Abstract

Epidemic algorithms are potentially effective solutions for disseminating information in large scale and dynamic systems. They are easy to deploy, robust and provide high resilience to failures. They proactively fight random process and network failures and do not need any reconfiguration when failures occur. This characteristic is particularly useful in peer-to-peer systems deployed on Internet or ad-hoc networks. In this paper, we study and compare five papers in this area, which investigate the epidemic algorithms and their performance. We also discuss some important issues for disseminating information in the real systems.

1. Introduction

Epidemic, or gossip, protocols have emerged as a robust and scalable method for dissemination of information in large-scale, dynamic systems. In these protocols, each node forwards its local information to randomly chosen peers and data movement only requires pairwise communication. Despite its simplicity, this communication mechanism provides high resilience to random process and network failures. The dissemination of information using epidemic protocols is an interesting problem that has been explored in a wide variety of applications such as data aggregation, database replication, resource discovery and monitoring, routing in wireless networks, and handling web hotspots. In this project, we analyze and compare five papers that investigate epidemic protocols and their applications. More precisely, we look at the ways in which the protocols multicast information and at the mathematical analysis of their performance.

2. Description and Classification of Epidemic Protocols

2.1 Communication Mechanisms

Consider a message that is disseminated in a system using an epidemic protocol. With respect to this message, the nodes are classified in three states:

- **Susceptible**: the node does not know anything about the message, but it can receive it.
- **Infective**: the node has received the message and is spreading it to other nodes.
• **Removed**: the node has received the message but does not spread it.

Based on the communication paradigm between the susceptible and infective nodes, all the gossip protocols fall into three major categories:

- **Push**: each node $u$ chooses a communication peer $v$ and sends it any new information it has. In this mode, the infective nodes are the initiators of the communication.

  ![Fig.2 Push algorithm](image)

- **Pull**: each node $u$ asks a chosen communication peer $v$ for any new information that the peer has. In this mode, the susceptible nodes are the initiators of the communication.

  ![Fig.3 Pull algorithm](image)

- **Push and pull**: each node $u$ chooses a communication peer $v$, sends to the peer any new information it has; at the same time, the node asks its peer for any new information that the peer has.

  ![Fig.4 Push & Pull algorithm](image)

From the algorithms that we have studied, the randomized rumor spreading in [3] and the bimodal multicast in [4] fall into the push-pull category, while the hierarchical and the adaptive disseminations described in [5] fall into the push category.

### 2.2 Stopping the Propagation of a Message

The redundancy and the randomness in the gossip paths are the fundamental reasons why gossip multicast can provide stable throughput and reliable message delivery. However, the redundancy needs to be limited. If each node resends forever every message it has received (the “infect forever” model described in [1]), then the number of disseminated messages in the entire system will continuously increase. In a real implementation, we need a way to stop propagating messages. All the algorithms that we have studied have the capability to decide when to stop spreading information.

In the “infect and die” model [1], a node forwards each received message only once. The node will discard any copies of the same message that it receives later. In [2] is introduced the “aging concept”, where each message has a counter $C_{\text{max}}$ (fig. 5) that decreases after each retransmission of the message. The message transmission stops when the value of $C_{\text{max}}$ becomes zero. The “median counter” algorithm presented in [3] uses $C_{\text{max}}$ to stop message propagation as follows: each unit has a counter for each rumor $R$. A node $P$ increases this counter if and only if all counters $C_{R}(Q_i)$ of its communication partners $Q_i$ in the previous round were at least as big as $C_{R}(P)$. If $C_{R}(P)$ becomes larger than $C_{\text{max}}$, $P$ stops spreading the rumor $R$.

In [4], each node has a counter that increases every time the node produces a new message. Each message is identified by the address of the sender and the counter. The message is initially sent out over a tree. Nodes ask explicitly for messages they did not receive. The algorithms described in [5] use controlled randomness in the message propagation instead of complete randomness. Instead of choosing $b$ other targets uniformly at random from the entire group membership, a node selects the targets that are closer in the leaf box hierarchy with a higher probability.
Fig. 5 Example for spreading information with $C_{max}=2$

3 Mathematical Analysis of the Performance of the Epidemic Algorithms

When a message is multicast using an epidemic protocol, the protocol provides a bimodal delivery guarantee, under which there is a high probability that each multicast will reach almost all processes, a low probability that a multicast will reach just a very small set of processes, and a very small probability that it will reach some intermediate number of processes. The traditional “all or nothing” guarantee thus becomes “almost all or almost none.” Starting from this definition, we measure the performance of different epidemic protocols by comparing the following variables:

- The expected number of nodes that receive a message in or after a given round.
- The probability of atomic broadcast of a message (every node receives that message)
- The number of synchronization rounds needed to propagate a message to all the nodes in the system.

3.1 Message propagation

Initially, we look at idealized systems, in which the components never fail, and the nodes use the “infect forever” model. Under these assumptions, paper [1] gives the expected fraction of nodes that receive a given message after $r$ rounds:

$$Y_r \approx \frac{1}{1+ne^{-fr}}$$

Paper [2] provides the relative number of nodes $s_{t+1}$ that receive a given message in round $(t+1)$, knowing that $s_t$ nodes received the message in the previous round. In the push protocols:

$$s_{t+1} = 2s_t(1 - \frac{c(ln(n))^2}{n})$$

In the pull protocols:

$$s_{t+1} = 2s_t - (s_t)^2$$

These formulas become more complex when they account for node and link failures. Paper [4] defines the probability that $s_{t+1}$ of the $r_t$ susceptible processes receive a gossip in one round, given the fact that the total number of infected processes is $s_t$, out of which $f_t$ are faulty, $p_{ij}$ represents the probability that process $i$ both gossips to process $j$ and the message is delivered, $q_{ij}=1-p_{ij}$. Finally, $q_{lo}$ and $q_{hi}$ represent lower and upper bounds for $q_{ij}$.

$$P(s_{t+1} | s_t, r_t, f_t) \leq \sum_{s_{t+1}=1}^{N} \binom{r_t}{i} (1 - \frac{q_{lo}^{s_t-f_t}}{q_{hi}^{s_t-f_t}})^{r_t-i}$$

$$- \sum_{s_{t+1}=1}^{N} \binom{r_t}{i} (1 - \frac{q_{hi}^{s_t-f_t}}{q_{lo}^{s_t-f_t}})^{r_t-i}$$

3.2 Probability of Atomic Broadcast

In the “infect forever” case, every message is delivered to each node with high probability [3]. The term with high probability means with probability at least $1-O(n^{-\alpha})$, where $\alpha$ is an arbitrary positive constant. It has been shown (see [1] for more details) that using the “infect and die” model instead of “infect forever” decreases the atomic broadcast probability to:
The parameter $c$ is a constant, where the mean number of infected processes evolves with the system size $N$, being equal to $\log(N) + c$.

For the hierarchical gossip presented in [5], the authors prove that, for a large value of the constant $b$ (the number of targets chosen in each round by a node that sends a message), the hierarchical gossip protocol delivers a gossip to all the nodes with probability

$$p = 1 - \frac{3}{2}N$$

In [4], the authors are mainly interested in providing the “almost all or almost none” guarantee. They define a failure predicate, which is an upper bound on the probability that the algorithm delivers a message to more than $(1-\sigma)N$ processes (in other words the almost all or almost none above guarantee is violated). In this recursive formula, $\sigma$ represents the probability of a process failure, $R(s_t, r_i, i, s_{t+1})$ is an upper bound on the probability that $s_{t+1}$ processes will receive the gossip in this round:

$$F_t(s_t, r_t, \tilde{f}_t) \leq \sum_{0 \leq i \leq \tilde{f}_t} \binom{s_t}{f_t}(\sigma)^{f_t}(1-\sigma)^{s_t-f_t} \max_{0 \leq i \leq \tilde{f}_t} \sum_{s_{t+1} \leq r_i} R(s_t, r_i, i, s_{t+1}) F_{t+1}(s_{t+1}, r_t - s_{t+1}, \tilde{f}_t + i)$$

### 3.3 Number of Synchronization Rounds Necessary to Propagate a Message

In the push protocol [2], the distribution of the rumor is terminated after some fixed number of $O(\log_2 n + \ln n + o(1))$ rounds. In the pull case [2], the number of rounds is $\log n + O(\log \log n)$. Finally, for the push & pull protocol ([2, 3]), the rumor is spread in $\log_2 n + O(\log \log n)$ rounds. In the absence of failures, in [4] the algorithm propagates a message to every node in $\log_{fanout} n$, where the fanout is the number of nodes to which a node sends a message during one round.

### 4 Additional Issues

Finally, we discuss some issues related to the epidemic algorithms [1]. Previously we gave some of the mathematical foundations of the algorithms, but when it comes to implement epidemic dissemination algorithm in practice we should consider some important issues.

Membership is a fundamental issue underlying the deployment of an epidemic algorithm for information dissemination. Indeed, in an epidemic dissemination, every process that receives a message may forward it only to other processes that it knows. It is then important to specify how any process acquires its own specific membership information, as this will impact the performance of subsequent epidemic disseminations.

The first approach to this problem is the assumption that every process knows every other process in the system, but it becomes unpractical when we have large system. Another solution, which tradeoff scalability against reliability is to use decentralized protocol providing each process with only partial view of the system. An attractive approach is whenever a process forward a message it includes in this message a set of processes it knows. However, this partial membership results should consider the subjects uniformity, adaptability and bootstrapping.

Another fundamental topic that we should consider is the network awareness or how to make the connections between processes reflect the actual network topology such that the performance is acceptable. For example if the message is sent to very remote process, this may cause high load of the network. The most popular solution is to use a hierarchical organization of the processes and then the epidemic dissemination algorithm then insures that the messages are mostly forwarded to processes within the same branch of the hierarch and this limits the load on core network routers.

Next issue is the buffer management or which information to drop when the storage buffer is full. The paper gives two main strategies. The first is to optimize memory usage by introducing priority between the messages and when the buffer is full and we
have to drop message, we remove the low priority messages preferentially. The other strategy is to reduce the flow of information by regulating its rate. Every process calculates its average rate and if the rate is too high in respect to the average, the process reduces the rate locally.

The last thing to take into account is how to decrease the probability that the processes receive and store information that is of no interest for them (message filtering). One solution is to arrange the processes in hierarchy according to their geographical distances and in the same time to group their interests at each level of the hierarchy.

5. Conclusions

Easy to deploy, robust, and highly resilient to failures, epidemic algorithms are a potentially effective mechanism for propagating information in large peer-to-peer systems deployed on Internet or ad hoc networks.

In this paper, we study and compare five paper considering information dissemination with epidemic algorithms. We classify the algorithms, evaluate their performance and give some important issues for the real implementation.

Although the general application of the epidemic algorithms is in the practical Internet-wide systems, the researchers have opened the question for using epidemic algorithms in applications such as failure detection, data aggregation, resource discovery and monitoring and database replication.

6. References