. of The Intl. Conf. on Dependable Systems and Networks (DSN)*, Goteborg, Sweden, July 2001.

[4] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulie. SCAMP: P2p lightweight membership service for large-scale group communication. In *Third Intl Workshop of Networked Group Communication*, London, Nov 2001.

[5] K. Guo, M. Hayden, W. Vogels, R. van Renesse, and K. Birman. Gsgc: An efficient gossip style garbage collection scheme. In *Tech. Report CS-TR-97-1656, Cornell Univ.*, Dec 1997.

[6] I. Gupta, A.-M. Kermarrec, and A. J. Ganesh. Efficient epidemic-style protocols for reliable and scalable multicast. In *IEEE SRDS*, Osaka, Japan, Oct 2002.

[7] R. M. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *IEEE Symp. on Foundations of Computer Science*, California, Nov 2000.

[8] D. Kempe, J. M. Kleinberg, and A. J. Demers. Spatial gossip and resource location protocols. In *ACM Symposium on Theory of Computing*, Crete, Greece, July 2001.

[9] A. Kermarrec, L. Massoulie, and A. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Trans. on Parallel and Distributed Systems*, vol. 14, March 2003.

[10] J. Luo, P. Eugster, and J.-P. Hubaux. Route driven gossip: Probabilistic reliable multicast in ad hoc networks. In *Proc. of INFOCOM*, San Fransisco, USA, April 2003.

[11] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure-detection service. In *Proc. IFIP Intl. Conf. on Dist. Sytsems Platforms and Open Dist. Processing*, Lake District, England, Sept 1998.