Modeling and Validating the Performance of Atomic Broadcast Algorithms in High Latency Networks

Richard Ekwall and André Schiper

École Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland
{nilsrichard.ekwall,andre.schiper}@epfl.ch

Abstract. The performance of consensus and atomic broadcast algorithms using failure detectors is often affected by a trade-off between the number of communication steps and the number of messages needed to reach a decision.
In this paper, we model the performance of three consensus and atomic broadcast algorithms using failure detectors in the oft-neglected setting of wide area networks and validate this model by experimentally evaluating the algorithms in several different setups.

1 Introduction

1.1 Context:

Chandra and Toueg introduced the concept of failure detectors in [5]. Since then, several atomic broadcast [7] and consensus [5, 11] algorithms based on failure detectors have been published.

The performance of these algorithms is affected by a trade-off between the number of communication steps and the number of messages needed to reach a decision. Some algorithms reach decisions in few communication steps but require more messages to do so. Others save messages at the expense of additional communication steps (to diffuse the decision to all processes in the system for example). This trade-off is heavily influenced by the message transmission and processing times. When deploying an atomic broadcast algorithm, the user must take these factors into account to choose the algorithm that is best adapted for the given network environment.

The performance of these algorithms has been evaluated in several environments, both real and simulated [15]. However, these evaluations are limited to a symmetrical setup: all processes are on the same local area network and have identical peer-to-peer round-trip times. Furthermore, they only consider low round-trip times between processes (and thus high message processing costs), which is favorable to algorithms which limit the number of sent messages, at the expense of additional communication steps.

1.2 Contributions:

In this paper, we model and evaluate the performance of three atomic broadcast algorithms using failure detectors with three different communication patterns (the first based on reduction to a centralized consensus algorithm [5], the second based on reduction to a decentralized consensus algorithm [11] and the third one, a ring based algorithm [7]) in wide area networks. We specifically focus on the case of a system with three processes, — i.e., supporting one failure — where either (i) all three processes are on different locations and (ii) the three processes are on two locations only (and thus one of the locations hosts two processes). The system with three processes is interesting as it (1) has no single point of failure, (2) represents the case in which the group communication algorithms reach their best performance and (3) can be well modeled analytically. The algorithms are experimentally evaluated with a large variation in link latency (e.g., round-trip times ranging from 4 to 300 ms).

We propose a simple model of the wide area network to analytically predict the performance of the three algorithms. The experimental evaluation confirms that the model correctly predicts the performance for average system loads and for all round-trip times that we considered.

The experimental evaluation of the algorithms leads to the following conclusions. First, the number of communication steps of the algorithms is the predominant factor in wide area networks, whether the round-trip time is high (300 ms) or, more surprisingly (since message processing times are no longer negligible), if it is low (4 ms). The performance ranking of the three algorithms is the same in all the wide area networks considered, despite the two orders of magnitude difference between the smallest and largest round trip times. Second, the performance of each of the algorithms heavily depends on setup issues that are orthogonal to the algorithm (typically the choice of the process that starts each iteration of the algorithm, which can be always the same process, or which can shift from one process to another at each iteration). These setup issues also determine the maximum achievable throughput. Finally, measurements referenced in the paper show, as expected, that the performance ranking of the three algorithms is fundamentally different in a wide area network than in a local area network.

2 Motivation and Related Work

In [8], the authors show that consensus cannot be solved in an asynchronous system with a single crash failure. Several extensions to the asynchronous model, such as failure detectors [5], have circumvented this impossibility and agreement algorithms [5, 11, 7] have been developed in this extended model. The performance of these atomic broadcast algorithms is evaluated in different ways. Usually, the formal presentation of the agreement algorithms is accompanied by analytical bounds on the number of messages and communication steps that are needed to solve the problem [5, 11, 16]. This coarse-grained evaluation of the performance of the algorithms is however not sufficiently representative of the situation in a real environment.

To get a more accurate estimation of the performance of the atomic broadcast algorithms, they have often been evaluated in local area networks [7], simulated in a symmetrical environment where all links between processes have identical round-trip times [15] or evaluated in hybrid models that introduce artificial delays to simulate wide area networks [16]. Although these performance evaluations do provide a representative estimate of the performance of atomic broadcast on a local area network, they cannot be used to extrapolate the performance of the algorithms on a wide area network, where the ratio between communication and processing costs is completely different. Furthermore, evaluating the performance of atomic broadcast on wide area networks is not only of theoretical interest. As [10] shows, it is feasible to use atomic broadcast as a service to provide consistent data replication on wide area networks. In this paper, we model the performance of these algorithms *and* validate this analysis by experimentally evaluating the algorithms in wide area networks.

We now discuss the central trade-off that explains the impact of network latency on the performance of atomic broadcast algorithms.

2.1 The trade-off between number of messages and communication steps:

The processes executing the atomic broadcast algorithms that we consider in this paper communicate with each other to agree on a common message delivery sequence. To do so, they need to exchange a minimum number of messages in a number of communication steps. There is here a trade-off on the number of communication steps and the number of sent messages. Usually, a higher number of messages enables the algorithm to reach a decision in fewer communication steps and vice-versa.

Each communication step has a cost. Indeed, each additional communication step induces a delay on the solution to the problem. This cost is typically low in a local area network, whereas it increases with the latency in a wide area network.

Sending messages also has a cost. Whenever a message is sent, it has to be handled by the system. This handling includes costs related to algorithmic computations on its content, serialization (i.e. transforming the message to and from an array of bytes that is sent on the network) and bandwidth used for the transmission.

These costs characterize the trade-off between the number of messages sent and communication steps needed by the algorithm. If a communication step costs nothing, then the algorithm that sends the least number of messages performs the best. If, on the other hand, a communication step is very expensive, the algorithm that sends most messages (and thus saves on the number of communication steps) has the best performance. In this paper, several network latencies are studied to evaluate their impact on this trade-off.

2.2 Related work:

In [2], the authors study the influence of network loss on the performance of two atomic broadcast algorithms in a wide area network. To do this, the authors combine experimental results obtained on a real network with an emulation of the atomic broadcast algorithms. The scope of the work in [2] is different from ours: they evaluate the impact that message loss has on the performance of atomic broadcast algorithms whereas we model and evaluate the impact of *network latency* on the relative performance of different algorithms.

Bakr and Keidar evaluate the duration of a communication round on the Internet in [3]. Their work focuses on the running time of four distributed algorithms with different message exchange patterns, and in particular, the effect of message loss on these algorithms. Their experiments are run on a large number of hosts (10) and the algorithms that they examine do not allow messages to be lost (i.e. an algorithm waits until it has received *all* messages it is expecting). The scope of [3] is similar to ours in that they analyze the relative performance of algorithms with different communication patterns on a wide area network. However, their algorithms are *not* representative of failure detector based atomic broadcast algorithms. Indeed, in the three algorithms we consider, processes never need to wait for messages from *all* the other processes. Thus, if messages from one process are delayed because of a high-latency link, it does not necessarily affect the performance of the atomic broadcast algorithm (whereas it would in [3]).

In [16], an atomic broadcast algorithm that is specifically targeted towards high latency networks is presented. The authors also evaluate the performance of the algorithm in a local area network with added artificial delays (to simulate the high latency links). The artificial delay is however not sufficient to adequately represent the network links of a wide area network. Indeed, such links are also characterized by a lower bandwidth than local area network links. In our performance measurements, we show that in some cases, the low bandwidth of the wide area links strongly limits the performance of the algorithms that are considered.

Several other papers (e.g. [15, 7, 13, 9]) have studied the performance of atomic broadcast algorithms, but these papers either study the performance of the algorithms in a local area network or through simulation.

3 System model and algorithms

We consider an asynchronous system of n processes $p_0, \ldots, p_{n-1}$. The processes communicate by message passing over reliable channels and at most f processes may fail by crashing (i.e. we do not consider Byzantine faults). The system is augmented with unreliable failure detectors [5, 7].

In the following paragraphs, we informally present reliable broadcast, consensus and atomic broadcast [5]. Reliable broadcast and consensus are building blocks for solving atomic broadcast in two of the atomic broadcast algorithms that we consider.

3.1 Reliable broadcast, consensus and atomic broadcast

In the *reliable broadcast* problem, defined by the primitives *rbroadcast* and *rdeliver*, all processes need to agree on a common set of delivered messages. In this paper, we consider the reliable broadcast algorithm presented in [5], which requires $O(n^2)$ messages and a single communication step to *rbroadcast* and *rdeliver* a message m.

Informally, in the *consensus* problem, defined by the two primitives *propose* and *decide*, a group of processes have to agree on a common decision. In this paper, we consider two consensus algorithms that use the $\Diamond\mathcal{S}$ failure detector [5] (presented in Section 3.2).

In the atomic broadcast problem, defined by the two primitives *abroadcast* and *adeliver*, a set of processes have to agree on a common total order delivery of a set of messages. It is a generalization of the reliable broadcast problem with an additional ordering constraint. In this paper, we consider two atomic broadcast algorithms which are described in Section 3.3.

3.2 Two consensus algorithms

The first consensus algorithm, proposed by Chandra and Toueg [5] and noted *CT*, is a centralized algorithm that requires 3 communication steps, $O(n)$ messages and 1 reliable broadcast for all processes to reach a decision in *good* runs (i.e. runs without any crashes or wrong suspicions). The behavior of the CT algorithm in good runs is detailed in [6].

The second consensus algorithm, proposed by Mostéfaoui and Raynal [11] and noted *MR*, is a decentralized algorithm that requires 2 communication steps and $O(n^2)$ messages[1] for all processes to reach a decision in good runs. The behavior of the MR consensus algorithm in good runs and in a system with $n = 3$ processes is detailed in [6].

On the choice of a coordinator: Both the CT and MR consensus algorithms use a *coordinator* that proposes the value that is to be decided upon. This coordinator can be any process in the system, as long as it can be deterministically chosen by all processes (based only on information that is locally held by each process). In the analytical and experimental evaluations of these algorithms, we examine how the choice of the first coordinator influences the performance of the algorithms. We also study the case where the first coordinator changes between instance number k of consensus and the next instance $k + 1$.

3.3 Two atomic broadcast algorithms

Chandra-Toueg atomic broadcast [5]: The Chandra and Toueg atomic broadcast algorithm requires at least one reliable broadcast and a consensus execution for all processes to *abroadcast* and *adeliver* messages.

[1] The MR consensus algorithm does not use reliable broadcast as a building block. Instead, reliable diffusion of the decision is ensured by an ad-hoc protocol using n^2 messages.

Whenever a message m is *abroadcast*, it is reliably broadcast to all processes (first communication step in good runs). The processes then execute consensus on the messages that haven't been *adelivered* yet (using the CT or MR algorithm in our case). If m is in the decision of consensus, then m is *adelivered*. A waiting period happens if a consensus execution is already in progress and therefore prevents m from being proposed at once for a new consensus.

Token using an unreliable failure detector [7]: The token-based atomic broadcast algorithm (noted *TokenFD*) solves atomic broadcast by using an unreliable failure detector noted $\mathcal{R}$ and by passing a token among the processes in the system. It requires three communication steps in a system with $n = 3$ processes and $O(n)$ messages for all processes to *abroadcast* and *adeliver* messages.

Whenever a message m is *abroadcast*, it is sent to all processes (first communication step). Message m is then added to the token that circulates among the processes. After the second communication step, m is *adelivered* by the token-holder which sends an update to all other processes about this delivery (third communication step). Again, a waiting period only happens if the token is already being sent on the network (without m) and therefore prevents m from being ordered immediately.

4 Performance metrics and workloads

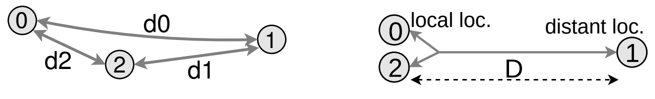
The following paragraphs describe the benchmarks (i.e., the performance metrics and the workloads) that were used to experimentally evaluate the performance of the three atomic broadcast algorithms (reduction to CT consensus; reduction to MR consensus; TokenFD algorithm). The benchmarks in [15, 7] are similar to the ones we use here.

Performance metric – latency vs. throughput: The performance metric that was used to evaluate the algorithms is the latency of atomic broadcast. For a single atomic broadcast, the latency L is defined as follows. Let t_a be the time at which the *abroadcast*(m) event occurred and let t_i be the time at which *adeliver*(m) occurred on process $p_i \in \{p_0, \ldots, p_{n-1}\}$. The latency L is then defined as $L \overset{def}{=} (\frac{1}{n}\sum_{i=0}^{n-1} t_i) - t_a$. In our experimental performance evaluation, the mean for L is computed over many messages and for several executions. 95% confidence intervals are shown for all the results.

Workloads: The workload specifies how the *abroadcast* events are generated. We chose a simple symmetric workload where all processes send atomic broadcast messages (without any payload) at the same constant rate and the *abroadcast* events follow a Poisson distribution. The global rate of atomic broadcasts is called the *throughput* T.

Furthermore, we only consider the system in a stationary state (when the rate of *abroadcast* messages is equal to the rate of *adelivered* messages) and we only evaluate the performance of the algorithms in good runs (i.e., without any process failures or wrong suspicions).

We specifically focus on the case of a system with three processes, supporting one failure. This system size might seem small. However, atomic broadcast



(a) Three locations (b) Two locations

Fig. 1. Theoretical model of a wide area network

provides strong consistency guarantees (that can be used to implement active replication for example [12]) and is limited to relatively small degrees of replication. If a large degree of replication is needed, then alternatives that provide weaker consistency should be considered [1].

5 Modeling the performance of the algorithms

This section discusses the analytical performance evaluation of the two atomic broadcast (and consensus) algorithms in a wide area network. We present the two wide area network models that are considered. Due to lack of space, the derivation of the latencies of the algorithms in this model is not further presented, but can be found in [6]. The predictions of the model (i.e. the numerical applications of the model to the four experimental setups) are shown alongside the experimental evaluation of the algorithms in Section 6.

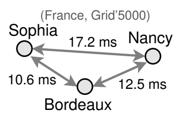
Figure 1(a) presents the model of a wide area network system with three processes on three different locations. The network latency between location i and location $i+1$ is noted d_i. Without loss of generality, we assume that $d_0 \geq d_1 \geq d_2$. The model is simplified, in the sense that the processing costs of the messages are considered negligible. Furthermore, the model does not take other factors into account, such as the bandwidth of the links or message loss.

Figure 1(b) presents the model of a system with three processes, one of which is on a distant location. The network latency between the distant location and the local location is noted D. The *two-location model* is a special case of the previous model, with $d_0 = d_1 = D$ and $d_2 = 0$.

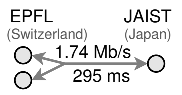
The average latency of the three atomic broadcast algorithms in this model can be found in [6].

6 Experimental performance evaluation

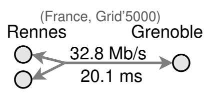
In the following section, the experimental performance of the atomic broadcast algorithms presented in Section 3 are compared. First, we briefly present the evaluation environments that were considered and then the results that were obtained are presented, analyzed and compared to the analytical evaluation of Section 5. The algorithms presented in this paper are all implemented in Java, using the Neko framework [14].



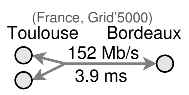
(a) WAN Three Locations



(b) WAN 295

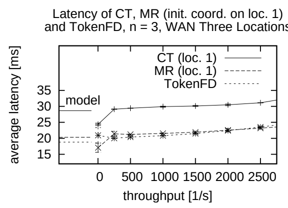


(c) WAN 20.1

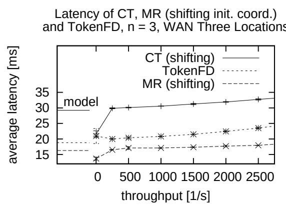


(d) WAN 3.9

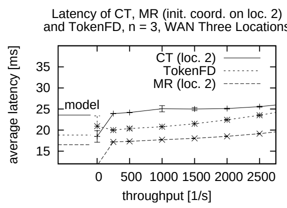
Fig. 2. Wide area network environments in decreasing order of round trip times.



(a) Init. coord. on loc. 1.



(b) Shifting init. coord.



(c) Init. coord. on loc. 2

Fig. 3. Average latency of the three algorithms as a function of the throughput in the WAN Three Locations setting.

6.1 Evaluation environments

Four wide area network environments were used to evaluate the performance of the three atomic broadcast and consensus algorithms (see Fig. 2) All machines run a Linux distribution (2.6.8 to 2.6.12 kernels) and a Sun Java 1.5.0 virtual machine. The following paragraphs describe the different wide area network environments in which the atomic broadcast algorithms are evaluated.

Three-location wide area network: The first evaluation environment (noted WAN Three Locations, Figure 2(a)) is a system with three locations on Grid'5000 [4], a French grid of interconnected clusters designed for the experimental evaluation of distributed and grid computing applications. The round-trip times of the links between the three processes are respectively $2d_0 = 17.2$ ms, $2d_1 = 12.5$ ms and $2d_2 = 10.6$ ms. The observed bandwidth of the three links are respectively 30.1 Mbits/s, 41.4 Mbits/s and 48.7 Mbits/s.

Two-location wide area networks: Three environments were used to evaluate the performance of atomic broadcast on wide area networks with two different locations:

– WAN 295 (Figure 2(b)): The first two-location environment consists of one location in Switzerland and one in Japan. The round-trip time between the locations is $2D = 295$ ms and the bandwidth of the connecting link is 1.74 Mb/s.

– WAN 20.1 and WAN 3.9: The two following environments are systems with both locations on Grid'5000. The WAN 20.1 system (Figure 2(c)) features a round-trip time between locations of $2D = 20.1$ ms and a link bandwidth of 32.8 Mb/s. The WAN 3.9 system (Figure 2(d)) features a round-trip time between locations of $2D = 3.9$ ms and a link bandwidth of 152 Mb/s. The performance characteristics of the three algorithms are similar in WAN 20.1 and WAN 3.9.

Table 1. Latency (in milliseconds) of the three algorithms for a throughput of 1000 msgs/s in the WAN Three Locations setting.

(a) Init. Coord. on location 1

Algorithm	Measured	Modeled	Diff.
CT	29.92	28.66	+4.4%
MR	21.55	20.32	+6.1%
TokenFD	20.81	18.82	+10.6%

(b) Shifting init. coord.

Algorithm	Measured	Modeled	Diff.
CT	30.57	29.24	+4.5%
MR	17.09	16.28	+5.0%
TokenFD	20.81	18.82	+10.6%

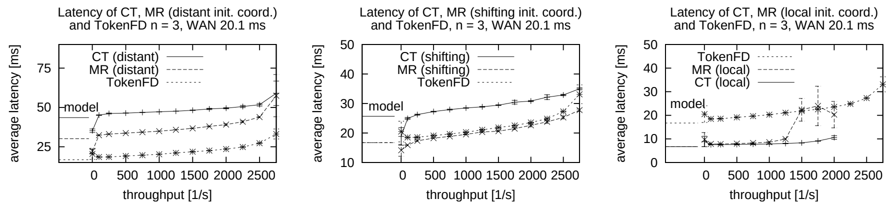
6.2 Validation of the model with the experimental results

We now discuss the validation of the model presented in Section 5 by the experimental evaluation of the three atomic broadcast algorithms. As mentioned in Section 4, the performance graphs present the average latency as a function of the throughput in the system. Furthermore, for the CT and MR consensus algorithms, the results are given for an initial coordinator that is fixed in one location or shifting with each new consensus execution. The TokenFD algorithm has no concept of coordinator and its results are the same for all three settings (they are repeated to give a point of comparison with respect to CT and MR). The modeled performance of the algorithms is shown on the far-left of each graph (noted "model").

In all experimental setups, the measurements confirm the estimations of the model (Figures 3 and 4), especially in the case of moderate throughputs. When the throughput increases, the load on the processors and on the network (which is not modeled) affects the latency of the algorithms (illustrated in particular in Figure 4(c)), which decreases the accuracy of the model.

Furthermore, when the throughput is very low, the measured latencies of CT and MR are lower than what the model predicts. Indeed, our analysis assumes a load in which messages are *abroadcast* often enough that there is always a consensus execution in progress. In the low throughput executions however, there is a pause between the consensus executions. An unordered message that is received during this pause is immediately proposed in a new consensus execution and thus, the *waiting* phase presented in Section 5 does not apply to that message.

Finally, the point that was not predicted by the analytical model is the result for high throughputs when the initial coordinator of CT and MR is on a local location, illustrated by Figure 4(c). Indeed, in this setting, the system never reaches a stationary state given a sufficiently high throughput. The processes on the local location reach consensus decisions very fast without needing any input from the distant location. The updates that are then sent to the distant location saturate the link between both locations (its bandwidth is only 32.8 Mbits/s in WAN 20.1). The process on the distant location thus takes decisions slower than the two local processes and prevents the average latency of atomic broadcast from stabilizing. This problem does not affect the settings with a distant or shifting initial coordinator, since the distant location periodically acts as a consensus



(a) Distant coord. (b) Shifting init. coord. (c) Local coord.

Fig. 4. Average latency of the three algorithms as a function of the throughput in the WAN 20.1 setting.

coordinator, providing a natural flow control. Setup issues, such as the choice of the initial coordinator, thus affect the maximum achievable throughput of the algorithms.

Table 1 presents the latency of the three algorithms in the WAN Three Locations setting, with a throughput of 1000 messages per second. The model predicts the experimental measurements with an error of 5 to 10%.

6.3 Comparing the performance of the three algorithms

WAN Three Locations: The average latency of the three algorithms in the WAN Three Locations environment is presented in Figure 3. TokenFD and MR outperform CT for all locations of the initial coordinator and for all throughputs, due to the additional communication step that is needed by CT. TokenFD and MR perform similarly when the initial MR coordinator is on site 1 (which is the worst-case scenario for MR), whereas MR achieves slightly better latencies than TokenFD for both other initial coordinator locations.

Surprisingly enough, the result of using a shifting initial coordinator in the CT and MR algorithms are opposite: in the case of MR, the latency is lower using a shifting initial coordinator than a fixed initial coordinator on any location, whereas in CT it is higher. The explanation is the following: MR and CT both start a new consensus execution after two communication steps if the coordinator is on a fixed location. If the coordinator shifts, a new execution can start as soon as the next non-coordinator process decides. This is done after *one* communication step in MR (if $n = 3$), but after *three* steps in CT, as explained in Section 5.

WAN 295, WAN 20.1 and WAN 3.9: The average latency of the three atomic broadcast and consensus algorithms in the WAN 20.1 environment is presented in Figures 4 (the WAN 295 and WAN 3.9 environements are presented in [6]). TokenFD has lower latencies than CT and MR when they use a distant initial coordinator (Figure 4(a)), whereas the situation is reversed when the coordinator is initially on a local location (Figure 4(c)). When the initial coordinator shifts at each new consensus execution, MR and TokenFD have similar latencies while CT is slightly slower. Finally, as mentioned earlier, the low bandwidth of the link between both locations prevents MR and CT from reaching stable average latencies when the initial coordinator is on the local locations and the throughput is high.

Communication steps versus number of messages: As expected, the performance results presented above show that *communication steps have the largest impact on performance in wide area networks, whereas the number of sent messages is a key to the performance in a local area network [6]*. The validity of this statement however varies with the round-trip time of the network that is considered. As the network latency decreases, the impact of the additional messages that need to be sent and processed increases. In the case of networks with 3.9 ms or even 20.1 ms round-trip times, this impact is clearly observable.

However, for a given set of parameters, the algorithm with the best performance is generally the same (whether a wide area network with a 3.9 ms round-trip time is considered or one with a 295 ms round-trip time) and it is correctly predicted by the model.

Finally, we also saw that choosing a CT and MR coordinator on the local location (without implementing an additional flow control mechanism) is not necessarily the best solution performance-wise, since the system cannot reach a stationary state as the total throughput increases. Shifting the initial coordinator between locations at each new consensus execution or choosing the TokenFD algorithm results in a natural flow control which enables the system to remain in a stationary state even for high throughputs (at the expense of a higher average *adelivery* latency).

7 Conclusion

This study confirms that the relative performance between the algorithms is fundamentally different between a local area network and a wide area network (even in wide area networks with small round-trip times): in the former case, the number of sent messages (i.e. the number of messages that need to be processed) largely determines the performance of the algorithms, whereas the communication steps have the most impact in the latter case.

Within wide area networks on the other hand, the performance ranking of the three algorithms remains the same, despite the (two order of magnitude) difference in the round-trip time between the smallest and largest wide area networks. Furthermore, this ranking is correctly predicted by our model. The study also showed that algorithms or parameters which provide a natural flow control (such as the TokenFD atomic broadcast algorithm or the Chandra-Toueg and Mostéfaoui-Raynal consensus algorithms with an initial coordinator that shifts between locations at each new consensus) are effective in reaching higher throughputs in wide area networks.

References

1. L. Alvisi and K. Marzullo. Waft: Support for fault-tolerance in wide-area object oriented systems. In *Proc. of the 2nd Information Survivability Workshop – ISW '98*, pages 5–10. IEEE Computer Society Press, October 1998.

2. T. Anker, D. Dolev, G. Greenman, and I. Shnayderman. Evaluating total order algorithms in WAN. In *Proc. International Workshop on Large-Scale Group Communication*, Florence, Italy, October 2003.
3. O. Bakr and I. Keidar. Evaluating the running time of a communication round over the internet. In *Proc. of the 21st ACM Ann. Symp. on Principles of Distributed Computing*, pages 243–252, 2002.
4. F. Cappello, E. Caron, M. Dayde, F. Desprez, E. Jeannot, Y. Jegou, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, and O. Richard. Grid'5000: a large scale, reconfigurable, controlable and monitorable Grid platform. In *Grid'2005 Workshop*, Seattle, USA, November 13-14 2005. IEEE/ACM.
5. T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of ACM*, 43(2):225–267, 1996.
6. R. Ekwall and A. Schiper. Comparing Atomic Broadcast Algorithms in High Latency Networks. Technical Report LSR-REPORT-2006-003, École Polytechnique Fédérale de Lausanne, Switzerland, July 2006.
7. R. Ekwall, A. Schiper, and P. Urbán. Token-based atomic broadcast using unreliable failure detectors. In *Proc. of the 23rd Symposium on Reliable Distributed Systems (SRDS 2004)*, Florianópolis, Brazil, Oct. 2004.
8. M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of ACM*, 32(2):374–382, Apr. 1985.
9. R. Guerraoui, R. R. Levy, B. Pochon, and V. Quéma. High Throughput Total Order Broadcast for Cluster Environments. In *IEEE International Conference on Dependable Systems and Networks (DSN 2006)*, June 2006.
10. Y. Lin, B. Kemme, M. Patiño-Martínez, and R. Jiménez-Peris. Consistent data replication: Is it feasible in WANs?. In *Proc. 11th International Euro-Par Conference*, pages 633–643, Lisbon, Portugal, September 2005.
11. A. Mostefaoui and M. Raynal. Solving Consensus using Chandra-Toueg's Unreliable Failure Detectors: A Synthetic Approach. In *13th. Intl. Symposium on Distributed Computing (DISC'99)*. Springer Verlag, LNCS 1693, September 1999.
12. F. B. Schneider. Replication management using the state-machine approach. In S. Mullender, editor, *Distributed Systems*, ACM Press Books, chapter 7, pages 169–198. Addison-Wesley, second edition, 1993.
13. A. Sousa, J. Pereira, F. Moura, and R. Oliveira. Optimistic Total Order in Wide Area Networks. In *21st IEEE Symp. on Reliable Distributed Systems (SRDS-21)*, pages 190–199, Osaka, Japan, October 2002.
14. P. Urbán, X. Défago, and A. Schiper. Neko: A single environment to simulate and prototype distributed algorithms. *Journal of Information Science and Engineering*, 18(6):981–997, Nov. 2002.
15. P. Urbán, I. Shnayderman, and A. Schiper. Comparison of failure detectors and group membership: Performance study of two atomic broadcast algorithms. In *Proc. of the Int'l Conf. on Dependable Systems and Networks (DSN)*, pages 645–654, June 2003.
16. P. Vicente and L. Rodrigues. An Indulgent Total Order Algorithm with Optimistic Delivery. In *21st IEEE Symp. on Reliable Distributed Systems (SRDS-21)*, pages 92–101, Osaka, Japan, October 2002.