Gossiping over storage systems is practical

Hakim Weatherspoon, Hugo Miranda, Konrad Iwanicki, Ali Ghodsi, Yann Busnel

GOSSIP: Gossip Over Storage Systems Is Practical

Hakim Weatherspoon
Cornell Univ.
USA
hweather@cs.cornell.edu

Hugo Miranda
Lisboa Univ.
Portugal
hmiranda@di.fc.ul.pt

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

Ali Ghodsi
SICS
Sweden
ali@sics.se

Yann Busnel
IRISA / Rennes 1 Univ.
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 the durability and consistency of data. Examples of such systems may be encountered in many different environments, in particular: wide-area networks (limited bandwidth), wireless sensor networks (limited resources), and mobile ad hoc networks (intermittent connectivity). In this paper, we demonstrate the qualities of gossiping in these three respective environments.

1. INTRODUCTION

Non-traditional storage systems are pushing the boundaries into new distributed storage domains, such as the wide-area networks (WANs)[28, 1], wireless sensor networks (WSNs) [3, 25], and Mobile Ad hoc NETworks (MANETs) [18]. 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*.

Providing guarantees for *durability* and *consistency* is challenging 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 object is permanently lost. 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 [20] or one-copy serializability [6, 27] difficult to provide. A feasible consistency model in such challenged networks is *eventual consistency* [13, 34]. 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 [21]. 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.

2. OVERVIEW

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.

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.

Reactive replication is quick and can provide durability at low average cost. It is therefore a perfect 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 attainable 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 [33]. 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, 23, 35]. 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, λ, and replica creation rate, μ, represent a balance between how fast a system loses replicas compared to how fast it can create replicas to compensate for the attrition. This ratio between replica creation rate and node failure rate determines the average replicas per object the system should expect to support [10, 12, 31]. 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 μ/λ as θ. More importantly, θ can be used to estimate the rate of data loss. Dabek [12] demonstrated that the rate of data loss (the probability that no replicas exists) decreases as θ increases by $p[\text{data loss}] = e^{-\theta}$. This model assumes that inter-arrival failure and repair times are exponentially distributed.

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 algorithms. 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, 27]. 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, 36]. 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 [34].

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) [28]. 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.

In the next two subsections, we demonstrate how gossip-based mechanisms are used to ensure a target durability (probability 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 μ is the replica creation rate and λ is the node failure rate. $\theta \geq 1$ is required for a feasible system.

To get an idea of a real-world value of θ, we estimate λ and μ based on the historical failure record for permanent node failures on PlanetLab [28]. 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, $\lambda \approx 0.439$ permanent node failures per year.

The replica creation rate μ 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. For 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.

Therefore, in a system with these characteristics, we can estimate the ratio between the replica creation rate and node failure rate $\theta = \mu/\lambda = 4.4/0.439 \approx 10.04$. This yields an estimated rate of data loss as 1 out of 10,000 objects per year. In practice, the value of θ is somewhat lower; for example, nodes cannot make copies during downtimes or shortly after a permanent node failure. However, the fact remains that θ 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 θ decrease the time it takes to repair an object, and thus the "window of vulnerability" during which additional failures can cause the object 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.

3.2 Update Propagation

Assuming that a node storing a replica of an object can contact any other node storing another replica of this object, 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 an object is replicated on all n nodes. Moreover, nodes gossip in rounds, that is, every ΔT 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.

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.

Additionally, Pittel [29] 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 (WSNs) observe the environment, record and possibly respond to their observations. 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 [35], 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, μ. However, unlike wide-area networks where the node failure rate, λ, is independent of the rate of gossip, increasing the rate of gossip in WSNs increases the node failure rate, λ, as more intensive communication drains batteries of the sensors much faster. Once the gossip rate is defined, particular value for the replica creation rate, μ, and the node failure rate, λ, can be derived. Based on these values, we can estimate the impact of gossip on durability using the ratio $\theta = \frac{\mu}{\lambda}$.

For example, assume a network consists of Mica mote devices (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 [24], 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 μ 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. 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 object 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 objects 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 θ 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 objects or piggyback replication with other traffic as discussed in Section 5. In any case, setting the gossip rate needs to consider affects 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 object 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 [22] provide a proof of dissemination reliability. On a random graph, if a node's *view* (nodes within transmission range) is constrained to $c = O(\log(n))$, where n 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 $t = O(\log(n))$, we obtain the same propagation speed provided by Equation 1.

5. MOBILE AD HOC NETWORKS

Mobile Ad Hoc NETworks (MANETs) are decentralized, infrastructure-less 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. In MANETs, devices communicate with each other using a wireless medium. Because wireless devices have a limited transmission range, it may not be possible for each two of them to establish a direct communication channel. The network is expected to be unstable, with nodes frequently establishing and loosing links between them due to device disconnection or movement, possibly of some of the intermediary hops. Mobility is the key difference between MANETs and WSNs[1].

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. This uncertainty makes, for example, strong consistency nearly impossible. Hence, replication models providing eventual consistency have been the most common approach. In addition, typical application scenarios do not consider dependencies between objects. In particular, data objects are independently generated and may possibly be related only by the queries.

Next, we illustrate how gossip can be used to overcome these challenges to provide durability and update propagation.

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

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).

Given the limited memory of the devices, information collected while in the field may be excessively large to be stored at every node. As a result, research in MANETs focus mostly on partial replication models where no one node contains all the data, yet the system still maintains a high probability that at least one copy for each object survives the frequent failure of the nodes and network partitioning. The most popular trend is to find the most adequate number and location of replicas for different scenarios and application models. Gossip-based algorithms can be found in some of the projects in this research area. The examples presented below show that although gossip algorithms have been considered as the mechanism to support the dissemination of replicas, special care must be taken to prevent an excessive consumption of resources of the devices.

The algorithms described in [18] 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 [32, 16] implement an algorithm that follows in spirit the underlying principles of 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. Instead, messages are addressed to a geographical address. The gossip algorithm relies on the properties of the underlying geographical routing protocol to ensure the delivery of the messages to the correct destinations.

The algorithm described in [26] uses an "on-demand" gossip model for the early dissemination of data objects, which gossips a new object as soon as it arrives. 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. In such a scenario, updates for temporarily off-line nodes must be deferred until the node becomes available.

Interestingly, the literature is vague in the distinction of new and updated data objects. 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 [19] 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 (e.g. [30, 14]).

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] National Security Agency. Global Information Grid (GIG). `http://www.nsa.gov/ia/industry/gig.cfm`. Last accessed February 2007.

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

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

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

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

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

[7] C. Blake and R. Rodrigues. High availability, scalable storage, dynamic peer networks: Pick two. In *Proc. 9th USENIX Workshop on Hot Topics in Operating Systems (HotOS IX)*, page 1.6, 2003.

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

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

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

[11] B. A. Coan, B. M. Oki, and E. K. Kolodner. Limitations on database availability when networks partition. In *Proc. of the 5th ACM Symp. on Principles of Distributed Computing (PODC '86)*, pages 187–194, 1986.

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

[13] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proc. of the 6th ACM Symp. on Principles of Distributed Computing (PODC '87)*, pages 1–12, 1987.

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

[15] D. Gavidia, S. Voulgaris, and M. van Steen. Epidemic-style monitoring in large-scale wireless sensor networks. Technical Report IR-CS-012, 2005.

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

[17] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[18] T. Hara. 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)*, volume 3, pages 1568–1576, 2001.

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

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

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

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

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

[24] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA02)*, 2002.

[25] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. In *Information Processing in Sensor Networks (IPSN) Track on Sensor Platform, Tools and Design Methods for Networked Embedded Systems (IPSN-SPOTS)*, 2006.

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

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

[28] L. Peterson, A. Bavier E. Fiuczynski, and S. Muir. Experiences building planetlab. In *7th Symp. on Operating Systems Design and Implementation (OSDI)*, 2006.

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

[30] S. Pleisch, M. Balakrishnan, K. Birman, and R. van Renesse. MISTRAL: efficient flooding in mobile ad-hoc networks. In *Proc. of the 7th ACM Int'l Symp. on Mobile ad hoc networking and computing (MobiHoc '06)*, pages 1–12, 2006.

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

[32] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: a geographic hash table for data-centric storage. In *Proc. of the 1st ACM Int'l Workshop on Wireless sensor networks and applications (WSNA '02)*, pages 78–87, 2002.

[33] E. Sit, A. Haeberlen, F. Dabek, B.-G. Chun, H. Weatherspoon, R. Morris, M. F. Kaashoek, and J. Kubiatowicz. Proactive replication for data durability. In *Proc. of the 5th Int'l Workshop on Peer-to-Peer Systems (IPTPS '06)*, 2006.

[34] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in bayou, a weakly connected replicated storage system. In *Proc. of the 15th ACM Symp. on Operating Systems Principles (SOSP '95)*, pages 172–183, 1995.

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

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