Quantifying the Efficiency, Scalability, and Robustness of WinFS Replication

Vuk Ercegovac
University of Wisconsin – Madison
1210 W. Dayton St., Madison, WI 53703
fax: 608-262-9777
e-mail: vuk@cs.wisc.edu

Douglas B. Terry
Microsoft Research – Silicon Valley
1065 La Avenida, Mountain View, CA 94043
fax: 650-693-3329
e-mail: terry@microsoft.com

ABSTRACT

The WinFS storage platform supports update-anywhere, peer-to-peer data replication. Due to the wide range of possible system configurations, we study the performance of the novel data replication protocol using simulation. Of interest are how many network messages are sent, the convergence time needed for a modified data object to be propagated to all sites, and how messages and convergence time are affected by failures in the network. The results for configurations of various real and synthetic network topologies show an efficient network utilization since each site receives each modification at most once despite the peer-to-peer architecture. In addition, convergence time is shown to be scalable as the number of sites increases. Finally, the protocol's robustness to link and site failures is quantified and shown to provide good performance in the face of lost messages and transient site unavailability.

Keywords: replicated data, peer-to-peer, weak consistency, scalability, robustness

1. INTRODUCTION

The epidemic-style data replication protocol incorporated into Microsoft's new WinFS storage platform was designed to achieve unprecedented levels of efficiency, scalability, and robustness. Three key requirements guided the development of this novel protocol. First, the protocol must make efficient use of network bandwidth by ensuring that each updated data item is transmitted at most once to each replica in the system even though sites exchange updates in a peer-to-peer fashion over an overlay network of arbitrary topology. Second, the storage platform must scale to thousands of geographically distributed replicas. Third, the protocol must be robust in the face of lost messages, terminated communication sessions, and intermittently connected sites.

This paper presents the results of a simulation study undertaken to demonstrate that the WinFS replication protocol satisfies these demanding design goals. Although the WinFS platform has been fully implemented, simulation was used to evaluate the systems' replication performance under a wide class of configuration and workload parameters that would be difficult to explore in practice. Different communication topologies among replicas were simulated, including the topology of a deployed and widely replicated Active Directory system. The correctness of our custom simulator was validated by comparing its results to the running WinFS system.

Although many optimistic replication protocols have been devised with varied performance characteristics[11], few comprehensive studies have been conducted to evaluate such protocols. The performance results that have been reported mostly concentrate on consistency [13] and conflicts [1][8] rather than overall system properties like robustness and message traffic.

The main contributions of this paper are:

- *precise, measurable definitions of efficiency, scalability, and robustness in a large replicated system, and*
- *performance characterization and analysis of a new knowledge-driven, peer-to-peer replication protocol.*

The following section gives an overview of the WinFS replication architecture and protocol that should be sufficient to understand the key simulation results. Section 3 briefly describes our custom-built simulator. Sections 4 through 6 then present a set of questions dealing with efficiency, scalability, and robustness that we answered using the simulator; these sections show that the examined replication protocol does indeed provide the desired performance characteristics.

Section 7 reviews related work and Section 8 summarizes the conclusions that we draw from this simulation study.

2. WINFS REPLICATION DESIGN

WinFS is an innovative storage system developed at Microsoft that incorporates a new optimistic, state-based [11], peer-to-peer replication protocol [7]. WinFS stores *items* that represent real-world objects such as people and places as well as digital artifacts such as digital photos and email. Items are XML-like data objects that are described by a schema.

A collection of items can be grouped into a *community folder* that is shared among members of a *community*. Each member, referred to as a *site*, maintains a local *replica* of the community folder. A site has the ability to modify (insert, update, and delete) any item without consulting other sites. Under such conditions, multiple, and possibly conflicting, versions of an item may exist in the community. It is the responsibility of the synchronization protocol to propagate updated items between pairs of replicas, thereby driving them towards a consistent state; in the process, update conflicts are detected and resolved.

The sites in a community are assumed to be at least occasionally connected to each other by a network. The network may become partitioned due to link or site failures, or a site may operate in a disconnected mode (such as when a person is working on a laptop while traveling on an airplane). The sites that participate in synchronizations form a topology that is overlaid on the underlying physical network. The choice of which sites synchronize with each other and how frequently sites initiate synchronization are parameters that depend on the needs of a community.

The replication protocol is initiated periodically, say once every 5 minutes, by each site wishing to synchronize data with a neighboring site. Updated items flow one-way from a sender **S** to a receiver **R** as illustrated in Figure 1. First, **R** lets **S** know how up-to-date it is by sending its *knowledge* of the updates that it has learned. A site's knowledge is represented compactly in the form of a version vector plus exceptions [5][7]. Next, **S** determines the set of items *U* that are in its replica but unknown to **R**, and **S** sends each item *u* in *U* to **R**. When **R** receives item *u,* **R** accepts it into its replica, possibly detecting conflicts and resolving them as necessary. Finally, **S** sends **R** a *complete* message including **S**'s knowledge. Assuming all items are received reliably, **R** knows as much as **S** when the synchronization session is completed. Hence **R** can add **S**'s knowledge to its own.

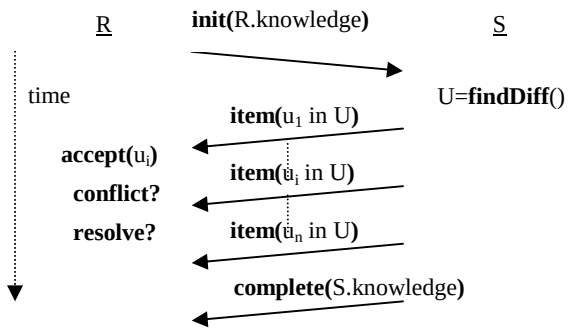

Figure 1: Synchronization protocol initiated by site R receiving needed changes from site S.

R detects conflicting updates by testing whether two versions of the same item were made concurrently. Conceptually, the test is done by associating a history of modifications per item version and checking whether both versions are unknown in the history of the other version. If so, it can be concluded that the versions were created concurrently. It is shown in [5] how the history is implemented concisely (using version vectors). Conflicts can be resolved automatically according to application specific rules or manually. Resolving a conflict produces a new version of the item that propagates to other sites via the normal synchronization protocol. Full details of the WinFS replication design are available in a companion technical report [7].

3. OUR SIMULATOR

Our event driven simulator implements the WinFS synchronization protocol and models an update workload as well as network and site failures. Four synchronization topologies were used in this simulation study: a *clique* in which all sites are fully connected, a *list* in which sites have exactly two neighbors (except for the two endpoints), a *star* in which a central site is connected to all other sites, and the Active Directory topology, a two-level partial mesh of sites described more fully in Section 5.1. Each simulation is a series of synchronization *rounds*. In each round, a number of randomly selected items are modified at various replicas and then each site synchronizes in both directions with a single partner chosen at random from its neighbor sites according to the topology.

In the following sections, we use the simulator to evaluate the WinFS protocol with regard to efficiency, scalability, robustness. The results given are averages over ten runs for each set of input parameters. The simulator was validated by comparing the outputs obtained from running the same workload on the simulator and on an installation of WinFS. We confirmed that our simulator behaves the same as the actual WinFS system for a variety of input parameters, thereby giving credence to the broader simulation study presented in the rest of this paper.

4. EFFICIENCY

To understand the efficiency of the WinFS replication protocol, we asked the following questions and relied on our simulator to provide the answers:

- *For each update performed to an item, how many messages are sent over the network to convey this update to other replicas?*
- *What benefits does state-based replication have on message traffic compared to log-based schemes?*

To be classified as *efficient*, the protocol traffic should increase at most linearly with the number of updates.

For studying the protocol's network message traffic, we distinguish two types of messages: *overhead* and *data*. Overhead messages are the handshake messages, 'init' and 'complete', shown in Figure 1. Data messages are sent to convey updated items that are unknown to the receiving site.

4.1 Network Message Traffic

Intuitively, the total number of data messages seen in the network during a single synchronization round should depend on the total number of modifications between rounds as well as the number of sites in the network. The protocol requires sending an item if its version is unknown at a site and avoids sending an item if it is already known. Due to the property of eventual convergence, for a given modification *m*, each site *s* will eventually synchronize with some site that knows of *m*, thus requiring one data message on behalf of *m* and *s*.

In contrast, overhead messages depend on the number of synchronizations rather than the number of modifications. The number of synchronizations, in turn, depends primarily on the number of sites in the network and its topology in addition to the frequency with which sites initiate synchronization.

In a failure-free network of size *n*, the expectation is that *(n-1)* data messages will be sent for each modification. Furthermore, the number of overhead messages should remain constant as the number of modifications increases. The result in Figure 2 shows the total number of data and overhead messages sent for a given number of modifications made at a single site. For example, assuming that modifications do not overwrite each other, 8 modifications result in 56 data messages, which is expected for an 8 site network.

The result also shows that overhead messages are independent of the number of modifications. Specifically, the number of overhead messages *o* is a little less than 4 per site (3.71) in each round. We will show that *o* varies for different topologies using analytical and empirical evidence.

Intuitively, assuming each site initiates a pair-wise synchronization during a round, four overhead messages are expected to be exchanged: two overhead messages per sync direction. Therefore, we should expect *4n* overhead messages for an *n* site network. However, if a site chooses a partner site that has already chosen it, then the synchronization is unnecessary and therefore skipped. As a result, *4n* total overhead messages are an upper-bound. It will occur, for example, in a ring topology when all sites pick a partner following the same direction around the ring.

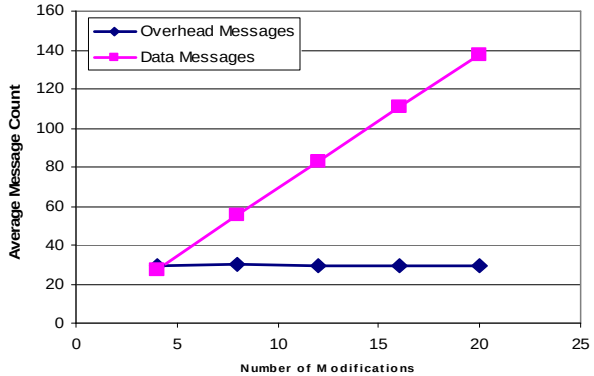

Figure 2: Data and overhead messages per synchronization round as the number of modifications increases in an 8 site clique network.

The last example specifies a particular path of synchronizations for a topology. In order to get a sense

for an average path, the simulator has each site select a random adjacent partner where adjacency is specified by the topology. As a result, pairs of sites can select each other, resulting in a *skip*. The number of skips determines how much lower than the upper-bound we can expect. The number of skips in turn depends on the topology. Intuitively, as the connectivity in the topology increases, assuming partners are randomly selected, the chance of a skip decreases, therefore incurring more overhead messages.

For example, consider a clique topology which has maximum connectivity. The upper-bound for the number of messages is *4n* whereas the lower-bound is $4 * \left\lceil \frac{n}{2} \right\rceil$. The lower-bound occurs when all pairs of syncs are mutually disjoint. However, as *n* increases, selecting disjoint pairs through random selection will decrease in likelihood. Consider a pair of sites *a* and *b*. The probability that they select each other is $\left(\frac{1}{n-1}\right)^2$. Let the random variable X be 1 if *a* and *b* choose each other and 0 otherwise. Thus X follows a Bernoulli distribution with $p = \left(\frac{1}{n-1}\right)^2$. Let the random variable Y be the number of pairs of sites that choose each other, i.e., the number of skips. There are $\binom{n}{2}$ possible pairs so Y is distributed according to the Binomial distribution *Bin(m,p)* where $m = \binom{n}{2}$ and $p = \left(\frac{1}{n-1}\right)^2$. The expected number of skips for *n* sites is $E[Y] = \left(\frac{n}{2(n-1)}\right)$. Computing the number of overhead messages proceeds by subtracting the expected number of skips:

$$4 * \left(n - \left(\frac{n}{2(n-1)}\right)\right)$$

Equation 1: Expected number of overhead messages

For example, the average number of overhead messages in Figure 2 is 29.71 which is identical to the expected number of messages for *n=8*. Equation 1 approaches *4*n* as n tends to infinity.

Other topologies are similarly analyzed. For example, consider a star topology of size *n* where there is one central site and *(n-1)* outlying sites, each of which are adjacent only to the central site. In this case, each of the outlying sites chooses the central site as a partner while the central site chooses one of the outlying sites. As a result, there will be only one skipped synchronization which corresponds to *4*(n-1)* messages. For *n=8*, 28 messages are expected, which we confirmed empirically by repeating the experiment shown in Figure 2, but using a star topology instead.

Another topology with reduced connectivity is a linked list. In this case, sites on the end of the list have one adjacent partner while all other, internal sites have two adjacent partners. The end sites have a ½ chance for their partner to pick them whereas pairs of internal sites have a ¼ chance. Since there are *n-2* internal sites, leading to *n-3* pairs, we have *(1/2 + ½ + (n-3)*1/4)* expected syncs, thus *4*(n – (1 + (n-3)*1/4))* messages. For *n=8*, 23 messages are expected. Empirically, 23.9 messages are observed on average which corresponds closely to the expected number of messages. Unlike the star topology, randomness in selecting partners is expected to produce variance in the results.

In summary, data messages scale linearly with the number of modifications per round whereas overhead messages are constant and depend on network size and topology. As a result, with respect to the number of modifications, the WinFS protocol results in an efficient total number of messages.

4.2 Effect of State-based Replication

In the previous section, the workload was chosen so as to avoid multiple writes to the same item: overwrites. An example of such a workload is an insert-only workload. Similar behavior would be observed in a replication scheme that logs each local update and sends logged entries during synchronization, such as in the Bayou system [9]. When using state-based replication, however, only the most recent version of an item is sent. In the case of overwrites, only the version from the most recent write will be sent. Therefore, for the case of overwrites, it is reasonable to expect that state based replication will result in data message traffic that is sub-linear with respect to the modification workload.

In the following experiment, overwrites are permitted in a fixed size database while the number of updates per round is increased. The results in Figure 3 show the average number of data messages sent for each modification. For an 8 site network, it is expected that 7 messages are required to update the other sites when modifications do not overwrite each other.

Furthermore, the result should be independent of the total number of modifications. In the case of overwrites however, increasing the number of modifications in a fixed size database, assuming items are selected randomly for modification, reduces the data messages needed per modification. The results in Table 1 show the percentage reduction in network traffic of the overwrite workload when compared to the no-overwrite workload. With state-based replication, since the most recent version is sent, the traffic reduction increases as the update rate increases. Such reductions in message traffic would also be observed with increased write locality.

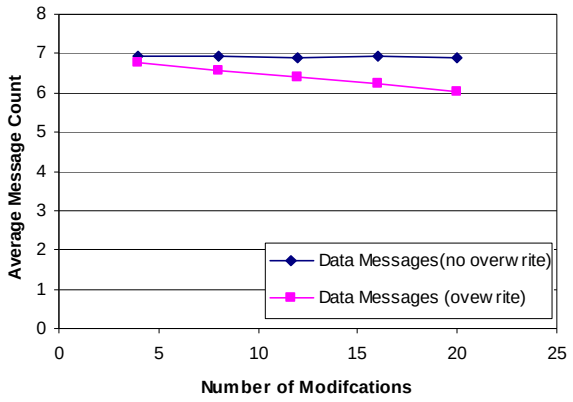

Figure 3: Average number of data messages sent per modification in an 8 site clique network.

A similar situation arises in Section 6.2 where the effect of network and site failures on message traffic is studied. Briefly, errors effectively increase the length of a round by causing synchronizations to be skipped. As a result, more modifications per synchronization round result in more overwrites, and thus fewer data messages.

Number of Modifications	4	8	12	16	20
Percentage Reduction	2%	5%	7%	9%	11%

Table 1: Percentage reduction in message traffic of an overwrite workload in comparison to no-overwrite workload.

Note that objects are randomly selected for updates in the preceding experiments. In contrast, consider a workload that includes a hot spot of update activity. For example, *n* updates are made to an object in one round. The advantage of the state-based scheme is unbounded: one data message is sent whereas a log-based scheme may send *n* data messages.

5. SCALABILITY

Scalability is a measure of how well a system performs as certain configuration or workload parameters are increased. In studying the WinFS replication protocol, we were particularly interested in answers to the following questions:

- *How does the convergence time, the number of synchronization rounds to fully propagate an updated item to all replicas, increase with the number of replicas?*
- *How does the resolution time, the number of synchronization rounds to resolve conflicting updates, increase with the number of replicas and the conflict rate?*
- *Are certain communication topologies more scalable than others?*
- *How does message traffic increase as the number of sites increases?*

To be classified as *scalable*, the average convergence time for propagating updates should increase by no more than 10% when the number of replicas increases by 100%. The resolution time should behave similarly. For message traffic, a single update should be sent at most once per site. Therefore it should not scale worse than linear in terms of the number of sites.

5.1 Convergence Time

Through model checking and formal analysis, the WinFS replication protocol has been proven to guarantee convergence. In this section, we show how long it takes for a number of sites to converge on a modification. Previous work on epidemic algorithms has shown that the expected convergence time is $O(\log n)$ for uniform random connections in a clique [2]. Our first set of experiments confirms this finding and explores other topologies. These results assume a failure-free network; the experiments that follow in section 6 investigate networks with failures.

The experiments when assuming a failure-free network vary the topology and the number of sites to show scalability. The topologies that are considered are clique, linked list, and star from before. In addition, we

considered a topology derived from a real system. Since a real WinFS deployment for this purpose did not exist at the time of the study, we simulated the topology of the Active Directory system in daily use within Microsoft. Active Directory allows a repository of objects to be replicated and written at multiple sites. As in WinFS, weak consistency is provided and sites exchange updated objects through a pair-wise synchronization protocol [12].

An abstraction of the Active Directory topology that we simulated is shown in Figure 4. The illustration shows a two-level organization of sites. At the top-level, the circles are connected by high latency links as is common in a wide-area network (WAN). Each circle represents a cluster of one or more sites that are interconnected by low latency links as is common in a LAN. The overlaid network used for synchronization is constructed automatically but with the possibility for manual overrides. The synchronization network amongst circles is organized such that a minimum spanning tree is formed. There are extra links in the graph at this level for redundancy. Within a circle, a synchronization network of diameter three is formed (no two pairs are more than three hops apart).

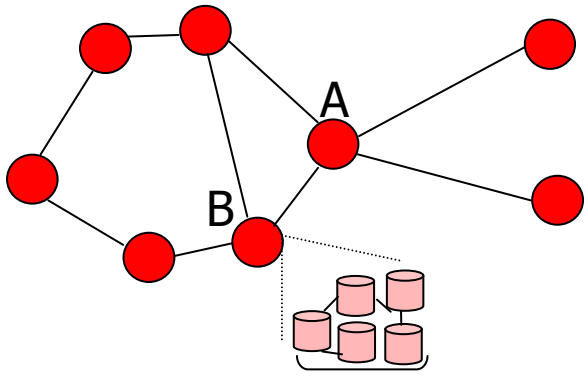

Figure 4: Topology for Active Directory system.

The experiments were run for several sizes of network for each topology. The synthetic topologies are easily scaled to increasing number of sites but it was not obvious how the Active Directory topology should be scaled. To build larger networks, we maintained the same ratios of sites to top-level clusters as observed in the real topology. That is, top-level circle ‘A’ in Figure 4 contains 50% of the total number of sites, top-level circle ‘B’ contain 12% and the remainder are spread evenly amongst the other top-level clusters.

The results are shown in Figure 5, where convergence time is measured in the number of synchronization rounds. A linked list performs the worst, as expected, since it requires the longest possible path. The star topology does best, yielding convergence in constant time due to the path between non-center nodes being of length two. The path between any two nodes is deterministic due the links to the center node. These simulation results assume unlimited bandwidth. That is, in a single round, a site, such as the center of a star topology, can be a partner of any number of other sites with whom it is a neighbor. The clique topology results in the expected logarithmic growth.

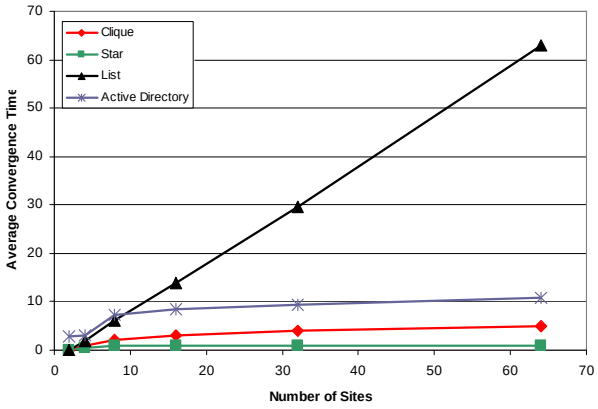

Figure 5: Average convergence time for several topologies as the number of sites increases.

The results in Figure 6 illustrate these trends for a larger number of sites. This figure confirms that, in terms of convergence time, the WinFS design can scale up to thousands of replicas for both star and clique topologies. The scalability of a star-configured system, however, is likely to be limited by the capacity of the central site and its network connections. The Active Directory topology is similar in shape to a clique but with convergence times between those of the clique and the linked list. The results show that in order to achieve scalable convergence time, a topology must be used such that the average path length taken for convergence is logarithmic with respect to the number of sites in the network.

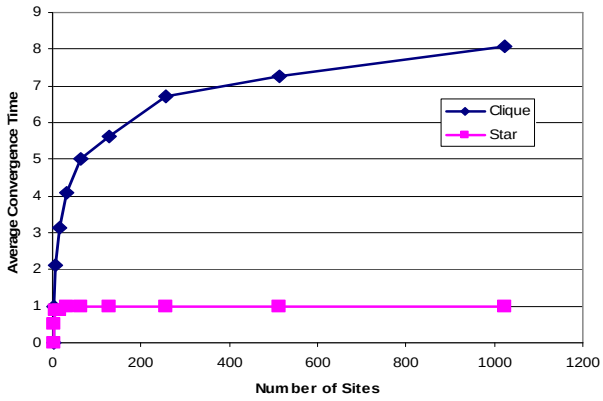

Figure 6: Convergence time for star topology is constant at 1 whereas a clique topology is logarithmic in the number of sites.

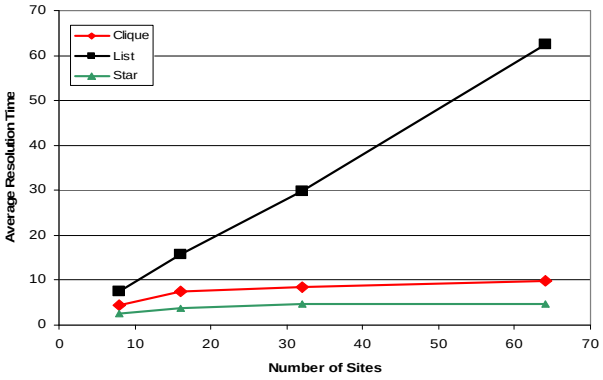

Figure 7: Resolution time is similar to convergence time for all topologies.

5.2 Resolution Time

Resolution time is similar to convergence time but is tailored to the case when the workload includes conflicting modifications to the same item. In this case, as the modifications are sent through the network, conflicts will be detected and resolved, thereby producing new versions to be propagated. Since any site can potentially resolve conflicts, the same conflict may be detected and resolved independently at several sites, possibly leading to versions of an item that themselves need to be resolved. WinFS ensures that the resolution process eventually converges. In this experiment, we performed a number of conflicting updates to an item before the first synchronization round and then measured the time (in rounds) until all sites agree on the same version, assuming that each site can resolve conflicts. The results shown in Figure 7 are similar to the convergence times shown in Figure 5, which confirms that conflict resolution is scalable for clique and star topologies.

We also consider the case where not all sites are capable of detecting and resolving conflicts. Fewer resolving sites require potentially more time for conflicting modifications to reach the resolving sites. On the other hand, having more sites that can resolve conflicts increases the number of concurrent, and hence conflicting, resolutions, which can lead to longer resolution times. Would it be beneficial to use one site as the authoritative conflict resolver and simply propagate conflicting items at the remaining, non-resolving sites? The following experiments consider this question by varying the percentage of sites that can resolve conflicts.

The results in Figure 8 illustrate the competing effects of time to resolve versus number of resolving sites. They show that resolution time differs per topology as the percentage of resolving sites is varied. For clique, the percentage of resolving sites is not a factor. Consider a network with one resolving site. It is likely that not all concurrent modifications will arrive in the same round. As a result, multiple resolutions are propagated into the network resulting in a system-wide state with possibly many versions. However, when all conflicting modifications have been seen by the resolving site, all sites contribute to propagating the final version to all other sites. Now consider a network with two resolving sites. Both may resolve the same conflict identically. When the two resolutions are detected in WinFS, the resolution from the site with the lower site id will win. Thus, all conflicts must go through the site with the lower id which results in a resolution time that is similar to the network with one resolving site.

Resolution time for the star topology can be understood by considering its asymmetry. If the central site is a resolving site, then resolution time is considerably shorter. Since sites are assigned to be resolution sites at random, when the percentage is sufficiently high, the likelihood that the central node is not assigned to be a resolution site becomes very low, thus resulting in a high likelihood of shorter resolution times.

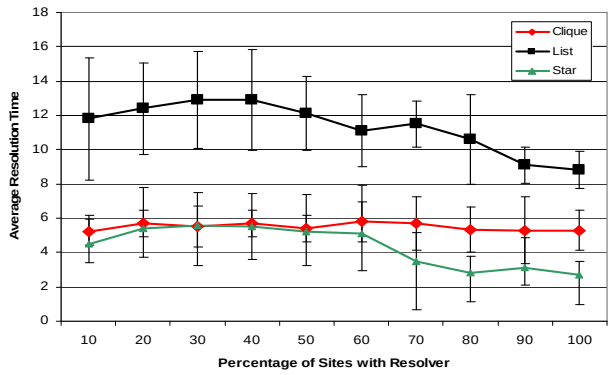

Figure 8: Resolution time for a percentage of resolving sites. Each network has 10 sites and 4 concurrent modifications are made in the first synchronization round.

For the list topology, when there is one resolving site, all conflicting modifications must propagate to the resolving site, and then propagate back out to all the other sites. By adding resolving sites, propagation of the final resolution can proceed in parallel. Consider the case where one site resolves all conflicts and propagates its resolution to another resolving site. If the first site has a lower id, the second site will use the received resolution as a winner. Resolution will complete without having to revisit the already seen sites. However, there is a possibility that not all conflicts are seen by the first site when the two resolutions are detected. As a result, potentially more time will be required in order to revisit previously seen sites. The latter factor has greater influence when there are fewer resolution sites since intermediate, non-resolving sites propagate old resolutions. However, when there is a large percentage of resolving sites, revisiting previously seen sites following the resolution of all conflicts is less likely, thus leading to faster resolution times.

5.3 Message Traffic

The results in Section 4 show that the number of messages is efficient with respect to the number of local modifications. The next experiment shows how message traffic increases as the number of sites *n* is increased. The results in Figure 9 show that when the modification workload remains constant at *u*, adding extra sites results in a proportional number of total messages. The rapid rise in the region of the curve where the number of sites is small is due to larger than *2x* amount of work for *2n* sites. In a system with few sites, the number of overhead and data messages is reduced because of synchronizations that are skipped due to redundantly chosen partners. For example, with 2 sites, there is only one other site with which to perform synchronization whereas for 4 sites, there are 3 other potential partners. Hence, the traffic with 4 sites is more than twice that with 2 sites. After about 20 sites, the curve levels off indicating that the protocol meets the stated scalability requirements.

The results in Figure 9 re-enforce the expected, reduced message traffic as a result of using a state-based representation. The curve corresponding to the analytic model is simply the sum of Equation 1 and the expected data messages *(n-1)*u* divided by the number of total sites, *n*. It assumes that there are no overwrites in the workload so overestimates the results from the simulator that allow for overwrites.

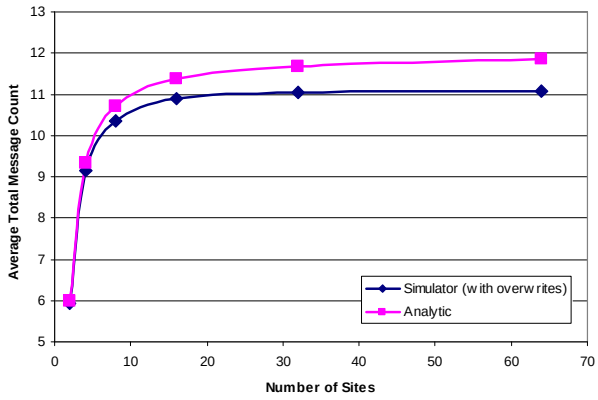

Figure 9: Average total messages per site, per round as the number of sites increase in a clique network. The number of modifications is constant at 8 per round.

6. ROBUSTNESS

A system is robust if it continues to operate effectively and correctly in spite of site and communication failures. In the case of WinFS replication, we explored the following questions:

- *How do communication outages and message loss affect the convergence time?*
- *Which synchronization topologies are more tolerant of network failures?*
- *How does site availability affect overall message traffic?*

To be classified as *robust*, the WinFS replication protocol should (a) guarantee that replicas eventually converge to a consistent state as long as communication between replicas is not permanently disrupted and (b) experience an increase in convergence time, compared to a fault-free system, of at most a factor of two when site availability drops to 80%, when the message loss rate approaches 20%, or when 10% of synchronization sessions terminate prematurely. Eventual convergence of the WinFS protocol was proven using formal methods and model checking, and so our simulation study focused on the performance degradation resulting from transient failures.

We consider three types of failures: site unavailability, message loss, and session termination. Site availability is the inverse of site unavailability, which indicates how many sites are unreachable in the network due to either link failures or site failures. For example, if a site is 80% available, then 8 out of 10 attempts to synchronize with the site will succeed. While our simulations consider a broad range of availability, we expect practical distributed systems to operate above the 80% level used to measure the robustness of WinFS. Given target availability, a frequency and duration of failures is determined to yield the availability used in the experiments.

Despite all sites being functional, individual messages may be lost in the network if the underlying transport protocol is unreliable (e.g. UDP). Message loss determines the probability that a message from a sending site is not delivered to the receiving site. A message may be lost at any point during synchronization. If WinFS replication performs well up to 20% message loss, then we consider it to be robust.

If the transport protocol is connection oriented (e.g., TCP), then the failure of concern is session termination rather than message loss. Session termination is similar to message loss except that all messages following a lost message in the current synchronization session are also undelivered. This occurs, for instance, when a mobile device moves out of the range of a wireless base station. The session termination rate is the percentage of synchronization sessions that do not complete successfully. In our simulations, if a session is terminated, it terminates on a random message during synchronization. Any data messages delivered before a session is terminated are accepted by the receiving site.

6.1 Robustness of Convergence Time

The next set of experiments considers failures in the network. With failures, not all modifications are necessarily learned in a single synchronization session and not all sites are available for synchronization. Thus, convergence time is expected to increase. Of interest is how much of an increase is observed, how does the increase differ per topology, and up to what failure rate is the decreased convergence time tolerable? For this experiment, we vary topology and failure rate while keeping the number of sites constant at 8.

The results in Figure 10 show that convergence time varies significantly for different topologies. The results confirm the intuition that a more highly connected topology such as clique is more robust than the topologies with lower connectivity. Furthermore, for all topologies, the knee in the curve where convergence time rises dramatically is well below the target 80% availability.

The results in Table 2 focus on the portion of the graph corresponding to availability greater than 70%. This table presents the percentage increase in convergence time for various topologies and site availability compared to convergence time assuming no failures. Convergence time for clique and list topologies slows down by less than 100% when site availability is 80% or above. In contrast, for a star topology, convergence slows down unacceptably for availability below 90%. Thus, a star is not a robust topology according to our definition. This is due to the central site potentially blocking all progress in the case that it is unavailable. In a realistic implementation, however, the central site may have a failure rate that is lower than the other connecting sites. In a clique, on the other hand, the replication process can make progress by potentially circumventing unavailable sites. When a list topology is used, the replication process makes progress for higher levels of availability due to a decreased probability of being blocked. Furthermore, the replication process can make progress towards unavailable sites during which time these sites may become available. However, when availability is too low, the probability of being blocked by neighboring, unavailable sites increases, thus impeding progress.

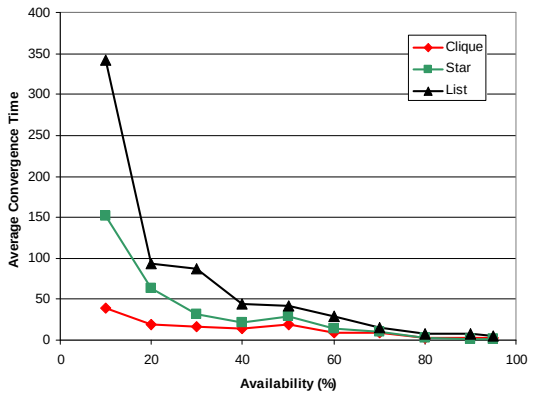

Figure 10: Convergence time for different topologies as site availability increases.

Availability / Topology	95%	90%	80%	70%
Clique	21	26	41	328
Star	77	96	177	977
List	5	34	57	199

Table 2: The percentage slowdown of convergence time for several combinations of topologies and availability.

Similar experiments were conducted for studying the effect of message loss and session termination on convergence time. Table 3 shows the percentage increase in convergence time as the message loss rate increases. Cliques and lists experience an increase of less than 100% up to a message loss rate of 20%. These results demonstrate once again that clique and list are robust topologies whereas a star topology is not.

Message Loss / Topology	5%	10%	20%
Clique	13	26	62
Star	60	137	271
List	3	34	65

Table 3: The percentage slowdown of convergence time for combinations of topologies and message loss rate.

The results in Table 4 show a similar relationship between topology and robustness when sessions are terminated. The clique topology is robust up to a 10% session termination rate. In addition, the results show that session termination has a larger effect on convergence time than message loss since, for each synchronization that is terminated, all messages that would have followed the failure are lost. However, it is assumed that session termination using a reliable transport occurs less frequently than message loss with an unreliable transport, and so a lower threshold for determining robustness is reasonable.

Session Termination / Topology	5%	7.5%	10%
Clique	46	68	96
Star	191	445	629
List	62	75	140

Table 4: The percentage slowdown in convergence time for combinations of topologies and session termination rate.

6.2 Robustness of Message Traffic

The final experiments test the robustness of message traffic when sites are unavailable. The results in Figure 11 extend the results shown in Table 1 where we study the effect of overwriting modifications. Network failures and site unavailability have a similar effect. Specifically, unavailable sites increase the number of modifications per site between successful synchronizations. As a result, the likelihood of an overwrite increases, leading to a decrease in the number of messages due to the state-based replication scheme. In addition, although not shown, topology has an effect on message traffic. Due to the increased connectivity, and hence robustness, of the clique topology, more messages are sent in comparison to star and list topologies.

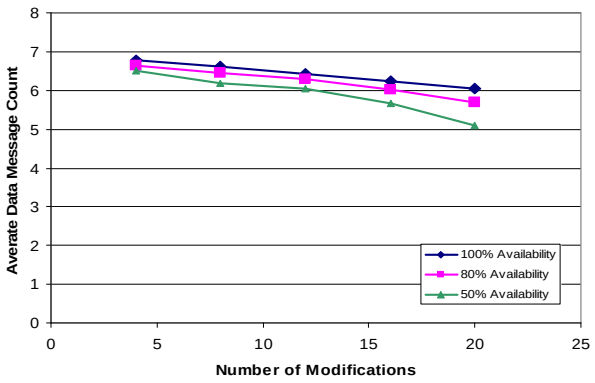

Figure 11: Message counts decrease as availability decreases for an 8 site clique network.

7. RELATED WORK

In this paper, the WinFS replication protocol is characterized with respect to overall system properties such as convergence time and message traffic. In contrast, the designers of other optimistic replication protocols have mostly evaluated them with respect to consistency [13] and conflicts [1][8]. Other researchers have simply noted the difficulty of measuring the quality of service in optimistically replicated systems [4].

One notable exception is the seminal paper on epidemic algorithms that evaluated both anti-entropy and rumor mongering convergence through analysis and simulation [2]. However, the rumor mongering protocols studied by Demers *et al.* have the unfortunate property of leaving a non-zero residue; in other words, replicas only receive updates with some probability. Rumor mongering protocols also are inefficient in that an update may be delivered to a replica multiple times. Table 1 in the epidemic algorithms paper [2] shows that the number of messages sent per replica per update is 6.68 in order to achieve a residue of 0.0012. Mullender and van der Valk also used simulation to explore the tradeoffs between message traffic and infection rates in epidemic protocols [6]. In contrast, the WinFS replication protocol studied in this paper guarantees eventual convergence of all replicas and, as shown in section 4, is efficient in that it maintains a traffic ratio of 1 or less for all topologies. Demers *et al.* suggest backing up a complex epidemic with anti-entropy to ensure full convergence. The efficiency and scalability of WinFS replication, which uses a novel anti-entropy protocol, allows it to be used as the sole means of propagating updates between replicas.

Our simulations of convergence time compliment the previous studies of anti-entropy protocols. In this paper, we study various communication topologies in which synchronization partners are selected at random from the set of neighboring replicas, whereas Demers *et al.* evaluated different distance-biased distributions for selecting partners given the fixed topology of the Xerox Corporate Internet [2]. Golding and Long simulated their timestamped anti-entropy protocol for a number of partner selection policies and topologies, including rings, trees, and meshes [3]. As in our study, they conclude that the mesh topology is the most scalable.

The Ficus [8] and Bayou [9] systems share some key architectural features of WinFS, namely an update-anywhere, optimistic replication model with peer-to-peer reconciliation. Papers on Ficus and Bayou include performance measurements of individual reconciliation times and file system benchmarks but not scalability, efficiency, or robustness. Roam is a descendant of Ficus that employs a two-level hierarchy of replicas to improve scalability by reducing the storage overhead and update distribution time [10]. Ratner *et al.* provide an analysis of update propagation times for a Roam system in which replicas at each level of the hierarchy are arranged in rings. This arrangement results in propagation times that are shorter than in a ring but longer than the other topologies that we studied in section 5.

8. CONCLUSIONS

Compared to replicated data protocols with strong consistency guarantees, such as one-copy serializability, optimistic replication schemes have been developed because of their increased availability and performance, though few researchers have studied their overall system behavior. WinFS employs an instance of an optimistic or epidemic-style protocol that allows sites to exchange updated data items in a peer-to-peer fashion while guaranteeing eventual convergence. Our simulation study confirms that the WinFS replication protocol meets its design goals for efficiency, scalability, and robustness.

The key WinFS replication properties validated or quantified in this study are:

- At most one data message per site is sent for each updated item regardless of the topology of the overlay network used for communication between

sites, the policy that selects synchronization partners, and the frequency of synchronization.

- State-based replication, compared to systems that maintain and replicate update logs, sends fewer messages as the update rate increases since data items may be overwritten multiple times in between synchronization sessions; reductions in message traffic of up to 11% were observed for modest numbers of modifications.

- For clique, star, and the two-level clustered synchronization topology used in the Active Directory system, convergence time and resolution time grow slowly with increasing numbers of replica sites, and hence systems configured in these topologies can scale to large numbers of replicas; list and ring topologies are not scalable.

- Increasing the percentage of sites that can automatically resolve update conflicts generally reduces the overall time for sites to agree on a resolution; however, there is a tension since a larger number of resolvers also results in higher likelihood of concurrent resolutions which prolong the resolution time.

- As the number of sites increases, overall message traffic increases due to fewer redundant synchronizations in a given round; this effect is pronounced for small numbers of replicas but the message traffic curve flattens out for large systems.

- Reducing the availability of communication channels between sites, such as when sites are intermittently connected, reduces the overall message traffic without impacting eventual consistency; however, low availability can have a considerable detrimental effect on convergence time.

- A clique topology is the most tolerant of network and site failures; convergence time increases by less than a factor of 2 for reasonable failure rates such as 20% site unavailability, 20% message loss, and a session termination rate of 10%.

The simulation results presented in this paper were based on synthetic workloads in which modifications were performed on random items at randomly selected replicas. While such workloads were sufficient to answer our questions concerning the efficiency, scalability, and robustness of the WinFS replication protocol, they did not allow us to predict the overall WinFS system performance for real user communities. In the future, we hope to extend the experiments by simulating update workloads obtained from actual distributed systems. These workloads are likely to exhibit more locality than random modifications but should not alter our fundamental conclusions.

9. ACKNOWLEDGMENTS

The replication protocol that we evaluated in this paper was designed and implemented by the WinFS product team in Microsoft and has found use in a variety of applications. We are grateful to the Microsoft product organizations for their support of our research and to the Active Directory team for allowing us to study their deployed system.

10. REFERENCES

[1] Carey, M. J. and Livny, M. Conflict Detection Tradeoffs for Replicated Data. *ACM Transactions on Database Systems* 16(4):703-746, December 1991.

[2] Demers, A., D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic Algorithms for Replicated Database Maintenance. *Proceedings Sixth Symposium on Principles of Distributed Computing*, Vancouver, B.C., Canada, August 1987, pages 1-12.

[3] Golding, R. A. and D. D. E. Long. Modeling Replica Divergence in a Weak-consistency Protocol for Global-scale Distributed Data Bases. Technical Report UCSC-CRL-93-09, University of California, Santa Cruz, 1993.

[4] Kuenning, G. H., R. Bagrodia, R. G. Guy, G. J. Popek, P. Reiher, and A. Wang. Measuring the quality of service of optimistic replication. *Proceedings of the ECOOP Workshop on Mobility and Replication*, Brussels, Belgium, July 1998.

[5] Malkhi, D. and D. Terry. Concise Version Vectors in WinFS. *Proceedings 19th Intl. Symposium on Distributed Computing (DISC)*, Cracow, Poland, September, 2005.

[6] Mullender, S. J. and M. van der Valk. Simulating wide-area replication. *Proceedings 7th ACM SIGOPS European Workshop: Systems Support of World-wide Applications*, Connemara, Ireland, September 1996.

[7] Novik, L., I. Hudis, D. B. Terry, S. Anand, V. J. Jhaveri, A. Shah, and Y. Wu. Peer-to-peer Replication in WinFS. Technical Report MSR-TR-2006-78, Microsoft, June 2006.

[8] Page, Jr., T. W. , R. G.. Guy, J. S. Heidemann, D. H. Ratner, P. L. Reiher, A. Goel, G. H. Kuenning, and G. Popek. Perspectives on optimistically replicated peer-to-peer filing. *Software -- Practice and Experience*, 11(1), December 1997.

[9] Petersen, K., M. J. Spreitzer, D. B. Terry, M. M. Theimer, and A. J. Demers. Flexible update propagation for weakly consistent replication. *Proceedings ACM Symposium on Operating Systems Principles (SOSP)*, Saint Malo, France, October 1997, pages 288-301.

[10] Ratner, D. , P. Reiher, G. J. Popek, and G. H. Kuenning. Replication requirements in mobile environments. *Mobile Networks and Applications* 6:525-533, 2001.

[11] Saito, Y. and M. Shapiro. Optimistic replication. *ACM Computing Surveys* 37(1): 42-81, March 2005.

[12] Stanek, W. R. Microsoft Windows Server 2003 Inside Out. Microsoft Press, June 2004.

[13] Yu, H. and A. Vahdat. The Costs and Limits of Availability for Replicated Services. *Proceedings ACM Symposium on Operating Systems Principles*, 2001, pages 29-42.