# *G*ossiping *O*ver *S*torage *S*ystems *I*s *P*ractical

Hakim Weatherspoon[*]
Cornell University
USA
hweather@cs.cornell.edu

Hugo Miranda
Universidade de Lisboa
Portugal
hmiranda@di.fc.ul.pt

Konrad Iwanicki
Vrije Universiteit
Netherlands
iwanicki@few.vu.nl

Ali Ghodsi
SICS
Sweden
ali@sics.se

Yann Busnel
IRISA / URennes 1
France
yann.busnel@irisa.fr

## ABSTRACT

Gossip-based mechanisms are touted for their simplicity, limited resource usage, robustness to failures, and tunable system behavior. These qualities make gossiping an ideal mechanism for storage systems that are responsible for maintaining and updating data in a mist of failures and limited resources (e.g., intermittent network connectivity, limited bandwidth, constrained communication range, or limited battery power). We focus on persistent storage systems that, unlike mere caches, are responsible for both the durability and the consistency of data. Examples of such systems may be encountered in many different environments, in particular: wide-area networks (constrained by limited bandwidth), wireless sensor networks (characterized by limited resources), and mobile ad hoc networks (suffering from intermittent connectivity). In this paper, we demonstrate the qualities of gossiping in these three respective environments.

## Categories and Subject Descriptors

C.2.4 [**Computer-communication networks**]: Distributed systems

## General Terms

Algorithms, Reliability

## Keywords

distributed storage, persistent storage, gossiping, gossip-based storage, durability, update propagation, wide-area networks, wireless sensor networks, mobile ad hoc networks

## 1. INTRODUCTION

Non-traditional storage systems are pushing the boundaries into new distributed storage domains, such as the wide-area networks (WANs)[1, 32], wireless sensor networks (WSNs) [3, 28], and Mobile Ad hoc NETworks (MANETs) [19]. These environments are often characterized by constrained resources, such as access link bandwidth in WANs, limited battery life in WSNs, and intermittent connectivity in MANETs. All of these environments suffer from some rate of node failure. We refer to the above environments as *challenged networks*.

[*]The authors are in the reverse alphabetical order.

Providing guarantees for *durability* and *consistency* is difficult in these new environments since node failures occur often and required resources are limited. Durability concerns data replicas vanishing from the system as a result of node failures. When the last replica disappears permanently (i.e. permanent node failure), the data item is permanently lost. A permanent node failure is when data stored by a node is lost forever, such as a disk failure where the node never recovers with data intact. Similarly, permanent data loss is when all replicas for a particular data item have been lost forever due to permanent node failures. Therefore, to provide durability guarantees, a storage system must minimize the probability of data loss in the presence of node failures.

Consistency defines how a sequence of read and write operations (possibly performed concurrently and/or over different replicas) are handled by the system and presented to the user. The mentioned constraints in challenged networks make strong consistency models such as linearizability [21] or one-copy serializability [6, 31] difficult to provide. A feasible consistency model in such challenged networks is *eventual consistency* [13, 38]. Eventual consistency means updates are propagated and applied at all nodes such that all replicas eventually become identical given a quiescent network. To achieve eventual consistency, all changes to data items must be propagated to all replicas of the item.

In this paper, we argue that *gossip-based mechanisms* are adequate to deal with the aforementioned storage challenges in these new environments. By gossip-based mechanisms we mean periodic pairwise exchange of bounded size messages between random nodes in the system, where the state of one (or both) changes to reflect the state of the other [13, 22]. Such mechanisms are touted for their simplicity, limited resource usage, robustness to failures, and tunable system behavior [13]. We present characteristics for three different storage environments and describe how gossip-based mechanisms are advantageous for each. However, gossiping is not a "one-size-fits-all" mechanism. There are situations where gossip makes less sense. We present those as well.

The rest of the paper is organized as follows. Section 2 gives background on how to achieve durability and update propagation in storage systems. Each of the subsequent sections will, thereafter, analyze durability and update propagation in a different distributed storage domain. Section 3 focuses on gossiping in Wide-Area Networks, Section 4 concentrates on Wireless Sensor Networks, and Section 5 looks into Mobile Ad-hoc Networks. Finally, Section 6 concludes.

## 2. GOSSIPING IN PERSISTENT STORAGE SYSTEMS

Based on the guarantees they give on the durability of data items, distributed storage systems can be classified into two groups: *caches* and *persistent storage systems*. Caches essentially assume that each data item is soft-state, and thus, at some moment, can permanently disappear from the system, potentially without any notification. A separate data source can later restore the lost data item to the cache. In contrast, the goal of persistent storage systems is to preserve data over time within the system. If the data is lost from the latter system, it cannot be restored and is lost forever. Here, we focus on persistent storage.

In this section, we discuss two components of persistent storage systems: durability and update propagation (Sections 2.1 and 2.2, respectively). We discuss their framework and limitations. Next, we relate gossip to both components in three different environments: wide-area networks (WANs), wireless sensor networks (WSNs), and mobile ad hoc networks (MANETs) (Section 3, 4, and 5, respectively). The goal is to derive intuition and bounds on the feasibility of using gossip in these environments.

### 2.1 Gossiping and Durability

A data item unintentionally vanishes from the system as a result of node failures. Therefore, to provide durability guarantees, a storage system must minimize the probability of data loss in the presence of node failures. This is primarily done through replication.

Replication strategies can be divided into *reactive* and *proactive* approaches. Reactive replication assumes that there exists a mechanism for monitoring the redundancy of data items. If, as a result of node failures, redundancy of some data items drops below a desired level, the system reacts by restoring lost redundancy on other nodes. Proactive replication, on the other hand, assumes that the system replicates data items in advance, in anticipation of future failures. For instance, additional redundancy may be created constantly at a low rate [30, 37].

Reactive replication is quick and can provide durability at low average cost. It is therefore a good solution for traditional storage settings, such as storage area networks (SAN), like EMC's SAN manager, or network attached storage (NAS) systems, like Network Appliance's "Filer", in which either node failures occur relatively rarely or provisioning bandwidth to create new replicas when failures do occur is not a problem. As a result, employing gossiping because of its crucial ability to tolerate failures is often overkill. Moreover, gossiping is a relatively slow process that could even endanger data by not responding to failures quickly enough.

The price to pay for supporting reactive replication, however, is potentially high and bursty bandwidth utilization: each time a number of nodes fail, the system must quickly produce new copies of the lost replicas [7]. Such recovery operations create spikes in bandwidth usage when responding to failures. Unfortunately, in many modern, real-world settings addressed in this paper, provisioning bandwidth for peak usage can be expensive.

For this reason, proactively replicating data items using gossiping may be a more tenable solution. In particular, gossiping allows for constantly creating additional redundancy at low rate and thus evens out peaks in replication-related traffic by shifting the time at which bandwidth is used [37]. Additionally, in power-constrained environments, such as wireless sensor networks and mobile ad hoc networks, predictable communication patterns, as ensured by gossiping, facilitate energy savings [2, 26, 39]. However, special measures must be taken to ensure that a given proactive replication scheme guarantees durability with a desired probability.

Intuitively, the node failure rate, $\lambda$, and replica creation rate, $\mu$, represent a balance between how fast a system loses replicas compared to how fast it can create replicas to compensate for the attrition[1]. This ratio between replica creation rate and node failure rate determines the average replicas per data item the system should expect to support [10, 12, 35]. For example, if the system must handle coincidental bursts of, say, five failures, it must be able to support at least six replicas and hence the replica creation rate must be at least 6 times higher than the replica failure rate. We will refer to the ratio $\mu/\lambda$ as $\theta$. More importantly, $\theta$ can be used to estimate the rate of data loss. Dabek [12] demonstrated that the rate of data loss (the probability that no replicas exist for a particular data item per unit time) decreases as $\theta$ increases by $p[\text{data loss}] = e^{-\theta}$.

This model assumes that inter-arrival failure and repair times are exponentially distributed and uses an $M/M/\infty$ birth-death Markov chain to model the number of replicas that exist for a particular data item (i.e. the current state), and transitions to higher and lower states (i.e. addition of new replicas or loss of replicas, respectively). In particular, a data item is in state $r_i$ when $i$ nodes store a replica of the data item. From a given state $r_i$, there is a transition to a state $r_{i+1}$ with rate $\mu$ when $i > 0$ corresponding to repair. There is a transition to the next lower state $r_{i-1}$ with rate $i\lambda$ because each of the nodes storing an existing replica might fail. When a data item is in state $r_0$, it is permanently lost. An analysis of this birth-death process shows that the expected state is $\theta = \mu/\lambda$. If $\theta < 1$, then the system is not feasible (i.e. not able to maintain data persistently) since the replica creation rate is unable to keep pace with the failure rate. For example, many file-sharing peer-to-peer environments are infeasible since node churn is too high to support persistent storage [7].

This model is a somewhat unrealistic view of the maintenance process: there can be an infinite number of replicas and there is a transition from $r_0$ (in which the data item is lost) to $r_1$. However, it conveys the intuition of how replicas are made and suffices for the purposes of these estimates. A detailed discussion of this and more complete models appear in [10, 12, 35].

### 2.2 Gossiping and Update Propagation

Many storage systems offer users the ability to modify stored data. Concurrent modifications of replicated data items necessitate deployment of *consistency* enforcement al-

[1]The node failure rate $\lambda$ is the average number of times a particular node fails per time unit (assuming that a node can be "renewed" – replaced, after each failure, and then returned to service immediately after failure). In some cases, such as with an exponential distribution, it is the reciprocal of the mean-time-between-failures (MTBF) where MTBF is the average time between failures of a particular node. The replica creation rate $\mu$ is the average number of times at which a particular node *gossips* a particular replica with another node not storing the replica per unit time.

gorithms. Such algorithms define and enforce contracts between the user and the storage system on how concurrent updates on a data item are reflected on its replicas.

The most natural consistency model is *one-copy serializability* [6, 31]. It states that there must exist a total order on all operations such that each operation looks as if it were completed at a single node (that maintained all possible data items). In other words, it gives strong consistency guarantees.

Apart from consistency, any real fault-tolerant distributed storage system must also consider two other aspects, that is: *availability* and *partition tolerance*. Availability requires every operation to be eventually completed (for more formal definitions see [11, 17]), whereas partition tolerance is the ability to complete requests in the presence of network partitions. Unfortunately, according to Brewer's conjecture [8, 17], while it is feasible to provide any two of one-copy serializability, availability, and partition tolerance, one cannot achieve all three of them. Moreover, Coan et al. [11] give a low tight upper bound on the maximal possible availability of a strongly-consistent system in the presence of network partitions. Consequently, systems for which availability and partition tolerance are at stake (see the examples in the subsequent sections), must trade off one-copy serializability for a weaker consistency model.

An example of such a model is *eventual consistency* [13]. It states that in the absence of changes, all replicas of a data item will gradually become identical. Eventual consistency can be easily ensured with gossiping. This approach has been shown to be quite useful in practice [13, 40]. It turns out that it is sufficient for many applications. Moreover, by allowing optimistic updates and providing applications with automated, customizable conflict resolution methods, an eventually-consistent storage system can provide high availability in the presence of network partitions [38].

In the following sections, we examine fault-tolerant distributed storage systems in three different settings: wide-area networks, wireless sensor networks, and mobile ad hoc networks. We show how gossip-based algorithms can be employed to proactively ensure durability of data and to facilitate eventually-consistent data updates with high availability, even in the presence of network partitions.

## 3. WIDE-AREA NETWORKS

Many new distributed wide-area networks (WANs)—like PlanetLab [5], the Global Information Grid (GIG) [1], and GRID—are composed of machines from multiple autonomous organizations that are geographically dispersed. In these systems, servers cooperate to provide services such as persistent storage. Systems designed in this manner exhibit good scalability and resilience to localized failures such as power failures or local disasters.

Unfortunately, distributed systems involving multiple, independently managed servers suffer from new challenges that must be addressed to provide data durability and update propagation guarantees. First, wide-area access link bandwidth is limited and shared between nodes within the same site. As a result, the most expensive operation in the system is sending a new replica over a wide-area link. Second, such systems often exhibit high levels of churn, failure, and even deliberate disruption. In PlanetLab, for example, typically less than half of the active servers are stable (available for 30 days or more) [32]. The dilemma is preventing loss of data and/or updates due to this constant rate of node churn and limited wide-area access link bandwidth[2].

In the next two subsections, we demonstrate how gossip-based mechanisms are used to ensure a target durability (rate of data loss) and rate of update propagation.

### 3.1 Ensuring Durability

Durability in distributed wide-area storage systems can be tuned via adjusting the rate of gossip. In particular, we can measure the impact that the rate of gossip has on durability via the ratio $\theta = \frac{\mu}{\lambda}$ where $\mu$ is the replica creation rate and $\lambda$ is the node failure rate. Recall from Section 2.1 that $\theta \geq 1$ is required for a feasible system; otherwise, replicas would be lost faster than they could be replaced and the system could not store data persistently.

To get an idea of a real-world value of $\theta$, we estimate $\lambda$ and $\mu$ based on the historical failure record for permanent node failures on PlanetLab [32]. From [10], the average permanent failure inter-arrival time for the entire PlanetLab test bed was 39.85 hours. On average, there were 490 nodes in the system, so we can estimate the mean time between permanent failures for a single node as $490 \cdot 39.85$ hours or 2.23 years. Hence, $\lambda \approx 0.449$ permanent node failures per year.

The replica creation rate $\mu$ is dependent on the number of times one node can gossip a particular replica to another node per time unit. It is limited by the achievable network throughput per node, as well as the amount of data that each node has to store. We assume that an entire nodes worth of data may be gossiped before a particular replica is gossiped again. As an illustrative example, if each node stores 1TB worth of data and the access link bandwidth is $150KBps$ (i.e. $1.2Mbps$, which is typical for many wide-area settings such as PlanetLab), then the replica creation rate would be limited by $1TB/150KBps = 83$ days, the amount of time to gossip a nodes worth of data. This yields $\mu \approx 4.4$ per year. Thus, the rate of gossip would be at most four times per year for a particular replica in this example[3].

Therefore, in a system with these characteristics, we can estimate the ratio between the replica creation rate and node failure rate $\theta = \mu/\lambda = 4.4/0.449 \approx 9.82$. This yields an estimated rate of data loss as 5 out of 100,000 data items per year. In practice, the value of $\theta$ is somewhat lower; for example, nodes cannot make copies during downtimes or shortly after a permanent node failure. However, the fact remains that $\theta$ is higher than the minimum for a feasible system ($\theta \geq 1$ defines a feasible system). The system still profits from this because higher values of $\theta$ decrease the time it takes to repair a data item, and thus the "window of vulnerability" during which additional failures can cause the data item to be destroyed.

In general, the rate of gossip can be tuned to match a target durability (rate of data loss) provided that the ratio between the replica creation rate and node failure rate defines a feasible system.

[2]We do not discuss file sharing environments since they are often not sustainable as persistent storage systems [7].

[3]Of course, a persistent storage system may allocate less bandwidth for background repair traffic or may not fill up an entire disk with replicas. As a result, estimates using PlanetLab characteristics may vary; however, the derivation would be similar.

## 3.2 Update Propagation

Assuming that a node storing a replica of a data item can contact any other node storing another replica of this data item, update propagation in wide-area networks has a well-founded mathematical model, based on the theory of infectious diseases [4]. For the sake of simplicity assume that a data item is replicated on all $n$ nodes. Moreover, nodes gossip in rounds, that is, every $\Delta T$ time units, and in each round a node is allowed to contact $f$ other random nodes.

There are essentially two classes of algorithms for update propagation: "infect and die", in which a node with an updated replica tries to propagate the update for $k$ rounds and then stops, and "infect forever", in which all updates are being propagated without any round limit. Here we focus on the latter class, as it ensures eventual consistency. Bailey [4] proves that the ratio of the number of nodes that received the update within $r$ rounds to the total number of nodes is equal to

$$\frac{1}{1 + n \cdot e^{-f \cdot r}}.$$

In other words, this value increases exponentially fast on average, by a factor of $e^f$ in each round.

Additionally, Pittel [33] shows the latency of update propagation in terms of the number of rounds. More specifically, the average number of rounds necessary to propagate the update among all nodes respects the following equation

$$log_{f+1}(n) + \frac{1}{f} log(n) + O(1).$$

Thus, using gossiping, updates are spread very fast in that it takes a logarithmic number of rounds to reach every node in the system.

## 4. WIRELESS SENSOR NETWORKS

Wireless sensor networks (WSN) monitor, track, and possibly control parts of their surrounding environment. They are composed of many small, low-cost, low-power devices that must communicate using wireless medium to store, and share the observations [3]. In this paper, we refer to such devices as *sensors*.

Given the potentially large number of sensors participating in a network and their limited resources, it is crucial to deploy fully decentralized solutions and to balance the load as evenly as possible between participating sensors. In particular, since wireless communication, necessary to store, update, and retrieve data, is the most expensive operation in terms of the power consumed [39], battery life time and small communication range represent the fundamental limitations of a sensor. Despite these limitations of individual devices, a *delay-tolerant* WSN must store information for long periods of time until the information can later be extracted. To this end, delay-tolerant WSNs must provide power-aware durability and update propagation mechanisms, as explained in the remainder of this section.

## 4.1 Ensuring Durability

Similar to wide-area networks, durability in wireless sensor networks can be tuned via adjusting the rate of gossip. Increasing the rate of gossip increases the replica creation rate, $\mu$. However, unlike wide-area networks where the node failure rate, $\lambda$, is independent of the rate of gossip, increasing the rate of gossip in WSNs increases the node failure rate, $\lambda$, as more intensive communication drains batteries of the sensors much faster. Once the gossip rate is defined, a particular value for the replica creation rate, $\mu$, and the node failure rate, $\lambda$, can be derived. Based on these values, we can estimate the impact of gossip on durability using the ratio $\theta = \frac{\mu}{\lambda}$.

| Operation | nAh |
|---|---|
| Transmitting a packet | 20.000 |
| Receiving a packet | 8.000 |
| Radio listening for 1 millisecond | 1.250 |
| Operating sensor for 1 sample (analog) | 1.080 |
| Operating sensor for 1 sample (digital) | 0.347 |
| Reading a sample from the ADC | 0.011 |
| Flash Read Data | 1.111 |
| Flash Write/Erase Data | 83.333 |

**Table 1: Typical power requirement for various operations of a Mica mote.**

For example, assume a network consists of Mica mote devices (which typical consumption is presented Table 1). This particular device has sufficient power to operate for 10 months if each mote uses approximately 8 mAh per day while gossiping 100 times per day. In [27], Mica node hardware has been slightly modified, which reduces the energy consumption for similar tasks to approximately 7 mAh per day increasing node lifetime. With these characteristics, the Mica motes have sufficient power to operate for more than 10 months or 0.83 years. Hence, $\lambda = 1.2$ permanent node failures per year.

Similar to wide-area storage networks, the replica creation rate $\mu$ is dependent on the number of times one node can gossip a particular replica to another node per unit time. From the node failure rate calculation above, the node gossip rate was defined to be 100 per day.

Due to the wireless communication capabilities, we assume that nodes are only gossiping with their neighbors located in their wireless transmission range (implies a 1-hop communication model). The replica creation rate for a particular replica is the gossip rate per node divided by the number of replicas per node. Hence, $\mu = \frac{100}{R}$ per day where $R$ is the number of replicas per node. Assuming $R = 5000$ replicas per node, the replica creation rate for a particular data item stored by a particular node is $\mu = \frac{100}{5000} = 0.02$ per day or $\mu = 7.3$ per year.

Therefore, in a system with these characteristics, we can estimate the ratio between the replica creation rate and node failure rate $\theta = \frac{\mu}{\lambda} = \frac{7.3}{1.2} \approx 6.1$. This yields an estimated rate of data loss as 1 out of 450 data items per year. If, on the other hand, we double the rate of gossip from 100 to 200 per day, the estimated rate of data loss would remain the same. In particular, both the replica creation and node failure rate would double and cancel each other leaving $\theta$ the same.

Often, increasing the rate of gossip decreases a nodes lifetime in WSNs. To tune durability, WSNs need to find opportunities to increase the rate of gossip without increasing the frequency of communication. For example, reducing the number of data items or piggyback replication with other traffic as discussed in Section 5. In any case, setting the gossip rate needs to consider effects on both the node failure rate and replica creation rate.

## 4.2 Update Propagation

Update propagation is a challenging and essential task for WSNs. Similar to durability, it must take into account the communication load on each sensor. Updating data replicas has to be persistent, non-intrusive, and diffuse. Gossip-based protocols are a promising candidate to disseminate information through the network while maintaining good node load balance.

Different update propagation proposals have been considered recently. For instance, software updates is an important application. As sensors are expected to be deployed for long periods of time, they are likely to need software upgrades during their lifetime. Updating the software code automatically on a large number of sensors is a tremendous task, which is usually infeasible to perform manually in a deployed system. As another example, data updates generated by nodes need to propagated to other nodes since the time data of collection is often unknown and not predictable. These delay tolerant networks (DTN) [15] continuously update stored data and need to make data available for later information collection.

In these kinds of update propagation applications, gossiping data represents an attractive solution for sharing information through the network. It permits a balance between the load of each node with an efficient speed propagation.

An analysis of gossip-based models [24] provide a proof of dissemination reliability. On a random graph, if a node's *view* (nodes within transmission range) is constrained to $c = O(\log(n_s))$, where $n_s$ is the size of the network, it assure that every nodes in the system is aware of each broadcast message with a probability of $e^{-e^{-c}}$.

In software update propagation, gossip can assure a propagation speed equivalent to the following equation [9].

$$\frac{\ln\left((n_s+2)\cdot(n_{nh}-2)+4\right)-\ln(2\cdot n_{nh})}{\ln(n_{nh})-\ln(2)}\times p_b \qquad (1)$$

where

- $n_{nh}$ represents the average size of neighborhood (average number of nodes located in the transmission range);
- $n_s$ represents the network size in term of number of nodes;
- $p_b$ represents the period of gossip data emission.

By limiting the number of transmission of the same data (maximum $t$ emission in the following), we can obtain the following result:

$$\frac{\ln\left(\frac{n_s}{2}\cdot\left(\frac{n_{nh}}{2}-1\right)\right)-\ln\left(\frac{n_{nh}}{2}-\left(\frac{n_{nh}}{2}\right)^{1-t}\right)}{\ln(n_{nh})-\ln(2)}\times p_b \qquad (2)$$

If $t = O(\log(n_s))$, we obtain an equivalent propagation speed than which provided by Equation 1.

## 5. MOBILE AD HOC NETWORKS

Mobile Ad Hoc NETworks (MANETs) are decentralized, infrastructure-less wireless networks composed of mobile devices with limited resources. They are of particular interest for situations where communication is required but it is too expensive or impossible to deploy infrastructure, for instance, search-and-rescue operations.

Mobility is the key difference between MANETs and WSNs[4]. The combination of node movement and limited transmission range makes node disconnection an integral part of the MANETs networking model. Disconnection, and therefore failure, are hard to estimate as they depend of multiple factors, some of which driven by the application scenario. Examples are the size of the covered region, the number of nodes, their movement speed, their energy reserves and power consumption rate and the correlation between the movement of different nodes. Next, we illustrate how gossip can be used to overcome these challenges to provide durability and update propagation.

## 5.1 Ensuring Durability

Durability of both the original data and updates are affected by the transient connectivity that characterizes MANETs. Further, MANET nodes are prone to permanent failures (e.g. a device can be damaged while in the field). A distributed storage system can improve the availability of this information both during the MANET lifetime (independently of network partitions) and after (thus surviving permanent failures of some nodes). MANETs amplify the problems that affect wired networks and that we have shown to motivate the use of gossip protocols.

A characteristic of wireless networks is that the number of nodes listening to each broadcast is determined by the transmission power of the sender. If two communicating endpoints are not within transmission range, the message must be retransmitted by intermediary nodes, consuming their resources. Therefore, and in contrast with wired networks, the geographical distance between two communicating endpoints has an impact on the power reserves of the participants. On the other hand, nodes that are geographically close have an increased probability of suffering from some localized failure like interference and network partitions. Designers are faced with a tradeoff: to gossip with nodes that are geographically distributed, improving the resilience to localized failures or to save power by selecting nearby nodes. Geographical distribution in MANETs may represent a significant burden on power consumption: it is necessary to learn about other nodes in the network and to devise routes to reach them. To reduce this burden, in [25] peers are selected from the nodes present in the routing tables constructed by source routing protocols (e.g. DSR [23]).

Many data distribution algorithms for MANETs try to devise the most adequate number and location of replicas for some scenario and application model. Gossip-based algorithms can be found in some of them although, in general, with some adaptation to reduce the number of messages transmitted by the nodes.

The algorithms described in [19] have a strong resemblance with the typical gossip algorithms applied in other network settings: periodically, neighboring nodes gossip the state of their storage space and agree on a disjoint set of data objects to be replicated to improve their availability. One of the concerns is the tradeoff between the frequency of the rounds and the utility of the algorithms in face of node movement.

Data-centric storage algorithms [16, 36] implement an algorithm that follows in spirit the underlying principles of

[4]Although, WSNs can be mobile, we did not consider mobility in the Section 4, instead we consider it now in the context of MANETs.

gossip: messages containing data objects are periodically broadcasted by the node responsible for the primary copy to ensure its replication. A peculiarity of data-centric storage is that the node is unaware of the nodes that store a backup copy. The gossip algorithm relies on the properties of the underlying geographical routing protocol to ensure the delivery of the messages to the correct destinations.

Unawareness of replica location is also a characteristic of the algorithm described in [29]. Nodes use a probabilistic algorithm to decide to store the objects. To save resources, new replicas are created in background, by piggybacking the data objects on unrelated traffic packets.

## 5.2 Update Propagation

The frequent disconnections in MANETs make update propagation a challenging problem. There is a non-negligible probability that at least one of the nodes storing a replica is not available whenever an update is triggered by some other node. Interestingly, the literature is vague in the distinction of new and updated data objects or in declaring the consistency model provided. The problem is trivially solved in the Data-Centric storage since it uses a primary-backup replication model but neglected in the remaining examples above, which define application models that do not consider the need for updates.

To the extent of our knowledge, this issue has not been addressed in a satisfactory manner in multi-hop ad hoc networks and deserves additional attention (see for example [20] for preliminary work). It is our position that gossip protocols can play a substantial role in this particular aspect. This claim is supported by noticing that updates can be rapidly propagated using any algorithm aiming at reliable message delivery in MANETs, some of which even implement gossip-like algorithms (e.g. [14, 18, 34]).

## 6. CONCLUSION

Gossip is a mechanism that performs well in distributed storage settings where resources are limited and failures need to be tolerated while reducing the risk of data and update loss. We provide three case studies: wide-area networks with limited access link bandwidth, wireless sensor networks with limited battery life and communication range, and mobile ad hoc networks with limited connectivity. In these environments, we demonstrate how to tune the rate of gossip to achieve desirable durability levels (rate of data loss) and rate of update propagation. Update propagation is used to ensure eventual consistency. Outside of these environments or others like them, however, gossip may not be a good mechanism if node failure is a rare occurrence or new replicas can readily be created since resources such as bandwidth or battery life are not limited.

## 7. REFERENCES

[1] Agency, N. S. Global Information Grid (GIG). `http://www.nsa.gov/ia/industry/gig.cfm`. May 2007.

[2] Akdere, M., Bilgin, C. Ç., Korpeoglu, I., Ulusoy, O., and Çetintemel, U. A comparison of epidemic algorithms in wireless sensor networks. *Computer Communications (Elsevier) 29* (2006), 2450–2457.

[3] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. A survey on sensor networks. *IEEE Communications Magazine 40*, 8 (2002), 102–114.

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

[5] Bavier, A., Bowman, M., Chun, B., Culler, D., Karlin, S., Muir, S., Peterson, L., Roscoe, T., Spalink, T., , and Wawrzoniak, M. Operating system support for planetary-scale network services. In *Proceedings of the 1st USENIX Symp. on Network Systems Design and Implementation (NSDI 2004)* (2004).

[6] Bernstein, P. A., and Goodman, N. Multiversion concurrency control-theory and algorithms. *ACM Trans. on Database Systems 8*, 4 (1983), 465–483.

[7] Blake, C., and Rodrigues, R. High availability, scalable storage, dynamic peer networks: Pick two. In *Proceedings of the 9th USENIX Workshop on Hot Topics in Operating Systems (HotOS IX)* (2003), pp. 1–6.

[8] Brewer, E. A. Towards robust distributed systems (invited talk). In *The 19th ACM Symp. on Principles of Distributed Computing (PODC 2000)* (2000).

[9] Busnel, Y., Bertier, M., Fleury, E., and Kermarrec, A.-M. GCP : Gossip-based code propagation for large-scaled mobile wireless sensor network. Tech. rep., INRIA, 2007.

[10] Chun, B., Dabek, F., Haeberlen, A., Sit, E., Weatherspoon, H., Kaashoek, M. F., Kubiatowicz, J., and Morris, R. Efficient replica maintenance for distributed storage systems. In *Proceedings of the 3rd USENIX Symp. on Network Systems Design and Implementation (NSDI 2006)* (2006).

[11] Coan, B. A., Oki, B. M., and Kolodner, E. K. Limitations on database availability when networks partition. In *Proceedings of the 5th ACM Symp. on Principles of Distributed Computing (PODC 1986)* (1986), pp. 187–194.

[12] Dabek, F. *A Distributed Hash Table*. PhD thesis, Massachusetts Institute of Technology, 2005.

[13] Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., and Terry, D. Epidemic algorithms for replicated database maintenance. In *Proceedings of the 6th ACM Symp. on Principles of Distributed Computing (PODC 1987)* (1987), pp. 1–12.

[14] Drabkin, V., Friedman, R., Kliot, G., and Segal, M. RAPID: Reliable probabilistic dissemination in wireless ad-hoc networks. Tech. Rep. CS-2006-19, Computer Science Department, Technion - Israel Institute of Technology, 2006.

[15] Gavidia, D., Voulgaris, S., and van Steen, M. Epidemic-style monitoring in large-scale wireless sensor networks. Tech. Rep. IR-CS-012, Vrije Universiteit Amsterdam, 2005.

[16] Ghose, A., Grossklags, J., and Chuang, J. Resilient data-centric storage in wireless sensor networks. *IEEE Distributed Systems Online* (2003).

[17] Gilbert, S., and Lynch, N. Brewer's conjecture and the feasibility of consistent, available,

partition-tolerant web services. *SIGACT News 33*, 2 (2002), 51–59.

[18] Haas, Z. J., Halpern, J. Y., and Li, L. Gossip-based ad hoc routing. In *Proceedings of the 21st Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2002)* (2002), pp. 1707–1716.

[19] Hara, T. Effective replica allocation in ad hoc networks for improving data accessibility. In *Proc. of the 20th Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2001)* (2001), pp. 1568–1576.

[20] Hara, T. Replica allocation in ad hoc networks with periodic data update. In *Proceedings of the 3rd IEEE Int'l Conf. on Mobile Data Management, (MDM 2002)* (2002), pp. 79–86.

[21] Herlihy, M., and Wing, J. M. Linearizability: A correctness condition for concurrent objects. *ACM Trans. on Programming Language Systems (TOPLAS). 12*, 3 (1990), 463–492.

[22] Jelasity, M., and Babaoglu, O. T-Man: Fast gossip-based construction of large-scale overlay topologies. Tech. Rep. UBLCS-2004-7, University of Bologna, Department of Computer Science, 2004.

[23] Johnson, D. B., Maltz, D. A., and Broch, J. *Ad Hoc Networking.* Addison-Wesley, 2001, ch. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks, pp. 139–172.

[24] Kermarrec, A.-M., Massouli, L., and Ganesh, A. J. Probabilistic reliable dissemination in large-scale systems. *IEEE Trans. on Parallel and Distributed Systems 14*, 3 (2003).

[25] Luo, J., Eugster, P., and Hubaux, J.-P. Route driven gossip: probabilistic reliable multicast in ad hoc networks. In *Proceedings of the 22nd Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2003)* (2003), pp. 2229–2239.

[26] Luo, J., Eugster, P. T., and Hubaux, J.-P. Pilot: Probabilistic lightweight group communication system for ad hoc networks. *IEEE Trans. on Mobile Computing 3*, 2 (2004), 164–179.

[27] Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., and Anderson, J. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA 2002)* (2002).

[28] Mathur, G., Desnoyers, P., Ganesan, D., and Shenoy, P. Ultra-low power data storage for sensor networks. In *Proceedings of the 2007 Conf. on Information Processing in Sensor Networks (IPSN) - SPOTS Track* (2006).

[29] Miranda, H., Leggio, S., Rodrigues, L., and Raatikainen, K. A stateless neighbour-aware cooperative caching protocol for ad-hoc networks. DI/FCUL TR 05–23, Department of Informatics, University of Lisbon, 2005.

[30] Morales, R., and Gupta, I. Providing both scale and security through a single core probabilstic protocol. In *Proceedings of the Workshop on Stochasticity in Distributed Systems (StoDiS 2005)* (2005).

[31] Papadimitriou, C. H. Serializability of concurrent database updates. *Journal of the ACM 24*, 4 (1979), 631–653.

[32] Peterson, L., Fiuczynski, A. B. E., and Muir, S. Experiences building planetlab. In *Proceedings of the 7th USENIX Symp. on Operating Systems Design and Implementation (OSDI 2006)* (2006).

[33] Pittel, B. On spreading of a rumor. *SIAM Journal of Applied Mathematics 47* (1987), 213–223.

[34] Pleisch, S., Balakrishnan, M., Birman, K., and van Renesse, R. MISTRAL: Efficient flooding in mobile ad-hoc networks. In *Proceedings of the 7th ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc 2006)* (2006), pp. 1–12.

[35] Ramabhadran, S., and Pasquale, J. Analysis of long-running replicated systems. In *Proceedings of the 25th Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2006)* (2006).

[36] Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R., and Shenker, S. GHT: a geographic hash table for data-centric storage. In *Proceedings of the 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA 2002)* (2002), pp. 78–87.

[37] Sit, E., Haeberlen, A., Dabek, F., Chun, B.-G., Weatherspoon, H., Morris, R., Kaashoek, M. F., and Kubiatowicz, J. Proactive replication for data durability. In *Proceedings of the 5th Int'l Workshop on Peer-to-Peer Systems (IPTPS 2006)* (2006).

[38] Terry, D. B., Theimer, M. M., Petersen, K., Demers, A. J., Spreitzer, M. J., and Hauser, C. H. Managing update conflicts in bayou, a weakly connected replicated storage system. In *Proceedings of the 15th ACM Symp. on Operating Systems Principles (SOSP 1995)* (1995), pp. 172–183.

[39] van Renesse, R. Power-aware epidemics. In *Int'l Workshop on Reliable Peer-to-Peer Systems* (2002).

[40] van Renesse, R., Birman, K. P., and Vogels, W. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Trans. on Computer Systems (TOCS) 21*, 2 (2003), 164–206.