Comparing Atomic Broadcast Algorithms in High Latency Networks

Richard Ekwall
nilsrichard.ekwall@epfl.ch

André Schiper
andre.schiper@epfl.ch

École Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland

Technical Report LSR-REPORT-2006-003

Abstract

Since the introduction of the concept of failure detectors, several consensus and atomic broadcast algorithms based on these detectors have been published. The performance of these algorithms is often 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 an additional communication step to diffuse the decision to all processes in the system. This trade-off is heavily influenced by the network latency and the message processing times and therefore yields fundamentally different results in wide and local area networks.

Performance evaluations of these algorithms, both in simulated or in real environments, have been published. These evaluations often consider a symmetrical setup : all processes are on the same network and have identical peer-to-peer latencies.

In this paper, we model and evaluate the performance of three consensus and atomic broadcast algorithms using failure detectors in several wide area networks. We specifically focus on the case of a system with three processes, located in two or three different locations.

Keywords: Atomic broadcast, wide area network, model, performance evaluation

1 Introduction

1.1 Context

Chandra and Toueg introduced the concept of failure detectors in [4]. Since then, several atomic broadcast [7] and consensus [4, 13, 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 [7, 5] and simulated [18, 19]. 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 comparatively high message processing costs): a setting 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 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 [4], the second based on reduction to a decentralized consensus algorithm [13] 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 algorithms are 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 moderate system loads.

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. Indeed, the performance ranking of the three algorithms is the same in all the wide area networks considered, despite the difference in latency between the smallest (4 ms) and largest (300 ms) round trip time, and is correlated to the number of communication steps. 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, measures mentioned in the Appendix show, as expected, that the performance ranking of the three algorithms is fundamentally different in a wide area network and in a local area network. This is explained by the fact that, in a local area network, the predominant factor is the number of messages, while in a wide area network it is the number of communication steps.

The paper is structured as follows. Section 2 discusses the motivation for evaluating the atomic broadcast algorithms in wide area networks. Sections 3 and 4 present respectively the system model and the performance metrics that are used. The evaluation of the algorithms is then presented in Section 5 (analytical) and Section 6 (experimental). Finally, Section 7 concludes the paper.

2 Motivation and Related Work

2.1 Motivation

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 [4], have circumvented this impossibility and agreement algorithms [4, 13, 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 [4, 13, 20]. 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, 5], simulated in a symmetrical environment where all links between processes have identical round-trip times [18, 19] or evaluated in hybrid models that introduce artificial delays to simulate wide area networks [20].

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 [12] shows, it is feasible to use atomic broadcast as a service to provide consistent data replication on wide area networks.

The following paragraph describes the central trade-off that explains the impact of network latency on the performance of atomic broadcast algorithms.

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 (corresponding to half of the round-trip time between the processes involved in the communication). 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. Additional messages mainly imply additional processing costs. Indeed, 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 [1], 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 [1] is different from ours : they evaluate the impact that message loss has on the performance of atomic broadcast algorithms whereas we evaluate the impact of *network latency* on the relative performance of different algorithms. Furthermore, the performance evaluation in [1] does not take the processing time of messages into account (the results are based on message logs and emulated algorithms). Arguably, this processing time is negligible when a network with large round-trip times is considered (as was the case in [1]), but its importance increases as the round-trip times decrease.

Bakr and Keidar evaluate the duration of a communication round on the Internet in [2]. 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 [2] 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 [2]).

In [20], 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 such as [19, 18, 7, 5, 6, 15, 9] have studied the performance of atomic broadcast algorithms that use failure detectors or properties related to the spontaneous ordering of messages in local area networks. These papers however, either study the performance of the algorithms in a local area network or through simulation. None of these evaluations adequately models the impact of the high latency links in a wide area network on the performance trade-off between the number of messages that are sent and the number of communication steps needed by the agreement algorithm.

3 System model, consensus and atomic broadcast

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). A process that never crashes is said to be *correct*, otherwise it is *faulty*. The system is augmented with unreliable failure detectors [4, 7].

In the following paragraphs, we informally present reliable broadcast, consensus and atomic broadcast (the formal definition of the three problems can be found in [10]). Reliable broadcast and consensus are building blocks for solving atomic broadcast in two of the atomic broadcast algorithms that we consider. In Sections 3.2 and 3.3, we shortly present the algorithms that are evaluated later.

3.1 Reliable broadcast, consensus and atomic broadcast

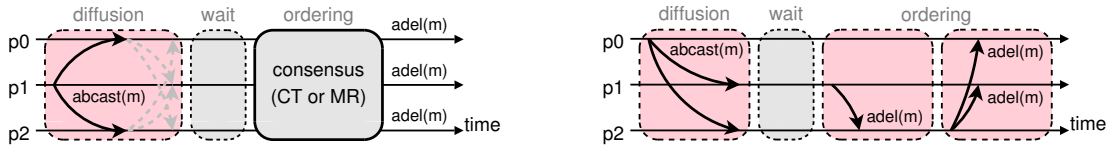
In the *reliable broadcast* problem, defined by the primitives *rbroadcast* and *rdeliver*, all processes need to agree on a common set of delivered messages. In this paper, we consider the reliable broadcast algorithm presented in [4], which requires $O(n^2)$ messages and a single communication step 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 [4] (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

Chandra-Toueg (CT) [4]: The first consensus algorithm, 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). In good runs, the CT algorithm behaves as follows: one of the processes, called the coordinator, sends its proposal to all processes (first communication step). The other processes reply by sending an acknowledgment to the coordinator (second step). The coordinator waits for $\lceil \frac{n+1}{2} \rceil$ acknowledgments (including its own) and reliably broadcasts the decided value to all processes (third step). Whenever a process reliably delivers a decision message for the first time, it decides.



(a) Chandra-Toueg (b) TokenFD

Figure 1: Communication pattern of the Chandra-Toueg and TokenFD atomic broadcast algorithms in runs without failures or wrong suspicions.

Mostéfaoui-Raynal (MR) [13]: The second consensus algorithm, 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. In good runs (and in a system with $n = 3$ processes), the MR consensus algorithm behaves as follows: as in CT, one of the processes, called the coordinator, sends its proposal to all processes (first communication step). The other processes reply by sending an acknowledgment to *all* processes (second step). All processes wait for $\lceil \frac{n+1}{2} \rceil$ acknowledgments (including their own). To guarantee the *Termination* property of consensus, a process that decides sends its decided value to all processes. In the case of a system with $n = 3$ processes, the non-coordinator processes can already decide after the first communication step, since they have received the coordinator's proposal and their own acknowledgment at that point.

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 experimental evaluation 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 [4]: Figure 1(a)[2] shows the communication pattern of Chandra and Toueg's atomic broadcast algorithm. It requires at least one reliable broadcast and a consensus execution for all processes to *abroadcast* and *adeliver* messages.

Whenever a message m is *abroadcast*, it is reliably broadcast to all processes (first communication step). 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*. The waiting period that is shown in Figure 1(a) 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. In runs without failures and suspicions, the TokenFD algorithm behaves as in Figure 1(b).

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 (in Figure 1(b), the token circulates between p_2 and p_3 in communication step 2). After the second communication step, m is *adelivered* by the token-holder which sends an update to all other processes about this

[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.

[2]Most figures use colors and are easier to understand when the paper is printed on a color printer.

delivery (third communication step). Again, the waiting period that is shown in Figure 1(b) only happens if the token is already being sent on the network 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 evaluate the performance of the three atomic broadcast algorithms (reduction to CT consensus; reduction to MR consensus; TokenFD algorithm). The benchmarks in [16, 19, 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_0 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, with $i \in 1, \ldots, n$. The latency L is then defined as $L \stackrel{def}{=} (\frac{1}{n}\sum_{i=1}^{n} t_i) - t_0$. In our 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 latency L is measured for a certain workload, which specifies how the *abroadcast* events are generated. We chose a simple symmetric workload where all processes send atomic broadcast messages[3] at the same constant rate and the *abroadcast* events follow a Poisson distribution. The global rate of atomic broadcasts is called the *throughput* T. We then evaluate the dependency between the latency L and 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. This state can only be reached if the throughput is below some maximum threshold T_{max}. Beyond T_{max}, some processes are left behind. We ensure that the system stays in a stationary state by verifying that the latencies of all processes stabilize over time.

Finally, we only evaluate the performance of the algorithms in good runs, i.e., without any process failures or wrong suspicions. The latency of the algorithms is measured once the system has reached a stationary state (at a sufficiently long time after the start up). The parameters that influence the latency are n (the number of processes), the algorithm (TokenFD, Chandra-Toueg atomic broadcast with CT or MR consensus) and the throughput.

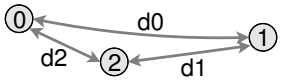
In the next section, we start by presenting an analytical model for evaluating the average latency of atomic broadcast algorithms in a wide area network. This model gives a first estimate of the relative performance of the processes. The evaluation of the algorithms in real wide area networks is then presented in Section 6. We specifically focus on the case of a system with three processes, supporting up to one failure.

5 Analytical performance evaluation

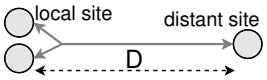
This section discusses an analytical performance evaluation of the three atomic broadcast (and consensus) algorithms in a wide area network. We start by describing the different phases that are common to all three algorithms and then present the two wide area network models that are considered. Due to lack of space, the derivation of the average latencies of the three algorithms is not presented here, but can be found in Appendices B and C.

The three phases of atomic broadcast. Both atomic broadcast algorithms work in three phases, as previously illustrated in Figure 1: upon *abroadcasting* a message m, (1) m is sent to all processes and (2) waits to be ordered. (3) Its order is then decided (using consensus or directly within the TokenFD atomic broadcast algorithm) and m is *adelivered*. The cost of the different

[3]The atomic broadcast messages do not contain any payload, in order to reach the maximum possible performance when comparing the three atomic broadcast combinations.



(a) Sending a message between location i and $i+1$ takes d_i time units.



(b) Sending a message between a local and a distant location takes D time units.

Figure 2: Theoretical model of a wide area network with three locations (2(a)) or two locations (2(b)). The processing times of messages are considered negligible.

phases (and thus the average latency) is of course directly related to the atomic broadcast and consensus algorithms that are used. These costs are presented in detail in Appendix A. Below, we present the two wide area network models that we consider.

Wide-area network with three locations. Figure 2(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. This assumption is reasonable if the latencies d_i between locations are much larger than the processing times of the messages on the critical path of the atomic broadcast algorithms (which is reasonable in a wide area network, but does not hold in a local-area network). Furthermore, the model does not take other factors into account, such as the bandwidth of the links or message loss. The average latency of the three atomic broadcast algorithms in this model can be found in Appendix B.

The performance of the algorithms in this model depends heavily on the relationship between the values d_0, d_1 and d_2. The analytical comparison between TokenFD, MR and CT with shifting coordinators when d_0, d_1 and d_2 can take any value is complex and omitted here. In the case of a fixed initial coordinator (presented in Table 1 in Appendix B), MR always performs better than CT, and for both algorithms, the best latency is achieved if the initial coordinator is on location 2 (and the worst latencies occur when coordinator on location 1).

Wide-area network with two locations. The algorithms that are evaluated require a system with at least three processes to tolerate one failure. These three processes can be distributed on up to three different locations. The situation where three locations are used is modeled above and the case where all three processes are on a single location is outside the scope of this paper, since a wide area network is no longer necessary. The second case where the processes are on *two* locations is however interesting: this setup limits the damage due to a catastrophic event on one of the locations and offers the possibility of serving clients from two separate locations (thus reducing the response latency in some circumstances). The model of the two-location system is presented in the following paragraphs.

Figure 2(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 relative performance of the three algorithms in the analytical model with two locations is the following (see Appendix C for the details). The MR algorithm always achieves latencies that are lower than (or equal to) the CT algorithm. If the initial coordinator of these algorithms is on the distant location, then the TokenFD algorithm has a better performance than both. If the initial coordinator shifts between locations at each new consensus execution, the TokenFD and MR algorithms achieve similar performance results, whereas CT has a higher latency. Finally, when the coordinator is on a local location, both CT and MR achieve more than two times lower latencies than the TokenFD algorithm.

6 Experimental performance evaluation

In the following section, the experimental performance of the atomic broadcast algorithms presented in Section 3 are compared. The next paragraph briefly discusses the framework in which the algorithms were implemented. The evaluation environments that were used are then presented and finally, 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 [17]. The various algorithms are implemented as micro-protocols and composed to form the final protocol stack. Every process in the system runs one of these Neko protocol stacks. Furthermore, all processes are connected pair-wise through TCP channels.

6.1 Evaluation environments

Four wide area network environments were used to evaluate the performance of the three atomic broadcast and consensus algorithms. Figure 3 shows a schematic representation of these four environments. 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. The round-trip times between locations are given by ping (in milliseconds), whereas the bandwidth of the links connecting the locations are given by the iperf [14] utility (in Megabits per second).

Three-location wide area network: The first evaluation environment (noted WAN Three Locations) is a system with three locations (Figure 3(a)) on Grid'5000 [3], 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.

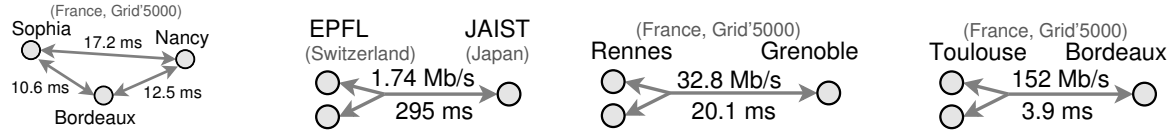
Two-location wide area networks: Three environments were used to evaluate the performance of atomic broadcast on wide area networks with two different locations:

– WAN 295: The first two-location environment consists of one location in Switzerland and one in Japan (Figure 3(b)). 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 3(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 3(d)) features a round-trip time between locations of $2D = 3.9$ ms and a link bandwidth of 152 Mb/s.

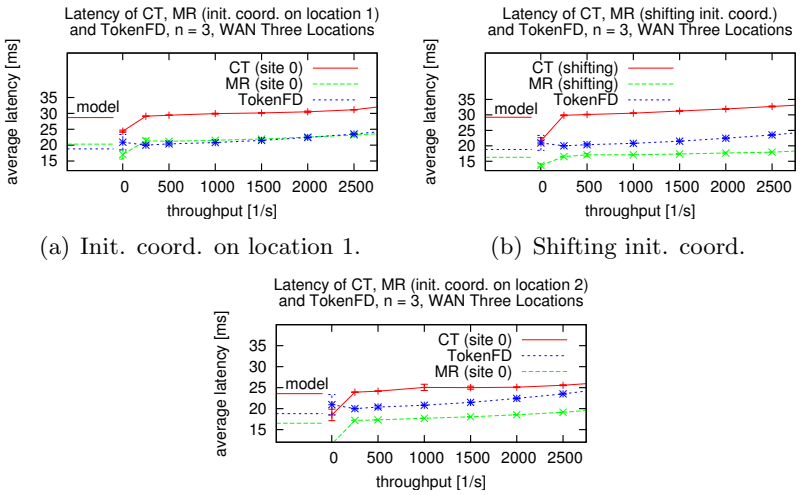
6.2 Comparing the experimental results with the model

The following paragraphs present 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



(a) WAN Three Locations (b) WAN 295 (c) WAN 20.1 (d) WAN 3.9

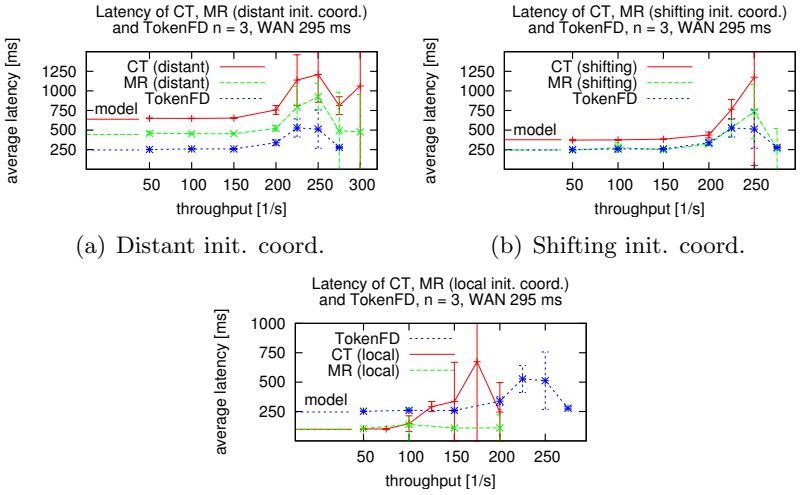
Figure 3: Wide area network evaluation environments in decreasing order of round trip times. The bandwidth and the round-trip time of the links between locations is shown for each environment.



(a) Init. coord. on location 1. (b) Shifting init. coord.

(c) Init. coord. on location 2

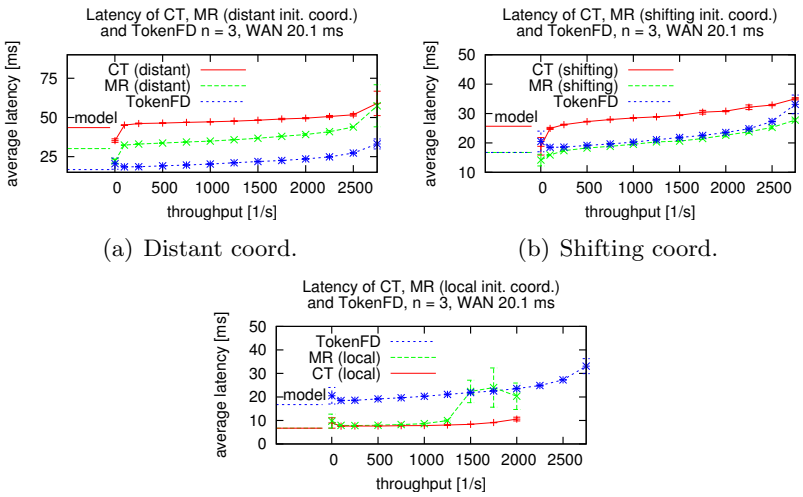
Figure 4: Average latency of the three atomic broadcast and consensus algorithms as a function of the throughput in the WAN Three Locations setting.



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

(c) Local init. coord.

Figure 5: Average latency of the three atomic broadcast and consensus algorithms as a function of the throughput in the WAN 295 setting.



(a) Distant coord.

(b) Shifting coord.

(c) Local coord.

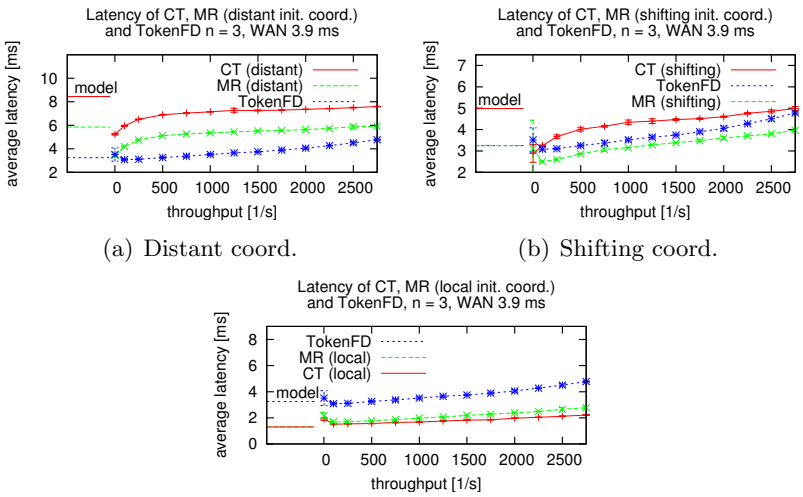
Figure 6: Average latency of the three atomic broadcast and consensus algorithms as a function of the throughput in the WAN 20.1 setting.

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 analytical performance of the algorithms in the model presented in Section 5 is shown on the far-left of each graph (noted "model").

In all experimental setups, the measurements confirm the estimations of the model (Figures 4 to 7), 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 Figures 5 and 6(c)), which increases the gap between the model's estimation and the actual measurements.

Furthermore, when the throughput is very low, as well as in the WAN 3.9 setting, 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. Similarly, in the WAN 3.9 setting, the consensus executions terminate fast enough that the waiting phase for many *abroadcast* messages only becomes a factor at higher throughputs, where the model is more accurate (Figures 7(a) and 7(b)).

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 Figures 5(c) and 6(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 1.74 Mbits/s in WAN 295 and 32.8 Mbits/s in WAN 20.1). The process on the distant location thus takes decisions at a slower rate than the two local processes and thus 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 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.



(a) Distant coord. (b) Shifting coord.

(c) Local coord.

Figure 7: Average latency of the three atomic broadcast and consensus algorithms as a function of the throughput in the WAN 3.9 setting.

6.3 Results of the performance evaluation

The following section discusses the performance ranking of the three atomic broadcast algorithms.

WAN Three Locations**:** The average latency of the three algorithms in the WAN Three Locations environment is presented in Figure 4. 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 lower 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 3.2.

WAN 295**,** WAN 20.1 **and** WAN 3.9**:** The average latency of the three atomic broadcast and consensus algorithms in the WAN 295, WAN 20.1 and WAN 3.9 environments are presented in Figures 5 to 7. TokenFD has lower latencies than CT and MR when they use a distant initial coordinator (Figures 5(a), 6(a) and 7(a)), whereas the situation is reversed when the coordinator is initially on a local location (Figures 5(c), 6(c) and 7(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 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 (as shown in Appendix D).* 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.

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

The performance of atomic broadcast and consensus algorithms based on failure detectors has been extensively studied in the context of local area networks.

In this paper, we presented an experimental evaluation in several wide area networks of three atomic broadcast and consensus algorithms using failure detectors and with different communication patterns. We also presented a simple analytical model of the performance of the three algorithms and validated it with the experimental measurements. The evaluation was performed in wide area networks with round-trip times ranging from about 4 to 300 milliseconds to examine the impact on the trade-off between the number of sent messages and the number of communication steps that the algorithms need to *adeliver* a message.

This study confirms that the relative performance between the algorithms is fundamentally different between a local area network and a wide area network: in the former case, the number of sent messages 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 from site to site at each new consensus) are effective in reaching higher throughputs in wide area networks.

References

[1] 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.

[2] O. Bakr and I. Keidar. Evaluating the running time of a communication round over the internet. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 243–252, New York, NY, USA, 2002. ACM Press.

[3] 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.

[4] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of ACM*, 43(2):225–267, 1996.

[5] A. Coccoli, P. Urbán, A. Bondavalli, and A. Schiper. Performance analysis of a consensus algorithm combining Stochastic Activity Networks and measurements. In *Proc. Int'l Performance and Dependability Symp.*, pages 551–560, Washington, DC, USA, June 2002.

[6] R. Ekwall and A. Schiper. Solving Atomic Broadcast with Indirect Consensus. In *IEEE International Conference on Dependable Systems and Networks (DSN 2006)*, June 2006.

[7] R. Ekwall, A. Schiper, and P. Urbán. Token-based atomic broadcast using unreliable failure detectors. In *Proceedings 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] V. Hadzilacos and S. Toueg. A modular approach to fault-tolerant broadcasts and related problems. TR 94-1425, Dept. of Computer Science, Cornell University, Ithaca, NY, USA, May 1994.
[11] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, 1998.
[12] 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.
[13] A. Mostéfaoui and M. Raynal. Solving consensus using Chandra-Toueg's unreliable failure detectors: A general quorum-based approach. In *Proceedings of the 13th International Symposium on Distributed Computing (DISC)*, number 1693 in Lecture Notes in Computer Science, pages 49–63, Bratislava, Slovak Republic, Sept. 1999. Springer-Verlag.
[14] National Laboratory for Applied Network Research. *Iperf 2.0.2*, 2006. http://dast.nlanr.net/Projects/Iperf/.
[15] 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.
[16] P. Urbán. *Evaluating the Performance of Distributed Agreement Algorithms: Tools, Methodology and Case Studies*. PhD thesis, École Polytechnique Fédérale de Lausanne, Switzerland, Aug. 2003. Number 2824.
[17] 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.
[18] P. Urbán, N. Hayashibara, A. Schiper, and T. Katayama. Performance comparison of a rotating coordinator and a leader based consensus algorithm. In *Proc. 23nd IEEE Int'l Symp. on Reliable Distributed Systems (SRDS)*, pages 4–17, Florianópolis, Brazil, October 2004.
[19] 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.
[20] 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.

Appendix

A The three phases of atomic broadcast

The three algorithms that are considered all work in three phases: upon *abroadcasting* a message m, (1) m is sent to all processes and (2) waits to be ordered. (3) Its order is then decided upon (using consensus or directly within the atomic broadcast algorithm) and m is *adelivered*. The cost of the different phases is of course directly related to the atomic broadcast and consensus algorithms that are used. However, the similarities between the algorithms lead to the following analysis.

We consider a message m *abroadcast* by a process p_i and analyze the costs of each one of the three phases. The cost of phases (1) and (2) depend on the cost of the third phase. Indeed, in phase (3), the atomic broadcast algorithm decides on the order of m and in all three algorithms, a "tutor" process p_j decides which messages should be ordered during this phase. In the CT and MR consensus algorithms, the tutor process is the coordinator whereas in the token-based atomic broadcast algorithm, it is the token-holder. The cost of phase (1), which consists in sending message m to all processes (and in particular to the tutor) depends on the choice of the tutor. Furthermore, the cost of phase (2), which consists in waiting for a new phase (3) to start, depends on the cost of the currently running phase (3).

For the phase (1), the cost of transmitting the message from p_i to the tutor p_j is noted $CostSend_{i,j}$. Similarly, the cost of phase (2) is noted $CostWait_{i,j}$ and the cost of ordering message m in phase (3) is noted $CostOrder_j$ (and only depends on the tutor process p_j). Thus, the cost of p_i *abroadcasting* a message m with p_j as tutor process is

$$CostSend_{i,j} + CostWait_{i,j} + CostOrder_j$$

Moreover, not all messages sent by a process p_i are necessarily ordered by the same tutor p_j. We define $Tutored_{i,j}$ to represent the fraction of all messages in the system that are *abroadcast* by p_i and ordered by tutor p_j (and we have $\sum_i \sum_j Tutored_{i,j} = 1$). Thus, by taking into account all sending processes p_i in the system, we have the average cost of *abroadcasting* a message m :

$$\sum_i \sum_j (CostSend_{i,j} + CostWait_{i,j} + CostOrder_j)\ Tutored_{i,j}$$

Deriving $CostSend_{i,j}$, $CostWait_{i,j}$, $CostOrder_j$ and $Tutored_{i,j}$ is done in Appendix B and C.

B Analytical performance in a wide area network with three locations

The following section presents the performance analysis of CT, MR and TokenFD in the wide area network model with three locations. The first part presents the derivation of the results of CT and MR using a fixed initial coordinator. The slightly more complex case of TokenFD or CT and MR using a shifting initial coordinator is then discussed.

B.1 Chandra-Toueg atomic broadcast

B.1.1 Fixed coordinator

Table 1 presents the average latency of Chandra-Toueg's atomic broadcast algorithm using the CT and MR consensus algorithms in a wide area network with three locations and a fixed initial coordinator. In the following paragraphs, we show how to derive the results for an initial coordinator that is always located on location p_0. The results for an initial coordinator on location p_1 or p_2 are derived similarly.

Phase (1): Message diffusion Let p_0 be the initial coordinator of the CT or MR consensus algorithm. Any process that *abroadcasts* a message, needs to send it to p_0 in phase (1). The cost of this diffusion is negligible if p_0 *abroadcasts* a message and equal to d_0 and d_2, if p_1, respectively p_2 are the *abroadcasters* of the message. On average (and since all three processes *abroadcast* at the same rate), we have a diffusion cost of $\frac{d_0+d_2}{3}$.

Phase (2): Waiting phase In phase (2), a message sent to the coordinator has to wait until a new consensus execution can start (assuming that a consensus execution is already underway). The coordinator p_0 can decide after two communication steps (with any of the two other processes) and thus starts a new consensus execution after that. The duration of these two communication steps is either $2d_0$ or $2d_2$ and since $d_0 \geq d_2$, the coordinator can decide after $2d_2$. Thus, a new consensus execution is started on average each $2d_2$ time units. Consequently, the average waiting time for an *abroadcast* message is d_2 time units.

Phase (3): Ordering (Consensus) Finally, the cost of phase (3) is the following. In the CT consensus algorithm, as seen previously, the coordinator decides after $2d_2$ time units and sends its decision to the two other processes. The process on location 1 thus decides d_0 time units after the coordinator whereas the process on location 2 decides d_2 time units after the coordinator. On average, this gives us a cost of phase (3) equal to $\frac{(2d_2)+(2d_2+d_0)+(2d_2+d_2)}{3} = \frac{7d_2+d_0}{3}$ for the CT consensus algorithm.

In the MR consensus algorithm, the coordinator also decides after $2d_2$ time units. However, if $n = 3$, the two other processes p_1 and p_2 can decide as soon as they get the coordinator's initial proposal, after d_0 and d_2 time units respectively. On average, this gives us a cost of phase (3) equal to $\frac{(2d_2)+(d_0)+(d_2)}{3} = \frac{3d_2+d_0}{3}$ for the MR consensus algorithm.

The sum of the three phases, which is the average latency for *abroadcasting* a message using Chandra-Toueg's atomic broadcast algorithm is thus equal to $\frac{11d_2+2d_0}{3}$ time units if CT's consensus algorithm is used and $\frac{7d_2+2d_0}{3}$ if MR's consensus algorithm is used.

B.1.2 Execution-based coordinator

The case of an initial coordinator that changes between consensus execution k and $k+1$ is slightly more difficult to analyze. Tables 2 and 3 present the $CostSend_{i,j}$, $CostWait_{i,j}$, $CostOrder_j$ and $Tutored_{i,j}$ matrices in a system where the CT, respectively the MR consensus algorithms are used and the initial coordinator changes at each consensus execution. The average latency of atomic broadcast is then easily derived following the results presented in Section A.

coord. location:	0	1	2
(1) diffusion:	$\frac{d_0+d_2}{3}$	$\frac{d_0+d_1}{3}$	$\frac{d_2+d_1}{3}$
(2) waiting:	d_2	d_1	d_2
(3) ordering:	$\frac{7d_2+d_0}{3}$	$\frac{7d_1+d_0}{3}$	$\frac{7d_2+d_1}{3}$
Latency:	$\mathbf{\frac{2d_0+11d_2}{3}}$	$\mathbf{\frac{2d_0+11d_1}{3}}$	$\mathbf{\frac{2d_1+11d_2}{3}}$

(a) CT consensus

0	1	2
$\frac{d_0+d_2}{3}$	$\frac{d_0+d_1}{3}$	$\frac{d_2+d_1}{3}$
d_2	d_1	d_2
$\frac{3d_2+d_0}{3}$	$\frac{3d_1+d_0}{3}$	$\frac{3d_2+d_1}{3}$
$\mathbf{\frac{2d_0+7d_2}{3}}$	$\mathbf{\frac{2d_0+7d_1}{3}}$	$\mathbf{\frac{2d_1+7d_2}{3}}$

(b) MR consensus

Table 1: Average latency to *adeliver* a message in the three-location wide area network model, using Chandra-Toueg's algorithm (with CT and MR's consensus algorithm and a fixed initial coordinator). Results are given for each phase of the algorithm and for a coordinator on each location.

B.2 TokenFD atomic broadcast

Table 4 present the $CostSend_{i,j}$, $CostWait_{i,j}$, $CostOrder_j$ and $Tutored_{i,j}$ matrices in a system with the TokenFD atomic broadcast algorithm. The average latency of atomic broadcast is then easily derived following the results presented in Section A.

$$\begin{pmatrix} 0 & d_0 & d_2 \\ d_0 & 0 & d_1 \\ d_2 & d_1 & 0 \end{pmatrix}$$

(a) $CostSend_{i,j}$

$$\begin{pmatrix} 2d_2 & \max(\frac{3d_2+d_1-d_0}{2}, 0) & d_2 \\ d_2 & 2d_2 + \frac{d_0}{2} + \min(\frac{d_0}{2}, \frac{3d_2+d_1}{2}) & \frac{3d_2+d_0-d_1}{2} \\ \frac{d_0+3d_1-d_2}{2} & d_1 & 2d_1 \end{pmatrix}$$

(b) $CostWait_{i,j}$

$$\begin{pmatrix} \frac{d_0+7d_2}{3} \\ \frac{d_0+7d_1}{3} \\ \frac{d_1+7d_2}{3} \end{pmatrix}$$

(c) $CostOrder_j$

$$\frac{1}{3\cdot(d_0+3d_1+5d_2)} \cdot \begin{pmatrix} 4d_2 & \max(3d_2+d_1-d_0, 0) & 2d_2 \\ 2d_2 & 2d_2+d_0+\min(d_0, 3d_2+d_1) & 3d_2+d_0-d_1 \\ d_0+3d_1-d_2 & 2d_1 & 4d_1 \end{pmatrix}$$

(d) $Tutored_{i,j}$

Table 2: $CostSend_{i,j}$, $CostWait_{i,j}$, $CostOrder_j$ and $Tutored_{i,j}$ in the three-location wide area network model, using Chandra-Toueg's algorithm (with CT's consensus algorithm and a shifting initial coordinator).

$$\begin{pmatrix} 0 & d_0 & d_2 \\ d_0 & 0 & d_1 \\ d_2 & d_1 & 0 \end{pmatrix}$$

(a) $CostSend_{i,j}$

$$\begin{pmatrix} d_2 & \max(\frac{d_1+d_2-d_0}{2}, 0) & 0 \\ 0 & \min(d_0, \frac{d_0+d_1+d_2}{2}) & \frac{d_0+d_2-d_1}{2} \\ \frac{d_0+d_1-d_2}{2} & 0 & d_1 \end{pmatrix}$$

(b) $CostWait_{i,j}$

$$\begin{pmatrix} \frac{d_0+3d_2}{3} \\ \frac{d_0+3d_1}{3} \\ \frac{d_1+3d_2}{3} \end{pmatrix}$$

(c) $CostOrder_j$

$$\frac{1}{3\cdot(d_0+d_1+d_2)} \cdot \begin{pmatrix} 2d_2 & \max(d_2+d_1-d_0, 0) & 0 \\ 0 & d_0+\min(d_0, d_1+d_2) & d_0+d_2-d_1 \\ d_0+d_1-d_2 & 0 & 2d_1 \end{pmatrix}$$

(d) $Tutored_{i,j}$

Table 3: $CostSend_{i,j}$, $CostWait_{i,j}$, $CostOrder_j$ and $Tutored_{i,j}$ in the three-location wide area network model, using Chandra-Toueg's algorithm (with MR's consensus algorithm and a shifting initial coordinator).

$$\begin{pmatrix} 0 & d_0 & d_2 \\ d_0 & 0 & d_1 \\ d_2 & d_1 & 0 \end{pmatrix}$$

(a) $CostSend_{i,j}$

$$\begin{pmatrix} d_2 & \max(\frac{d_1+d_2-d_0}{2}, 0) & 0 \\ 0 & \min(d_0, \frac{d_0+d_1+d_2}{2}) & \frac{d_2+d_0-d_1}{2} \\ \frac{d_0+d_1-d_2}{2} & 0 & d_1 \end{pmatrix}$$

(b) $CostWait_{i,j}$

$$\begin{pmatrix} \frac{4d_0+d_1}{3} \\ \frac{4d_1+d_2}{3} \\ \frac{4d_2+d_0}{3} \end{pmatrix}$$

(c) $CostOrder_j$

$$\frac{1}{3\cdot(d_0+d_1+d_2)} \cdot \begin{pmatrix} 2d_2 & \max(d_1+d_2-d_0, 0) & 0 \\ 0 & \min(2d_0, d_0+d_1+d_2) & d_0+d_2-d_1 \\ d_0+d_1-d_2 & 0 & 2d_1 \end{pmatrix}$$

(d) $Tutored_{i,j}$

Table 4: $CostSend_{i,j}$, $CostWait_{i,j}$, $CostOrder_j$ and $Tutored_{i,j}$ in the three-location wide area network model, using the TokenFD atomic broadcast algorithm.

C Analytical performance in a wide area network with two locations

The following paragraphs present the analytical performance of the TokenFD atomic broadcast algorithm and the Chandra-Toueg atomic broadcast algorithm using either the CT or the MR consensus algorithm. The results presented here can either be derived directly, or from the results in Appendix B by replacing d_0 and d_1 by D and d_2 by 0 (p_0 and p_2 are in the local location and p_1 is in the distant location). In the following paragraphs, we show how to derive these results directly.

C.1 Chandra-Toueg atomic broadcast

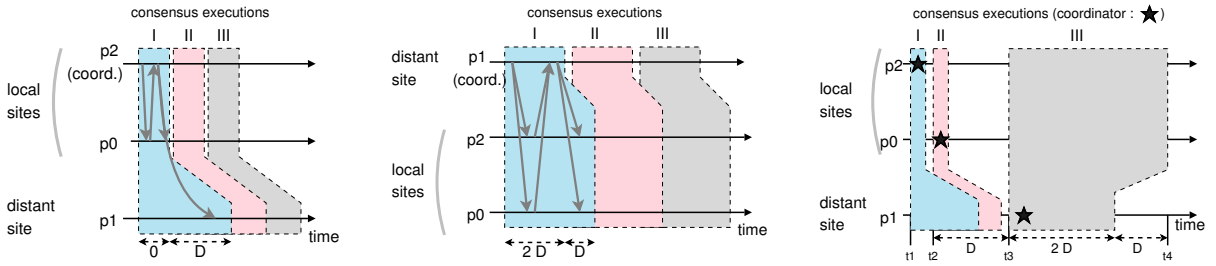
As previously discussed in Section A, the cost of the three phases depends on the choice of the tutor process in phase (3), i.e. the coordinator process in the CT and MR consensus algorithms. Three cases are considered : the initial coordinator (1) is always on a local location, (2) is always on the distant location or (3) shifts from location to location at each new consensus execution. Table 5 summarizes the latency of *abroadcast* in all three cases, when the CT (Table 5(a)) and MR (Table 5(b)) consensus algorithms are used. The analytical latencies of the cases where the initial coordinator is on a fixed location or shifts at each consensus execution are derived in the following paragraphs.

C.1.1 Fixed coordinator

Phase (1): Message diffusion The cost of the message diffusion phase is the following: (1) if the initial coordinator is on a local location, it receives *abroadcast* messages from itself and its local peer (with a negligible cost) and from the distant location (with a cost of D). The average cost for diffusing the message to the local coordinator is therefore $\frac{D}{3}$. (2) If the coordinator is on the distant location, it receives messages from the local locations (both with a cost of D) and from itself (with a negligible cost). The average cost for diffusing the message to the distant initial coordinator is thus $\frac{2D}{3}$.

Phase (2): Waiting phase The duration of the waiting phase is the following. In the CT consensus algorithm, the coordinator decides after 2 communication steps and atomic broadcast immediately starts a new consensus (if unordered messages are waiting), as illustrated in Figures 8(a) and 8(b) (the consensus execution II starts before all processes have finished consensus execution I). The cost of these two communication steps is negligible if the coordinator is on the local location (thus, the waiting time for messages to be proposed in a consensus is negligible). If the coordinator is distant, the two communication steps take $2D$ time units and the messages wait on average D time units to be proposed in a consensus (since messages are *abroadcast* following a Poisson process). The reasoning (and the average waiting times) for the MR consensus algorithm is similar, since the coordinator again needs two communication steps to reach a decision.

Phase (3): Ordering (Consensus) Finally, the cost of the CT consensus phase is illustrated in Figures 8(a) and 8(b) and is the following. The coordinator can decide after two communication steps and all other processes after three steps. (1) If the coordinator is on the local location, a decision is taken after two local communication steps (with a negligible cost) and is received by the other local location one local communication step later (again, with a negligible cost). The distant location receives the coordinator's decision after one distant communication step (with a cost of D). The average latency is thus $\frac{D}{3}$. (2) If the coordinator is on the distant



(a) Local coordinator (b) Distant coordinator (c) Shifting coordinator

Figure 8: Execution pattern of the Chandra-Toueg consensus algorithm in the two-location wide area network model and in the case of a coordinator on a local location, a distant location or shifting between locations at each execution.

location, it needs 2 distant communication steps to decide (with a cost of $2D$), whereas both local locations decide one distant communication step later (with a total cost of $3D$ for both local locations). The average decision latency with a distant coordinator is therefore $\frac{8D}{3}$.

The cost of the consensus phase using the MR algorithm is similar. The case of a local coordinator gives the same result as CT with an average latency of $\frac{D}{3}$. In the case of a distant coordinator, both local locations decide as soon as they receive the coordinator's proposal and their own acknowledgment (resulting in a latency of D). The coordinator decides as soon as it gets an acknowledgment from a local location, after a total time of $2D$. The average latency over all processes is therefore $\frac{4D}{3}$.

C.1.2 Execution-based coordinator:

When the initial coordinator changes at each new consensus execution, the analysis is slightly more complex than in the case of an initial coordinator that remains on a single location (presented in Appendix C.1.1). Indeed, the consensus executions no longer have the same duration, which in turn means that the messages that are *abroadcast* aren't uniformly distributed among the different consensus executions. For example, if short and long consensus executions alternate, then more messages are ordered in the short consensus (since more unordered messages are accumulated during the execution of the long consensus). A more precise end-to-end analysis of the latencies of the *abroadcast* messages is necessary and is presented in the following paragraphs for the case of the CT consensus algorithm. The results with the MR consensus algorithm have been derived similarly.

Phase (1): Message diffusion Figure 8(c) presents the sequence of CT consensus executions when the initial coordinator changes each time. Two consensus executions with an initial coordinator on the local location closely follow each other, followed by an execution with a coordinator on the distant location. We start by analyzing which messages are ordered in which consensus executions.

A message m *abroadcast* by p_2 or p_0 (the two processes on the local location) is (almost) always proposed in consensus executions where p_2 is the coordinator. Indeed, when m reaches p_1, either p_1 has already started the consensus in which it is coordinator (and cannot add m to that consensus) or m is being decided upon in a consensus where p_2 is the coordinator. A message m *abroadcast* by p_1 between t_1 and t_3 (see Figure 8(c)) is ordered in a consensus execution with p_1 as a coordinator. Messages *abroadcast* by p_1 between t_3 and $t_3 + 2D$ are received by p_2 and p_0 before t_4 and are thus ordered in a consensus execution with p_2 as coordinator. Finally, messages *abroadcast* between $t_3 + 2D$ and t_4 do not reach p_2 before t_4 and are later ordered in

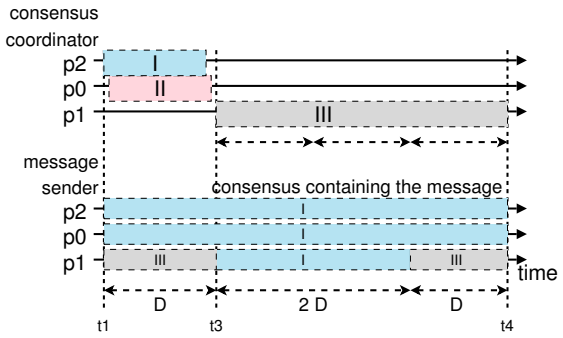


Figure 9: Messages *abroadcast* by p_0 or p_2 are all ordered in consensus executions with p_2 as coordinator (noted I). Messages *abroadcast* by p_1 are ordered in executions with p_1 or p_2 as coordinator (noted III and I respectively).

a consensus execution with p_1 as coordinator.

In total, $\frac{5}{6}$ of all messages are ordered in consensus executions with p_2 as coordinator and the remaining $\frac{1}{6}$ when p_1 is coordinator (consensus executions with p_0 as coordinator order only a negligible amount of messages), as summarized in Figure 9. The sending time of $\frac{5}{6}$ of the messages is thus 0, whereas it is equal to D in $\frac{1}{6}$ of the cases. The average sending time over all messages is thus $\frac{D}{6}$.

Phase (2): Waiting phase On average, the messages *abroadcast* by p_0 and p_2 ($\frac{2}{3}$ of the messages) wait $2D$ time units before being proposed in a consensus (with p_2 as a leader). Among the messages *abroadcast* by p_1 ($\frac{1}{3}$ of all messages), the waiting time is on average D. The average waiting time over all messages is thus $\frac{5}{3}D$.

Phase (3): Ordering (Consensus) As previously presented in Section C.1.1, the consensus executions where p_2 (on the local location) is the coordinator, the average latency is $\frac{D}{3}$. These executions order $\frac{5}{6}$ of all messages. The remaining $\frac{1}{6}$ of all messages are ordered in executions with p_1 (on the distant location) as coordinator, and thus with an average latency of $\frac{8D}{3}$. The average consensus execution time over all messages is thus $\frac{13D}{18}$.

Summary Globally, this gives an average latency of $\frac{23D}{9}$ for *abroadcasting* a message using the Chandra-Toueg atomic broadcast and consensus algorithms if the coordinator changes at each consensus execution.

A similar analysis gives an average latency of $\frac{5D}{3}$ for *abroadcasting* a message using the Chandra-Toueg atomic broadcast and Mostéfaoui-Raynal consensus algorithms if the coordinator changes at each consensus execution.

C.2 TokenFD atomic broadcast

The following paragraphs present the analytical latency of the TokenFD atomic broadcast algorithm. As previously shown in Figure 1(b), a token circulates among the three processes and the set of messages in the token proposal is *adelivered* by each token holder (which then proposes a new set of undelivered messages). The time needed to pass the token between processes is not uniform : indeed, it is negligible when the token is passed between two processes on the local location whereas it is equal to D when passed between a local and a distant location or vice-versa. Due to this non-uniformity, the set of messages transported by the token is not the same for all tokens, which in turn influences in the latency of atomic broadcast.

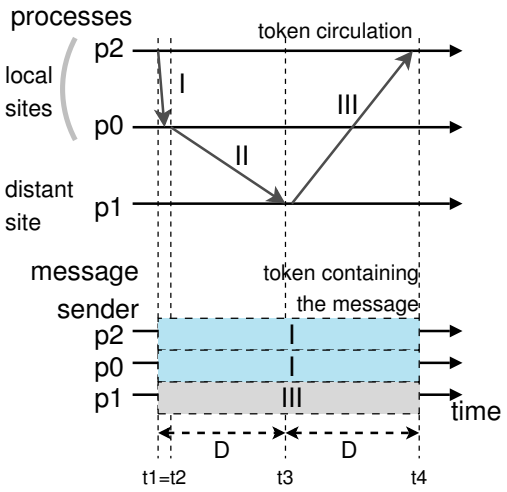


Figure 10: Token circulation (top) in the two-location wide area network model and a presentation of which token contains the messages *abroadcast* by the three processes (bottom).

Figure 10 shows the token circulation (top part) and an analysis of which token contains the messages *abroadcast* by the three processes (bottom part). (Almost) all messages sent by p_2 and p_0 are later contained in token I sent by p_2 : indeed, if a message is *abroadcast* between t_2 and t_4, it does not reach p_1 before t_3 and is therefore not added to the token III sent by p_1 (but later added to the token I sent by p_2). Similarly, all messages sent by p_1 are later ordered in the token III sent by p_1. Finally, token II, sent by p_0 contains only the messages received by the local hosts in the interval between t_1 and t_2 which is negligible in our model. To summarize, two-thirds of the messages are later ordered in token I sent by p_2, whereas the remaining third is ordered in token III sent by p_1.

Phase (1): Messages diffusion The latency of the message diffusion in the TokenFD atomic broadcast algorithm is negligible. Indeed, messages sent by p_2 and p_1 are later contained in tokens sent by p_2 and p_1 respectively, and therefore have no diffusion cost. Messages sent by p_0 are ordered in tokens sent by p_2, resulting in a negligible diffusion cost, since p_0 and p_2 are both on the local location.

Phase (2): Waiting phase The cost of the waiting phase in the TokenFD atomic broadcast algorithm is the following. Messages sent by p_0 and p_2 need to wait until the token is held by p_2 to be ordered. On average, this translates into a waiting time of D. The cost of the waiting phase for the messages sent by p_1 is derived in the same way and also yields an average waiting time of D.

Phase (3): Ordering phase There are only two tokens that contain (almost) all unordered messages. Token I, sent by p_2, reaches p_0 after a negligible amount of time. Process p_0 then sends updates to p_2 (negligible latency) and p_1 (latency of D) about the ordered messages. The average latency of the ordering phase over all processes of token I is thus $\frac{D}{3}$.

Token III, sent by p_1 reaches p_2 after D time units. Process p_0 receives an update from p_2 shortly after, whereas p_1 receives the update after D additional time units. The average latency of the ordering phase over all processes of token III is thus $\frac{4D}{3}$.

Since $\frac{2}{3}$ of the messages are ordered in token I and $\frac{1}{3}$ in token III, the average ordering latency over all messages is $\frac{2D}{3}$.

Summary By summing the latencies of the three phases, this gives an average latency of $\frac{5D}{3}$ for *abroadcasting* a message using the TokenFD atomic broadcast algorithm. These results are summarized in Table 7.

coord. loc.:	local	distant	shifting
(1) diffusion:	$\frac{D}{3}$	$\frac{2D}{3}$	$\frac{D}{6}$
(2) waiting:	0	D	$\frac{5D}{3}$
(3) ordering:	$\frac{D}{3}$	$\frac{8D}{3}$	$\frac{13D}{18}$
Latency:	$\mathbf{\frac{2D}{3}}$	$\mathbf{\frac{13D}{3}}$	$\mathbf{\frac{23D}{9}}$

(a) CT consensus

local	distant	shifting
$\frac{D}{3}$	$\frac{2D}{3}$	0
0	D	D
$\frac{D}{3}$	$\frac{4D}{3}$	$\frac{2D}{3}$
$\mathbf{\frac{2D}{3}}$	**3D**	$\mathbf{\frac{5D}{3}}$

(b) MR consensus

N/A
0
D
$\frac{2D}{3}$
$\mathbf{\frac{5D}{3}}$

(c) TokenFD alg.

Table 5: Average latency to *adeliver* a message in the two-location wide area network model, using Chandra-Toueg's algorithm (with CT or MR's consensus algorithm) or the TokenFD algorithm. Results are given for each phase of the algorithm and, for the two consensus algorithms, for an initial coordinator on a local location, on the distant location or that shifts at each new consensus execution.

consensus algorithm :	CT	MR
diffusion latency:	$\frac{D}{6}$	0
consensus latency:	$\frac{13D}{8}$	$\frac{2D}{3}$
waiting period:	$\frac{5D}{3}$	D
Total latency:	$\frac{23D}{9}$	$\frac{5D}{3}$

Table 6: Average latency to *adeliver* a message in the two-location wide area network model using Chandra-Toueg's atomic broadcast algorithm coupled with Chandra-Toueg's or Mostéfaoui-Raynal's consensus algorithm. The initial coordinator is on a different location for each new consensus execution.

diffusion latency:	0
ordering latency:	$\frac{2D}{3}$
waiting period:	D
Total latency:	$\frac{5D}{3}$

Table 7: Average latency to *adeliver* a message using the TokenFD atomic broadcast algorithm in the two-location wide area network model.

D Performance of the three algorithms in a local area network

We now briefly present the evaluation of the CT, MR and TokenFD algorithms in a local area network. Each local area node has a Pentium 4 processor at 3 GHz with 1GB of RAM. The nodes are interconnected by Gigabit Ethernet and have a round-trip time of about 0.1 ms (given by ping). The bandwidth of the link is about 942 Mbits/s (given by iperf).

Figure 11 shows the average latencies of the three algorithms in a local area network. This time, the number of communication steps has seemingly no impact on the performance of the algorithms. The main factor is the total processing time of the messages (which is influenced by the number of messages and the complexity of the data handling in the algorithm itself). CT consensus needs less messages than MR to reach a decision and logically achieves lower latencies, especially as the throughput increases. TokenFD needs least messages of all to *adeliver* a message, but also requires more processing time per message (at each token reception, for example, the algorithm needs to perform sequence comparisons that neither CT, nor MR need to do). As a consequence, TokenFD's initial performance is between CT and MR, but later worsens as the load on the processors increases with the throughput in the system.

Compared to the wide area network case presented in Section 6, (where CT performed worse than MR in almost all scenarios and worse than Token in most scenarios), the performance ranking between the algorithms is reversed this time.

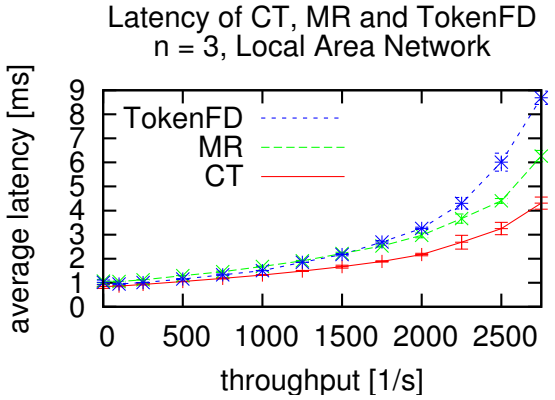


Figure 11: Average latency of the three atomic broadcast and consensus algorithms as a function of the throughput in the LAN setting.