Performance Analysis of Flat and Layered Gossip Services for Failure Detection and Consensus in Scalable Heterogeneous Clusters

K. Sistla, A. George, R. Todd, and R. Tilak
High-performance Computing and Simulation (HCS) Research Laboratory
ECE Department, University of Florida, Gainesville, FL

Abstract

Gossip protocols and services provide a means by which failures can be detected in large, distributed systems in an asynchronous manner without the limits associated with reliable multicasting for group communications. Gossiping with consensus can take place throughout the system via a flat structure, or it can be hierarchically distributed across cooperating layers of nodes. In this paper, the performance of flat and layered protocols is analyzed on an experimental testbed in terms of consensus time and scalability. Performance associated with layered gossip is analyzed with varying group sizes and is shown to scale well in a heterogeneous environment.

1. Introduction

With the constantly increasing performance levels and decreasing costs associated with uniprocessor and multiprocessor servers and desktop computers, high-performance cluster systems such as *Cplant* at Sandia National Labs hold the potential to provide a formidable supercomputing platform for a variety of grand-challenge applications. However, in order for heterogeneous and high-performance clusters to achieve their potential with parallel and distributed applications, a number of technical challenges must be overcome in terms of scalable, fault-tolerant computing. These challenges are made all the more complex with the increasing heterogeneity of technologies in network architectures, server architectures, operating systems, sharing strategies, storage subsystems, etc.

One particular need is the capability for large-scale, heterogeneous clusters based on open systems strategies to achieve failure detection and consensus in a scalable fashion. Applications capable of self-healing, perhaps with checkpointing, process migration, etc., require such services so that jobs involving many resources in a large-scale system can come to a consensus on the state of the system as nodes dynamically exit and enter operational status in the system. Classical group communications are inappropriate due to their inherent limits in scalability, and proprietary vendor solutions do not support the heterogeneous nature of the system.

A particularly promising approach towards this goal leverages the notion of gossip communication. Random gossiping, which is based on the individual exchange of liveliness information between nodes, has been investigated as a possible failure-detection approach for scalable heterogeneous systems [1-3]. This interest is due to several advantages of gossip-style failure detectors. For example, gossiping is resilient and does not critically depend upon any single node or message. Moreover, such protocols make minimal assumptions about the characteristics of the networks and hosts, and hold the potential to scale with system size.

In this paper, experimental results from the implementation of a new failure detection and consensus service based on gossiping are presented and analyzed. Results with several forms of gossiping are included and compared, based on random and round-robin intercommunication patterns, and a single-level or *flat* scheme is compared with a hierarchical or *layered* scheme. In the next section, background is briefly provided on concepts in gossip-based failure detection and consensus. Afterwards, experiments and results are presented from a heterogeneous PC cluster, analyses are rendered, conclusions are drawn, and directions for future research are enumerated.

2. Background

This section briefly describes the gossip protocols [1-3] and the consensus algorithm [3] incorporated in the implementation of the failure detection with consensus service developed in this project. In addition to basic definitions for time intervals in a gossip scheme, an overview is provided on random and deterministic gossip protocols, flat and layered gossip schemes, and an algorithm for consensus.

Three of the key parameters involved in the failure detection and consensus service are the gossip time, the cleanup time, and the consensus time. Gossip time, or

T_{gossip}, is the time interval between two consecutive gossip messages. Cleanup time, or $T_{cleanup}$, is the time interval after which a node is suspected to have failed. Finally, consensus time, or $T_{consensus}$, is the time interval after which consensus is reached about a failed node. The first two are input parameters configured for a particular gossip-based failure detection system. The cleanup time is some multiple of the gossip time, where the time required for information to reach all other nodes sets the lower bound for $T_{cleanup}$ (referred to herein as *minimum* $T_{cleanup}$). When both values are relatively small, the system is more quickly responsive to changes in node liveliness. When they are relatively large, response is slower but resource utilization decreases. The third parameter is a performance metric determining how quickly failures are detected and consensus is reached.

2.1. Random and Deterministic Gossip Protocols

Gossip is a scalable means of disseminating information in a distributed system. Due to its distributed nature, gossip is resilient to multiple failures in the system. Gossiping can be broadly classified into two categories: *random* and *deterministic*. In random gossiping, each node gossips to a randomly selected destination every T_{gossip} seconds. The random form of gossiping, also known as basic gossip, was first studied in [1]. While having the benefit of simplicity, the nature of random gossip implies that redundant communication will occur whereby state information is received that has already been received before. To address this limitation and others, deterministic algorithms define a predetermined communication pattern among the gossiping nodes. More efficient forms of gossip, namely round-robin (RR) and binary round-robin (BRR), were first studied in [3]. These two protocols are summarized and compared in Table 1 in terms of minimum $T_{cleanup}$ value, the relationship between source and destination nodes, the round (r), and the number of nodes in the system (n).

Table 1. Characteristics of deterministic gossip protocols.

Protocol	Destination ID	Minimum $T_{cleanup}$
Round-robin	Destination ID = Source ID + r, $1 \le r \le n$	$T_{cleanup} \ge \alpha \times T_{gossip}$, where $\frac{\alpha(\alpha-1)}{2}+1 \le n \le \frac{\alpha(\alpha+1)}{2}+1$
Binary round-robin	Destination ID = Source ID + 2^{r-1}, $1 \le r \le \log_2(n)$	$T_{cleanup} \ge (\log_2 n) \times T_{gossip}$

Basic Gossip

With the basic or random gossip, every T_{gossip} seconds each node randomly selects another node and transmits a heartbeat list and suspect matrix. There is no upper bound on the time it takes for a quantum of information to reach all other nodes. In addition, redundant communication typically occurs hence wasting bandwidth. Conversely, the basic protocol is resilient to network and host failures and is easy to implement.

Round-Robin (RR) Gossip

In the basic gossip protocol it is possible for a node not to receive a gossip for a long enough period of time that false failure detections can occur. The RR protocol is a deterministic gossip protocol aimed at reducing both redundant communication and false failure detections. In the RR gossip protocol, gossiping takes place in definite rounds every T_{gossip} seconds. Every round, each node will receive and send a single gossip message. Using this protocol, the number of rounds needed to establish one-to-one communication with every other node in the system is $n-1$, where n is the number of nodes in the system.

The RR gossip protocol guarantees that all nodes will receive an arbitrary node's updated heartbeat value in a bounded time. This trait allows us to set a lower limit to the value of $T_{cleanup}$. While the RR protocol reduces $T_{cleanup}$ by adopting a deterministic approach, it eliminates some but not all redundant communication.

Binary Round-Robin (BRR) Gossip

The communication redundancy inherent in the RR protocol is eliminated in the BRR gossiping protocol. If the number of nodes in a system is a power of two, by doubling the number of nodes that receive the state information every T_{gossip}, redundant communication can be completely eliminated. This observation is leveraged in BRR, however the simplest form of the BRR protocol works only for a system size that is a power of two and extensions are required for incremental changes in the system size.

2.2. The Consensus Algorithm

Each node maintains three data structures, a gossip list, a suspect vector and a suspect matrix. These three structures together represent the perceived state of each node within a system. The gossip list is a vector of size *n*, where *n* is the system size. Element *j* of the gossip list on node *i* contains the number of T_{gossip} intervals, termed the heartbeat value, since node *i* has received a gossip from node *j*. If the value in element *j* is greater than $T_{cleanup}$, then the corresponding element of the suspect vector is set to '1', otherwise it remains '0' indicating a healthy node.

The suspect vectors of all *n* nodes are joined together to form a suspect matrix of size $n \times n$. From each node a gossip is sent every T_{gossip} seconds containing the suspect matrix. On receipt of a gossip message, the state view of a node is updated by merging the data structures as explained in [3]. Consensus is reached on the state of node *j* if each element in column *j* of the suspect matrix representing unsuspected nodes contains a '1'. When a node detects that consensus has been reached it sends a broadcast message to all nodes in the system informing them of the event.

2.3. Layered Gossiping

The distributed consensus algorithm works efficiently for small systems. However, the size of the suspect matrix and the gossip list dramatically grows with the system size leading to both implementation problems and an increase in communication and processing overhead. Also, due to the increase in the minimum value of $T_{cleanup}$, failure detection time also increases dramatically. These problems can be addressed by layering groups of clusters into a hierarchical system.

Using the layered gossiping protocol, the total number of nodes is divided into groups or clusters in a hierarchy of two or more levels or layers, typically chosen to fit the network topology. In this setup, each node gossips only within its local group. One or more higher layers handle gossiping between the groups. In the case of a two-layer system, nodes in the lower layer take turns to participate in the upper-layer gossip. When consensus is reached within a lower-layer group, this information is broadcast to all nodes in all groups. An example of a two-layer approach is shown in Figure 1.

Layering groups of systems in this manner also facilitates increased heterogeneity in the system by grouping similar nodes together. The separate layers might have different gossiping parameters, as network partition failures are not as frequent as individual failures. Once a node fails, the time taken to reach consensus is largely independent of the system size and depends primarily on the size of the group in which the failure occurred, since consensus is achieved within the group and then propagated to the whole system.

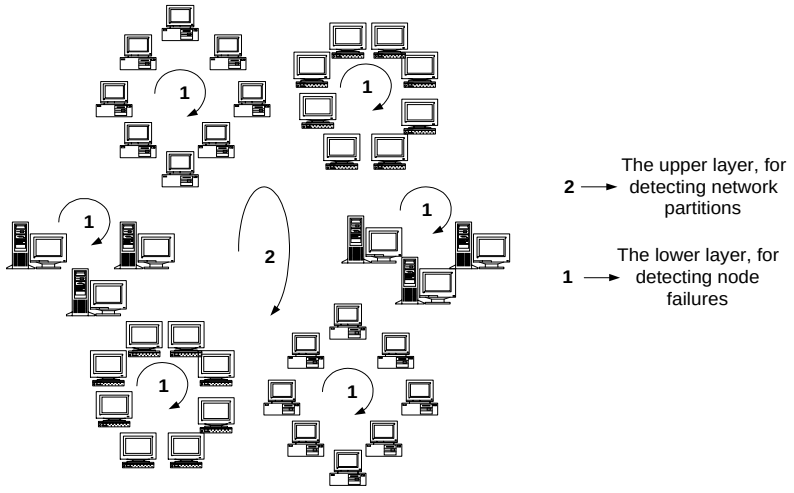

Figure 1. Layered implementation of consensus in a heterogeneous system.

3. Experiments and Results

A series of experiments was conducted using CARRIER (the Cluster Array for Interconnect Evaluation and Research), a heterogeneous PC cluster located in the HCS Research Lab, as a testbed. CARRIER is comprised of over one-hundred SMP and uniprocessor computers that feature a control network of switched Fast Ethernet and a variety of high-speed data networks including SCI, Myrinet, cLAN, and Gigabit Ethernet. The gossip-style

failure detection and consensus service was implemented in CARRIER as a standalone daemon on each of the nodes communicating over the control network using the UDP transport. All of the nodes execute the Redhat Linux V6.1 or V6.2 operating system with a 2.2 kernel.

At the beginning of each experiment, one node is selected randomly to be the faulty node. The faulty node stops gossiping at a fixed time between 0 and $1000 \times T_{gossip}$ seconds. The remaining nodes reach consensus on the faulty node and the consensus time is recorded. Each of the results presented represents the average of five repeated cases, each choosing a different node to stop responding at a different time. A value of 10ms for T_{gossip} is used for all experiments.

3.1. Flat gossiping

A careful choice of the $T_{cleanup}$ parameter is necessary to achieve a consensus time that is as small as possible. In this first set of experiments, relationships between cleanup time, consensus time, and system size are explored on a system with flat gossiping.

The effect of $T_{cleanup}$ on the consensus time for a 16-node system is shown in Figure 2. It is noted that consensus time decreases as $T_{cleanup}$ decreases, back to a minimum cleanup time below which true consensus cannot be reached. If a value for $T_{cleanup}$ were to be selected below the minimum, then false failure detections will increase and make consensus impossible. The minimum cleanup time is proportional to the number of rounds required to disseminate gossip messages to all the members in the system. The BRR protocol requires the least number of rounds to disseminate gossip messages and hence provides the lowest minimum $T_{cleanup}$. By contrast, the basic protocol requires the largest number of rounds and hence has the highest minimum $T_{cleanup}$. For values of cleanup time higher than the minimum, consensus time scales linearly with all three protocols and they share common performance characteristics.

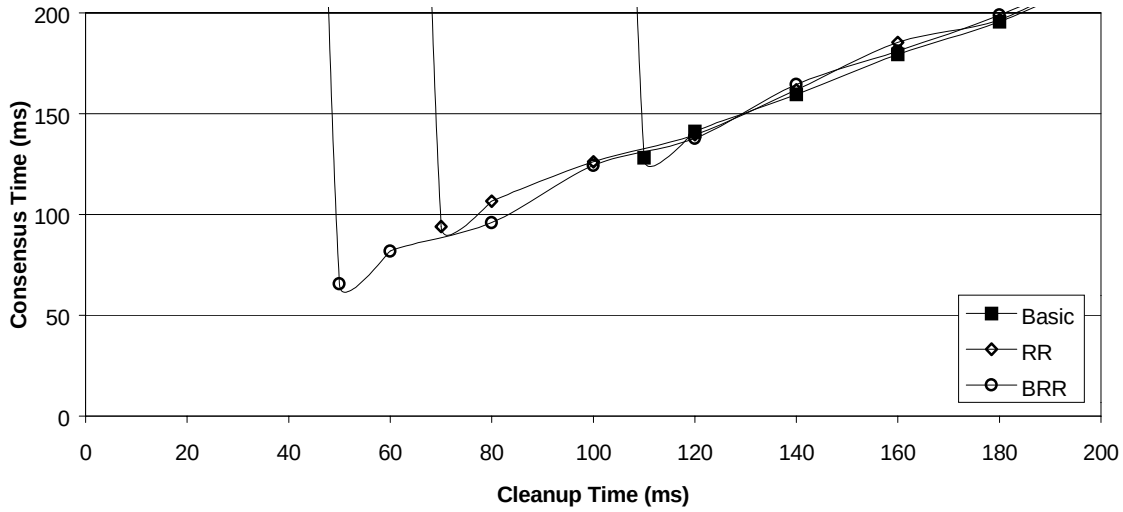

Figure 2. Impact of cleanup time on consensus time in a 16-node flat system.

Having determined the minimum value of cleanup time with each protocol on a given system, the next step is to ascertain how this value scales for different system sizes. Figure 3 shows the variation of minimum $T_{cleanup}$ when the system size is varied from 8 to 96 nodes. Since the BRR protocol requires system sizes that are a power of two, its results extend only to a system size of 64 nodes in this study. With each of the protocols it is observed that the minimum cleanup time increases in a piece-wise linear fashion with the increase in the number of nodes. An interesting observation is the crossover between basic and RR at a system size of 64. This behavior is attributed to lack of synchronization between nodes. Since no attempt was made to synchronize the participating nodes, the round-robin communication pattern deviates from the ideal with increase in the number of nodes.

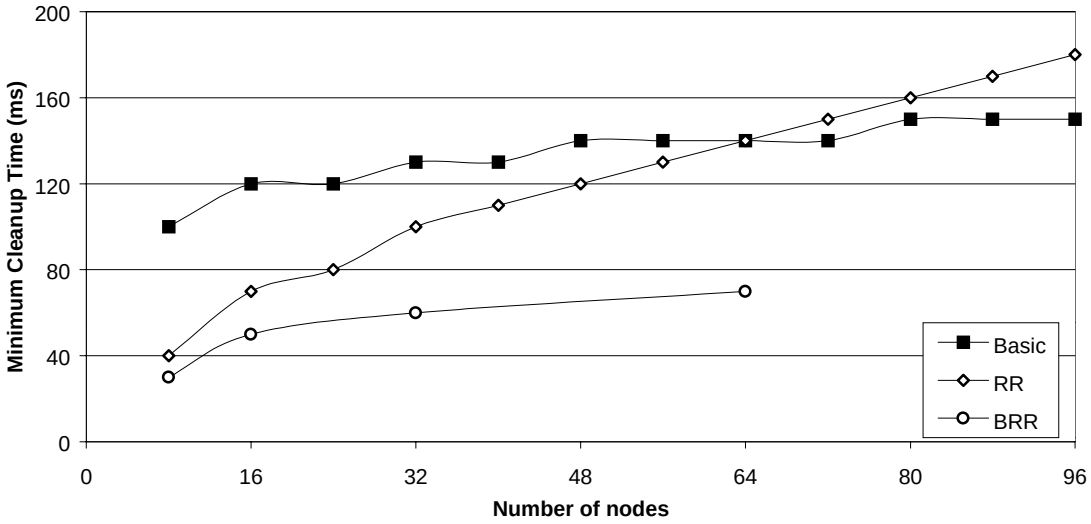

Figure 3. Relationship between minimum cleanup time and system size.

In Figure 4, the best consensus times for different system sizes are presented. In doing so, in each case the lowest possible consensus time is achieved by setting the $T_{cleanup}$ parameter to its minimum value for that system size. The results indicate that the basic protocol outperforms RR for system sizes larger than 72 for the same reasons as explained above. On average, all three protocols exhibit a linear scalability in consensus time versus system size.

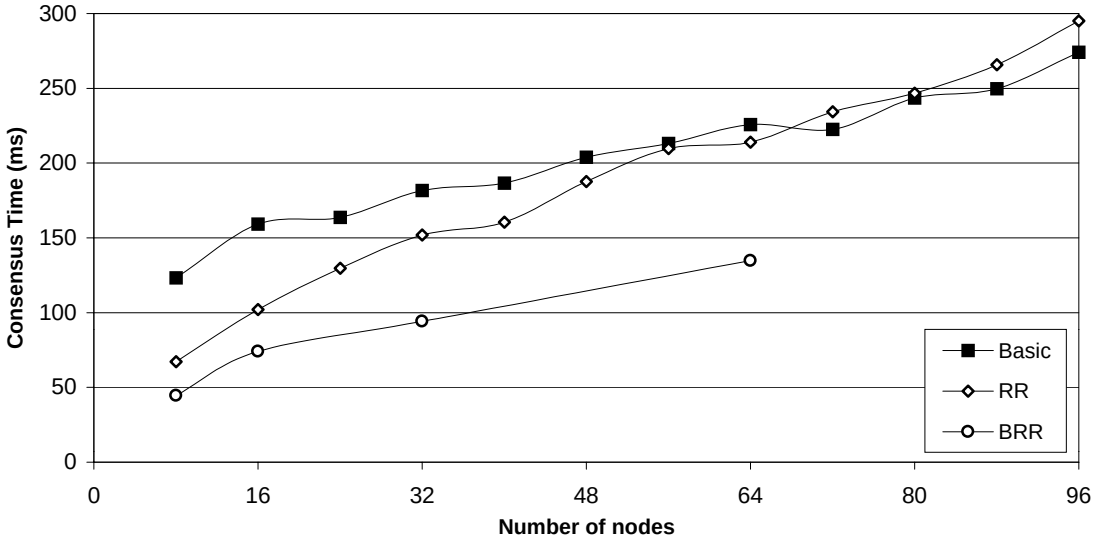

Figure 4. Scalability of consensus time in a flat system.

3.2. Layered gossiping

In this second set of experiments, we divide the system into a two-layer hierarchy with the lower layer (L1) employing the RR gossiping protocol and the upper layer (L2) using the basic protocol. All of the consensus times presented are obtained by setting $T_{cleanup}$ to the lowest possible value for that system size unless otherwise noted.

In the first experiment in this set, the number of nodes in the layered system is varied and consensus time is measured using several different group sizes. As the results in Figure 5 indicate, the consensus time is observed to be independent of system size, but shifts upward with an increase in the group size. In layered gossip, consensus on a failed node within a group needs to be reached only within that group and hence is independent of system size.

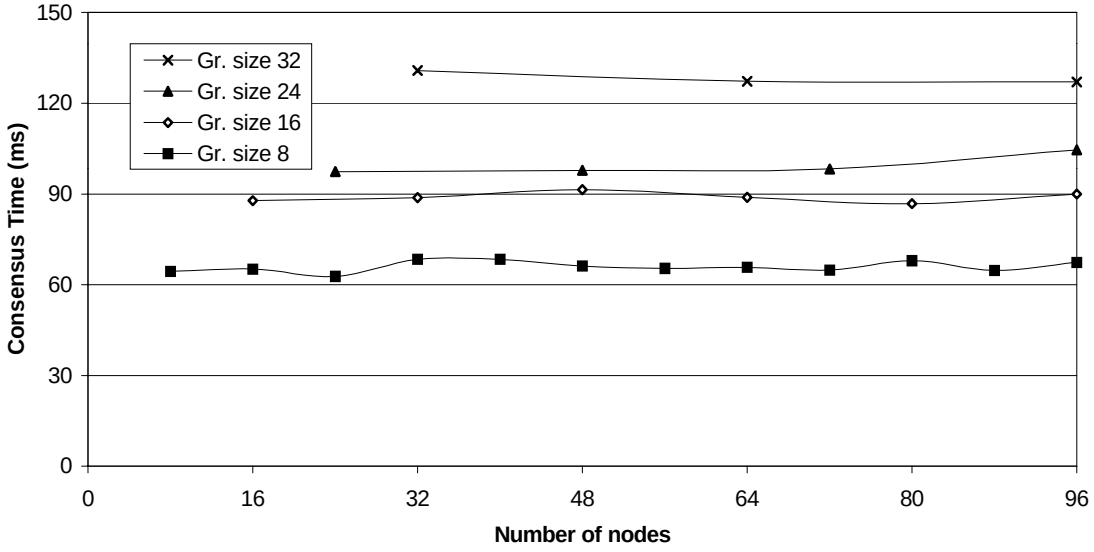

Figure 5. Scalability of consensus time in a layered system for several group sizes (where L1 is RR and L2 is basic).

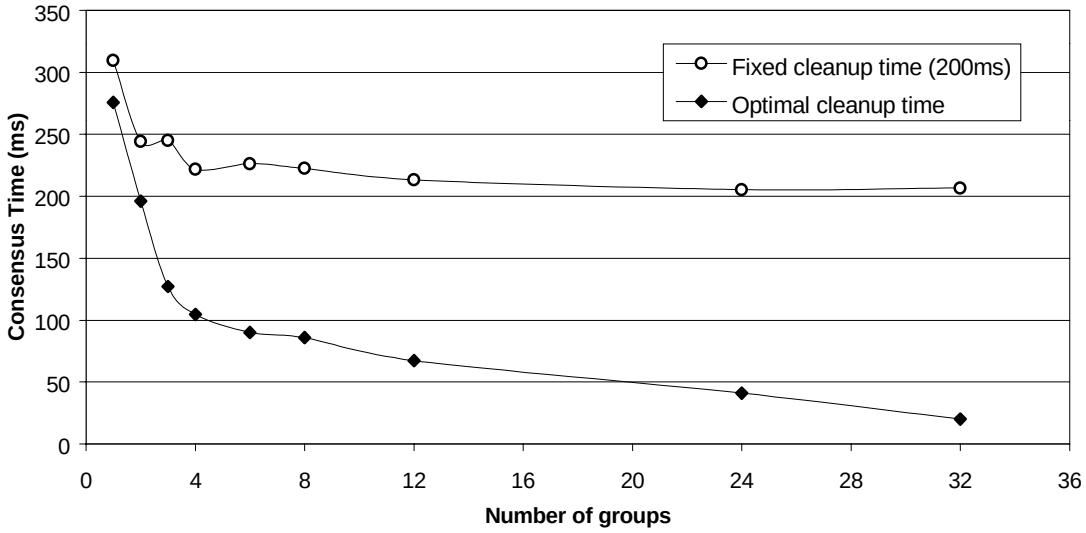

Figure 6. Impact of number of groups on consensus time in a layered system of 96 nodes.

In Figure 6, the consensus time is measured for a varying number of groups while keeping the system size fixed at 96 nodes. When a fixed value of $T_{cleanup}$ is used (i.e. 200ms in this case), an increase in the number of groups exhibits diminishing benefits in consensus time. By contrast, when an optimal $T_{cleanup}$ value is used for each group size, the consensus time decreases substantially with an increase in the number of groups. In terms of consensus time, the optimal group size is determined to be the smallest, however other issues may dictate otherwise. Too small a group size can cause a lack of tolerance within the group when faced with several faults. Moreover, a very small group size would mean a large number of groups, which would increase resource utilization (e.g. network utilization, processor utilization) for large system sizes. Thus, a compromise must be reached between small and large group sizes to balance the best consensus time with the best group reliability and resource utilization. Intuitively, in a two-layer system this balance might be expected to favor a group size on or about the square-root of the system size depending upon the topology of the network and the extent of resource utilization.

Finally, Figure 7 presents a scalability comparison of flat and layered gossiping. In this case, flat gossiping uses RR and layered gossiping is set up as in the previous experiments. The group size for layered gossip is set to eight. It is observed that layered gossip scales well compared to flat gossip, with the consensus time for a 96-node layered system being approximately 25% that of a comparable flat system. On average, the flat system exhibits a linear scalability, while the layered system is virtually constant and thereby ideal in terms of consensus time versus system size. As previously examined, the behavior of layered gossip remains constant across system size but shifts upwards or downwards with an increase or decrease in group size, respectively.

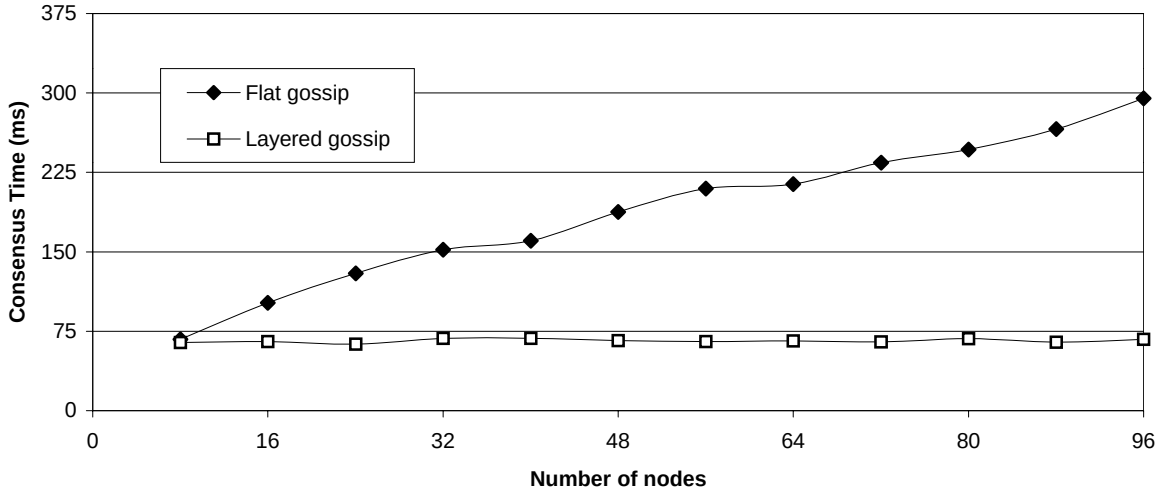

Figure 7. Comparison of scalability of consensus time in layered and flat systems.

4. Conclusions

In this paper, the performance of several gossiping alternatives is examined using a new implementation of a gossip-based failure detection and consensus service on a heterogeneous cluster of Linux PCs. With flat gossiping, consensus time was found to scale in a linear fashion with system size. For system sizes larger than 72, the basic protocol outperforms RR because protocols such as RR and BRR need accurate clock synchronization between nodes for optimal performance. In this work no effort was made to synchronize nodes, thus exposing an inherent weakness in patterned gossiping. In contrast with flat gossiping, layered gossiping was found to exhibit superior scalability with consensus times independent of system size. Moreover, consensus time for layered gossiping was found to substantially improve with decreases in the group size. However, very small groups create other challenges in terms of processor and network resource utilization and group reliability issues. These issues, along with desired consensus time, will dictate the best choice of group size.

Further research will concentrate on determining the resource utilization of various gossip methodologies. Key elements in this direction will include network utilization, processor utilization, and operating system overhead. Analytical models for consensus time and resource utilization also need to be developed to permit performance and scalability projections for increases in system size and heterogeneity beyond the capabilities of the testbed.

The issue of fault-tolerant, distributed clock synchronization also needs to be addressed in support of gossip protocols based on round-robin scheduling. An efficient solution to this problem may use gossiping itself to distribute the time stamps of the nodes involved. Such a system would provide several benefits, such as the ability to better support deterministic protocols for larger system sizes, simplification and streamlining of a scheme for the insertion of new nodes, and as a bookmark for a fault-recovery journaling scheme.

Acknowledgements

The support provided by Sandia National Labs on contract LG-9271 is acknowledged and appreciated, as are equipment grants from Nortel Networks, Intel, and Dell that made this work possible.

References

1. Van Renesse, R., Minsky, R., and Heyden, M., "A Gossip-style Failure Detection Service," *Proc. of IFP Intnl. Conf.*

on Distributed Systems Platforms and Open Distributed Processing Middleware '98, Lake District, England, September 15-18, 1998.

2. Burns, M., George, A., and Wallace, B., "Simulative Performance Analysis of Gossip Failure Detection for Scalable Distributed Systems," *Cluster Computing*, Vol. 2, No. 3, 1999, pp. 207-217.

3. Ranganathan, S., George, A., Todd, R., and Chidester, M., "Gossip-Style Failure Detection and Distributed Consensus for Scalable Heterogeneous Clusters," *Cluster Computing*, accepted and in press.

KRISHNAKANTH V. SISTLA received a B.Tech degree from the Indian Institute of Technology, Madras, India in 1999 and is a master's candidate in Electrical and Computer Engineering at the University of Florida. His research interests include fault tolerance, high-performance computer architectures, and high-performance networks. He can be reached at sistla@hcs.ufl.edu

ALAN D. GEORGE is an Associate Professor of Electrical and Computer Engineering at the University of Florida, and Director & Founder of the HCS Research Lab. He received the BS degree in Computer Science and the MS in Computer-Electrical Engineering from the Univ. of Central Florida, and the Ph.D. in Computer Science from Florida State Univ. Dr. George's research interests are in high-performance networks and architectures for parallel, distributed, and fault-tolerant systems and applications. He can be reached at george@hcs.ufl.edu.

ROBERT W. TODD received a BS degree in Electrical Engineering in 1995 and an MS degree in Electrical Engineering in 1996, both from Florida State University. His research interests include active networks and reconfigurable computing. He is currently a doctoral candidate in Electrical and Computer Engineering at the University of Florida and can be reached at todd@hcs.ufl.edu.

RAGHUKUL TILAK received a BE degree from the Birla Institute of Technology, Ranchi, India in 1997 and is a master's candidate in Electrical and Computer Engineering at the University of Florida. His research interests include cluster computing, high-performance computer architectures, and high-performance networks. He can be reached at tilak@hcs.ufl.edu.