The Road to a more Configurable and Adaptive Communication and Coordination Support

Luís Rodrigues

DI–FCUL TR–2003–2

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749-016 Lisboa
Portugal

Technical reports are available at `http://www.di.fc.ul.pt/biblioteca/tech-reports`. The files are stored in PDF, with the report number as filename. Alternatively, reports are available by post from the above address.

The Road to a more Configurable and Adaptive Communication and Coordination Support*

Luís Rodrigues†

Universidade de Lisboa
FCUL, Campo Grande,
1749-016 Lisboa, Portugal
ler@di.fc.ul.pt

Abstract

The implementation of distributed applications is an increasingly complex task. Not only the users require new and more complex functionalities but these have to be provided considering a large set of non-functional requirements such as high performance, fault-tolerance, timeliness, etc. Therefore, communication and coordination services, offered at the operating system or middleware level, assume a fundamental role in the development of efficient and robust software. This paper discusses the problem of designing and implementing the communication and coordination support for distributed applications. One way to implement these services is to rely on application-specific solutions, in an attempt to obtain the best performance possible. In this paper, we champion a different approach that consists in supporting adaptation, configuration and composition at all levels of the system development, namely: *i)* abstractions; *ii)* algorithms and; *iii)* implementations. Using different examples we show that this approach allows to obtain solutions that are generic, re-usable and efficient.

1 Introduction

The development of distributed applications is a complex task, mainly due to the large set of functional and non-functional requirements that need to be addressed by the programmer. Among others, performance, fault-tolerance, timeliness, and security, are some examples of relevant non-functional requirements. Additionally, due to commercial reasons, the development cycle of many applications is becoming shorter, emphasizing the need for an adequate communication

*Selected sections of this report will be published in the Proceedings of the 9th Workshop on Future Trends of Distributed Computing Systems. San Juan, Puerto Rico, May 2003.

†This work has been partially supported by the FCT project SHIFT (POSI/CHS/40088/2001).

and coordination support (this support can be provided at the operating system or middleware level).

Therefore, it is of utmost importance to identify abstractions that are able to encapsulate a significant amount of the complexity inherent to distributed computing. Abstractions such as consensus [16], atomic commitment [2], causal order [24], total order [18], and virtual synchrony [4], among others, capture recurring and fundamental problems in the design of distributed applications.

Unfortunately, when trying to apply in a naive way some of these abstractions to concrete applications, one may be faced with implementation difficulties that may limit the performance of the resulting system and raise doubts about the practicality of these abstractions [9, 38]. A superficial analysis of the causes for these limitations may lead to the conclusion that the only way to achieve good performance is to rely on tailor-made solutions. Typically, these solutions solve application-specific problems but cannot be applied outside their original design context.

In this paper we advocate an approach to the construction of the communication and coordination middleware that balances the advantages gained from using clean and elegant abstractions with the need to satisfy the performance requirements of distributed applications. The approach consists in supporting adaptation, configuration and composition at all levels of the system development, namely: *i)* abstractions; *ii)* algorithms and; *iii)* implementations. The conclusions drawn here are derived from the experience gained by the Distributed ALgorithms and Network Protocols (DIALNP) group at Faculty of Sciences, University of Lisboa in the attempt to answer the following questions:

- **Abstractions**: How to encapsulate the inherent difficulties of distributed computing without compromising the performance of the applications?
- **Algorithms**: Given an abstraction and a system model which is the best algorithm to implement the abstraction? Is it possible to develop adaptive algorithms that adjust themselves to different operational conditions?
- **Systems**: What is the most adequate software infra-structure to support the implementation of the previous algorithms? Can we satisfy the needs of different applications without being forced to implement a spectrum of specialized monolithic implementations?
- **Applications**: Are the abstractions, algorithms, and implementations proposed when addressing the previous questions, adequate to support real applications? In this paper we will use data replication as a case study to illustrate the advantages of offering adaptation, configuration and composition in the communication and coordination support.

The rest of the paper is structured as follows. The need for some basic abstractions to support data replication is motivated in Section 2. Section 3 shows how basic abstractions can be augmented to better match the application requirements. The design of algorithms that implement these abstractions is discussed in Section 4 and their implementation addressed in Section 5. Finally, the application of these results to real cases is described in Section 6. Section 7 concludes the paper.

2 Data Replication

To motivate our approach we use a concrete application area, namely, data replication. There are two well known main reasons to replicate data:

- *Improve performance.* A centralized datastore represents a bottleneck in the system. Additionally, clients geographically distant from the store suffer a significant delay in the data access. When data is replicated it can be placed near the clients. Also, read operations can be performed in parallel at different replicas.

- *Fault-tolerance.* A centralized datastore represents a single point of failure. A failure in the node that stores the data may cause the data to be lost or unavailable for a long period. Replication may ensure data availability in the presence of faults.

Unfortunately, there are many costs associated with data replication. In order to ensure consistency, data updates need to be coordinated. Coordination may require several rounds of message exchange and seriously limit the system performance. Due to these costs, typically data replication is more effective in systems where the number of read operations is significantly larger than the number of write operations. However, even in this case, it is important to ensure that the costs of coordination are acceptable.

Data replication is an interesting example because it illustrates how the choice of an inappropriate abstraction may invalidate a large set of applications. Early approaches to data replication were based in quorums [12, 11] and did not rely on other communication or distributed coordination primitives to simplify the task of serializing concurrent transactions [3]. The resulting high number of aborts due to deadlocks observed in some of these approaches raised the question of the feasibility of applying data replication in practice [14].

However, later work has shown that data replication is not only possible but can also outperform systems that do not use replication [22, 23, 1]. The new family of solutions is based on a set of abstractions which is substantially different from the abstraction used by early approaches. In particular, the new generation of systems relies on the following abstractions:

- *Uniform reliable multicast.* Informally, this primitive ensures that if a message is delivered to a given participant it is delivered to all correct participants.

- *Uniform total oder.* Informally, this primitive ensures that messages are delivered to all participants in the same order.

A primitive satisfying both reliability and order is usually called *atomic multicast*. Using these two abstractions, among others, it is possible to replicate data using one of the two following basic approaches [15]:

- *Passive replication.* This technique consists in electing a replica as a primary. All updates are performed at the primary which, in turn, propagates them to the backup replicas. This technique is more effective in systems where it is possible to implement an accurate failure detector to simplify the process of electing the primary.

- *Active replication.* This technique execute all updates in parallel at all replicas. To ensure that all updates are processed in the same order, an atomic multicast primitive must be used [18]. This technique does not require the explicit election of a primary but requires the execution of an atomic broadcast for each update/transaction.

The advantages of these abstractions have been illustrated by several projects on database replication over local-area networks [23, 19]. To achieve the same advantages in large-scale networks these basic abstractions may need to be augmented [1, 30, 39]. In the following section we will give two different examples of this approach.

3 New Abstractions

In the previous section we have identified some useful abstractions to support data replication. In this section we discuss the adaptation of two of these abstractions, more specifically, of reliable multicast and total order. We motivate these adaptations by discussing some of the limitations of the basic abstraction when applied to the development of concrete systems.

3.1 Semantic Reliability

By definition, a reliable multicast algorithm ensures that each message is delivered to all correct participants. When implementing these algorithms in practical systems one needs to resort to some form of acknowledgment (and retransmission) mechanism (to recover from omission faults). Usually, when a message has been acknowledged by all participants, it is said to be *stable*. Stable messages can be discarded from retransmission buffers. A large body of work has been performed on the optimization of stability tracking mechanisms [17].

A limitation of reliable multicast when applied to heterogeneous groups is that the complete system becomes dependent of the timely behavior of *all* members. It is enough that a single member is slow, and does not acknowledge the reception of messages in a timely manner, in order for the sender to accumulate messages in its buffer. If the slow member does not recover fast enough, the sender exhausts its buffer and is forced to stop until the slow member recovers. Unfortunately, this behavior can be observed even when the performance fault of the slow member is transitory [5]. This happens because different participants may be perturbed at different times, leading to a permanent degradation of performance in the system.

A possible solution to avoid the limitation described above consists in abandoning the provision of reliable multicast. However, when unreliable primitives are used one is usually forced to design application-specific recovery procedures. This approach is suggested, for instance, in [10] and [6]. However, in this paper we are interested in solutions that minimize the amount of specialized code that needs to be executed at the application level.

Using semantic information about the contents of the messages exchanged in the system, it is possible to propose a generalization of reliable multicast called *semantic reliability* [28]. This new abstraction exploits the fact that, in a substantial amount of applications, the transmission of a new message may

cause preceding messages to become obsolete. For instance, two consecutive transaction may update the same data items: in this case, the last transaction makes the updates from the previous transaction obsolete.

Let the fact that a message m is made obsolete by another message m' be represented by the relation $m \sqsubset m'$. Semantic reliability is characterized by the following agreement property[1]:

Agreement: If a correct process delivers a message m, and there is a time after which no process multicast a message m'' such that $m \sqsubset m''$, then all correct processes eventually deliver some message m' such that $m \sqsubseteq m'$.

It is interesting to note that the definition of semantic reliability is equivalent to uniform reliability when no message is made obsolete by any other message. The usage of semantic reliability assumes that the application is able to label the messages to express the obsolescence relation. A description of different techniques to capture and express obsolescence relations is given in [30].

3.2 Optimistic Total Order

We now give another example of an adaptation of a basic abstraction to create a more powerful, and efficient, new abstraction. This new abstraction, called *optimistic total order* [27] has been proposed to address the performance limitations of totally ordered multicast protocols.

It can be shown that the problem of ordering multicast messages in a total order is, in asynchronous systems, equivalent to the consensus problem [8]. On the other hand, there are also known lower bounds on the number of communication steps to solve consensus: more specifically, at least two communication steps are required [21] (but most total order protocols for the asynchronous system model require a larger number of steps [8, 33]). Therefore, total order introduces an overhead in terms of communication steps that cannot be avoided. However, in most algorithms, it is possible to have an estimate of the final total order before the algorithm is terminated.

The idea of optimistic total order is based on the fact that several applications, namely applications with transactional semantics, can make progress based on an early estimate of the final total order (and later confirm or abort this progress according to the final result of the algorithm). The approach assumes that it is possible to design total order algorithms where the estimate is accurate with high probability. In the following section we will briefly introduce a new algorithm with this characteristic.

As in the example of semantic reliability, we can also consider optimistic total order as a generalization of a more traditional total order abstraction (note that a protocol that delivers the estimate at the same time of the final order corresponds to the classical total order definition [18]).

3.3 Discussion

The two previous examples show that the satisfaction of performance requirements can be addressed at the abstraction level without relying on application-

[1] In fact, it is possible to define a complete protocol family based on the notion of semantic reliability [28, 29, 30]. A complete description of such family of protocols is outside the scope of the current paper.

specific solutions. The given examples are quite generic (actually, they can be seen as generalization of the traditional definitions) but include the necessary extensions to address performance aspects also at the algorithmic and implementation levels.

4 New Algorithms

Naturally, to define new abstractions is not enough to support the development of distributed application in an efficient manner. It is also necessary to design the algorithms able to implement these abstractions. In the following paragraphs we provide a brief summary of some new algorithms that implement the abstractions described in the previous section.

4.1 Semantic Reliable Multicast

As noted previously, any algorithm based on the notion of semantic reliability requires the user to label the message with information that captures the obsolescence relation. We will not address this problem here (the interested reader can refer to [30]). On the other hand, we will show how this information can be used by a concrete algorithm.

The basic principles of the algorithm are the following [29]: messages are sent to all recipients, as in a fully reliable algorithm. Also, the reception of messages is acknowledged, and these acknowledgments are disseminated to all participants (either using dedicated control messages or by piggybacking this information in data messages). When a message is acknowledged by a majority of group members, this message is said to be *safe*. When a message is acknowledged by all group members the message is said to be *stable* and can be eliminated from the retransmission buffers.

The motivation to the introduction of semantic reliability is the presence of slow group members, unable to acknowledge the reception of messages at an acceptable pace. To accommodate these nodes, the algorithm deletes from the buffers obsolete messages before they are stable without compromising the correctness of the applications. To achieve this goal, each node searches its local buffers for a message m, that has been made obsolete by some other message m' such that: *i)* a copy of m' is also stored in the local buffer *and*; *ii)* m' is safe. If such message m exists, m is deleted from the buffer, creating space for new messages. A process that deletes a massage in this way, replaces the message by a small marker (with the identity of the message). In all cases where the message should have been sent, the marker is sent as a replacement. This allows receivers to know that the message was made obsolete. The reception of markets is acknowledge in the same way as normal messages. Similarly, stable markers can also be discarded from buffers.

In a group where a majority of processes has no performance constraints, obsolete messages can be purged to create space for more up-to-date information. Depending on the buffer size and on the obsolescence pattern exhibited by the application, this strategy can successfully delay, or even avoid altogether, the blocking of the sender in the presence of transient performance faults in one or more group members.

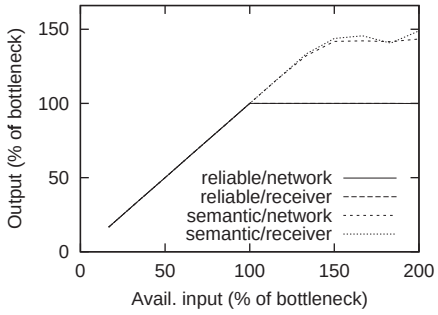


Figure 1: Advantages of a semantic reliable multicast algorithm.

Figure 1 illustrates the advantages of the algorithm by showing the throughput of a sender in the presence of a slow recipient or link. In this case, half of the messages never become obsolete and the other half obsoletes each other. The figure depicts four lines, showing the behavior of an algorithm without obsolescence and of the semantic reliable algorithm for the cases where the bottleneck is the sender and a network link. In a fully reliable protocol the throughput of the sender is limited by the bottleneck in the system. Using semantic reliability it is possible to sustain a throughput 50% higher (corresponding to the amount of messages that can be discarded from the buffers). Naturally, the traffic obsolescence pattern affects the shape of these lines. We will return to this issue later in the paper.

4.2 An Indulgent Optimistic Total Order Algorithm

The abstraction of optimistic total order assumes that it is possible to derive algorithms that are able to:

- Obtain an estimate of the final order of the messages before the termination of the algorithm;
- Ensure that in most runs the estimated order is accurate with regard to the final order.

The first proposed algorithms supporting optimistic delivery were based on characteristics specific to concrete classes of systems. For instance, the algorithm described in [27] assumes the availability of a local area network, as the estimated order is based on the network spontaneous order. On the other hand, the algorithm proposed in [36] can operate over large-scale networks but the estimate order is based on an evaluation of the network delay, and there is a non-negligible probability for the estimate to be inaccurate.

In the following paragraphs we provide a brief overview of an optimistic total order algorithm in which the estimate is always accurate as long as no processes are suspected. The algorithm, described in detail in [39] is to our knowledge the first algorithm to provide highly accurate optimistic delivery in a general setting.

The algorithm is based on the idea of combining a total order protocol that does not consider the occurrence of faults, that we simply designate by OPT,

with a consensus algorithm that only assumes the availability of an eventually strong failure detector ($\diamond\mathcal{S}$) [8]. Basically, the OPT algorithm is used to generate the estimated total order, since it exhibits a smaller latency than any algorithm designed for $\diamond\mathcal{S}$. The consensus algorithm is used to terminate on-going multicast when failures are suspected and to reconfigure OPT. The chosen OPT algorithm is the hybrid total order algorithm described in [32] due to its good performance in geographically large-scale systems. Specifically, the algorithm is based on the following strategy:

1. Based on an estimate of network delays and on the output of the $\diamond\mathcal{S}$ failure detector, a configuration for OPT is proposed. This step allows to adapt the configuration of OPT to the properties of the execution environment.

2. Algorithm OPT is used to order messages. The order proposed by OPT is delivered to the application as an estimate of the final order.

3. An additional communication step is executed to stabilize the estimated order and make it definitive [13]. If processes are not suspected to have failed, the estimate becomes stable without being required to execute explicitly a consensus algorithm.

4. If the failure of a process is suspected, the execution of OPT is suspended. A consensus protocol is executed to order on-going messages transmissions. The algorithm is re-started from step 1.

Table 1 illustrates the latency gains that can be achieved when this algorithm is used. The values have been measured using two local-area networks connected by a long-haul link exhibiting a latency in the order of $200ms$ (a detailed description of the experimental setting can be found in [39]). The table compares the latency of: *i)* a non-uniform total order protocol based on a perfect failure detector (TO); *ii)* the optimistic delivery of the protocol described above (UTO-opt) and; *iii)* the final total order for the same protocol (UTO). Results are shown for three different configuration of OPT, where the hybrid configuration is the one chosen by our adaptive mechanisms. It is possible to observe not only the smaller latency offered by the hybrid configuration but also a difference between the latency of the estimate and the final delivery in the order of one communication step.

	TO	UTO-opt	UTO
A single sequencer	475.0	479.0	652.7
All sequencers	328.0	328.0	484.3
Hybrid	303.0	303.0	496.7

Table 1: delivery latency (ms)

An interesting aspect of this algorithm is that it proposes a technique that allows to exploit the efficiency of algorithms designed for the perfect failure detector model ($\mathcal{P}$) in system where only an eventually strong failure detector can be assumed.

4.3 Discussion

It is possible to design new algorithms that, through adaptation, improve significantly the performance of non-adaptive versions. The semantic reliable multicast algorithm described above can be easily parameterized by the application and the optimistic total order algorithm has the ability to reconfigure in order to adapt to changes in the operational envelope.

5 An Implementation: using the *Appia* System

We have already addressed the issue of defining new abstractions and specifying new algorithms to support those abstractions. In this section we address the issue of implementing the algorithms in order to build systems.

One way to implement the communication support consists in using monolithic solutions, in which all protocols are included in a single non-modular software component. In some cases, one might obtain performance gains by having a tightly coupled implementation of a protocols suite. However, monolithic solutions cannot be easily adapted to match the application requirements. Therefore, monolithic solutions are contrary to the approach defended in this paper.

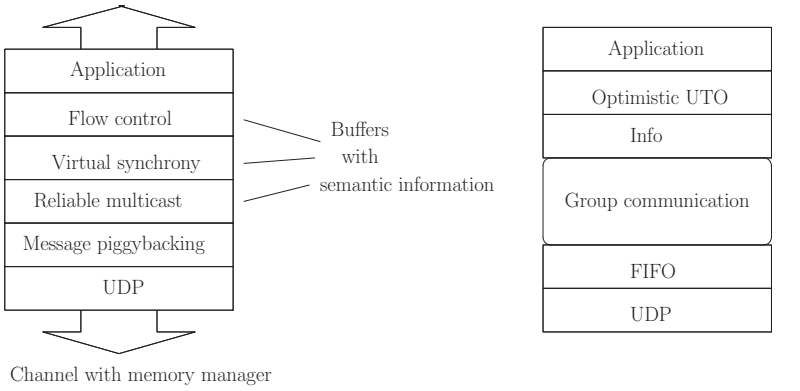
On the other hand, it is desirable to rely on the composition of modular protocols to build an adaptive communication support infra-structure. The DIALNP group is building a modular framework to support the composition and execution of modular protocol components called *Appia* [25]. In the following paragraphs we summarize the most relevant characteristics of the *Appia* system.

The *Appia* system is a protocol composition and execution framework developed in Java. Each *Appia* module is a *layer*, a micro-protocol that ensures a given set of properties. Layers are independent and can be combined to create powerful protocol stacks. Each vertical composition of layers defines a given quality of service (QoS). It is possible to create one or more *channels* offering the same quality of the service. To each channel is associated a stack of *sessions*: there is a session for each protocol layer in the stack. The purpose of session objects is to store the state required for the execution of its layer. For instance, a layer that implements a FIFO ordering protocol needs to preserve a sequence number in the session object. The *Appia* system allows the same session object to be shared by different channels. Using the example of the FIFO layer, it is possible to create channels that have independent sequence numbers or that share the same sequence number[2].

The interaction among layers is performed through the exchange of events. Each event has a type and each layer declares the types of events that it creates and that it is interested in processing. The system optimizes the flow of events in the protocol stack by ensuring that the events are only delivered to the layers interested in receiving them.

The use of the *Appia* system to implement the algorithms described in the previous sections has the advantage of supporting the reuse of other existing protocols, such as transport, consensus, FIFO, and failure detection protocols.

[2]An important feature of the *Appia* system that we do not discuss in the current paper is the possibility to express inter-channel constraints using the session sharing mechanism. For instance, it is simple to create a set of channels with different properties but that respect causal order among them. This sort of protocol composition is discussed in detail in [37].



a) Semantic reliability **b) Optimistic total order**

Figure 2: *Appia* protocol compositions.

It allows also an implementation of a protocol to be changed without requiring any modification in the remaining protocols of the stack. Furthermore, it is possible to add or remove properties (layers) to a given protocol stack in a very flexible manner (for instance, by inserting or removing a logging layer). Figure 2 illustrates the protocol compositions used to implement the algorithms described in the previous section.

5.1 Discussion

The implementation of a communication infra-structure adapted to the application does not require the use of custom made monolithic solutions. On the contrary, the use of protocol composition techniques may give to the application the tools needed to adapt the communication support to a specific operational envelope.

6 Applications

In this paper we use database replication as a case study to illustrate the application of the abstractions, algorithms and systems discussed in the previous sections. In particular, we use two different types of databases with different requirements:

- Small scale in-memory databases, such as the ones used to maintain the state of multi-user game servers.
- Persistent transactional object-oriented databases.

The replication of each of these types of databases is addressed in the following paragraphs.

6.1 Replication of a Game Server database

Most implementations of multi-user games are based on a centralized server that represents a single point of failure. A practical way of making this architecture fault-tolerant consists in using a primary-backup approach to replicate the server. Typically, the information that needs to be copied from the primary to the backup(s) is a set of registers with the state of the several objects managed by the game server (avatars, vehicles, items, missiles, etc).

This collection of registers can be described as a small in-memory database. This sort of database is characterized by a high load of updates (corresponding the the actions of the several players). Since there are frequent updates to the database, there is a high throughput of updates from the primary to the backups. Therefore, the backups can easily become a bottleneck in the system, if their performance is not good enough to keep up with the speed of the primary server.

An interesting feature of the access patterns to these sort of databases is that there is a significant amount of locality in the access to each register. This type of pattern corresponds to the movement of an avatar or vehicle by a player. Therefore, we can expect a significant amount of obsolescence in the traffic from the primary to the backups. This means that semantic reliability can be used to improve the performance of such replicated architecture.

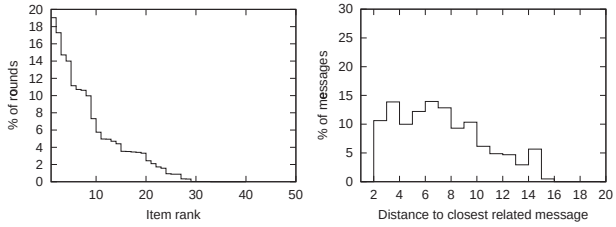
To validate this assumption we have used a concrete case-study to extract experimentally real access patterns to a game server database. To this goal, we have instrumented a public domain implementation of the Quake [20] multi-user game.

Figures 3a and 3b illustrate the obsolescence pattern observed in this application. It is possible to observe that some items are updated much more frequently than others. It is also possible to observe that the distance between related messages is relatively small (usually, below 15 messages): this is an indication that performance gains can be achieved with buffers of moderate dimension. Using simulations, we have estimated the advantages of applying semantic reliability to this application. These are illustrated in Figure 3c: while the throughput of a reliable multicast protocol drops if the backup replica experience delays, the usage of semantic reliability allows to sustain the input throughput even if the backup exhibits delays in the order of $30ms$. A prototype of the semantic reliable multicast protocol using the *Appia* system is currently being implemented [7]. There is also an on-going effort to apply this approach to other multi-user games, such as the Microsoft's Flight Simulator.

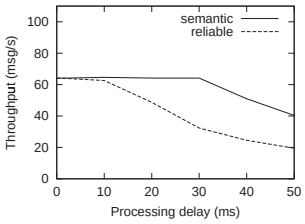
6.2 Replication of Transactional Object-Oriented Databases: the GlobData System

The GlobData [34] system is a *middleware* solution to support the replication of object-oriented databases. The system is being developed by an European consortium in the context of an IST project. The complete architecture includes many components, such as OQL compilers, interfaces to an underlying relational database store, stub compilers, etc [35]. The system also supports several replication strategies [26, 34].

The complete description of the GlobData system is outside the scope of this paper (the interested reader can refer to [34]). Here we briefly summa-



(a) Frequency. (b) Distance.



(c) Throughput.

Figure 3: Obsolescence and performance using Quake.

rize one of the replication algorithms, which is an adaptation of the algorithm described in [22] to the transactional model of the GlobData system. Concurrency control is performed combining the use of local clocks and the following coordination procedure:

1. All operations of a given transaction are executed locally in the process where the transaction was initiated (the delegate process). During its execution, the transaction obtains read and write locks on accessed data items (before being written, an object must be read).

2. When the transaction tries to commit, the write set is disseminated using an atomic broadcast primitive.

3. When a transaction is delivered by the atomic broadcast primitive, all processes try to acquire write locks for all objects in the transaction's write set. During this step, the transaction may be required to wait for some locks. Other transactions that have read locks on the objects are aborted (in this case, the delegate process informs all other processes by sending an abort message using an uniform broadcast primitive). When the delegate process obtains all the write locks, it disseminates an commit message using a uniform broadcast primitive. This phase is called the certification phase.

4. When a process receives the commit message, all the updates are applied to the database and the locks are released. When a node receives an abort message it releases the locks detained by the transaction without applying the updates to the database.

In this system, the optimistic total order protocol described in Section 4.2 is used to parallelize the following actions: *i)* the transmission of the commit message; *ii)* the termination of the total order protocol. When a node receives the estimate of the final order it executes immediately the transaction certification procedure described above. If the certification procedure indicates that the transaction cannot be committed, the transaction is aborted without waiting for the final total order. On the other hand, if the certification accepts the transaction, the commit message is sent immediately (even if the final total order has not been received yet). All processes confirm the transaction when: *i)* receive the commit message *and ii)* the definitive total order is the same as the estimate order. In this way, the system can save a communication step in the most frequent case.

6.3 Discussion

These results validate the previous observations, demonstrating that is in fact possible to improve the performance of distributed applications through the resource to configurable and adaptive solutions.

7 Conclusions and Future Work

This paper has addressed the problem of providing effective communication and coordination support for complex distributed applications. The goal is to achieve solutions that are both efficient and can be re-used in different contexts. It is also desirable that these subsystems offer simple and elegant abstractions. We have shown that to achieve this goal, configuration and adaptiveness has to be considered at all levels of system development, namely: abstractions, algorithms and implementations.

At the abstractions level, we have shown that is possible to depart from generic abstractions, with applicability to a large domain of systems, and augment these abstractions to improve expressiveness without limiting the applicability domain. At the algorithmic level, we have shown that it is possible to derive adaptive algorithms that can be used in diverse operational conditions. At the implementation level, we have emphasized the importance of using protocol composition and execution frameworks that are able to preserve the adaptiveness achieved at the upper levels. These frameworks allow the applications to configure the communication and coordination support according to the operational envelope (load, network topology, etc). Finally, these results were validated by applying them to concrete applications. The paper reported two concrete on-going experiments by the DIALNP research group.

The approach defended in this paper allows to combine the advantages of using powerful abstractions, that encapsulate the complexities of distributed computing, with the efficiency needs of modern applications. This advantage derives from considering adaptiveness as a central quality, that must be addressed at all levels of the system development. The DIALNP group is currently starting to apply this methodology to the development of communication and coordination support in middleware for mobile and ad hoc networks [31].

Acknowledgments

The work reported here was performed in collaboration with the members of the DIALNP group and with the partners of the national and international projects in which the group is involved.

References

[1] Y. Amir and C. Tutu. From total order to database replication. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, number CNDS-2001-6, July 2002.

[2] Ö. Babaoğlu and S. Toueg. Understanding non-blocking atomic commitement. In S. Mullender, editor, *Distributed Systems (2nd edition)*, chapter 6. Addison-Wesley, 1993.

[3] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

[4] K. Birman. The process group approach to reliable distributed computing. *Communications of the ACM*, 36(12):37–53, December 1993.

[5] K. Birman. A review of experiences with reliable multicast. *Software Practice and Experience*, 29(9):741–774, July 1999.

[6] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, 1999.

[7] N. Carvalho, J. Pereira, and L. Rodrigues. Concretização de protocolos com fiabilidade semântica. In *Actas da 5ª Conferência sobre Redes de Computadores*, Faro, Portugal, September 2002.

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

[9] D. Cheriton and D. Skeen. Understanding the limitations of causally and totally ordered communication. In *Proceedings of the 14th Symposium on Operating Systems Principles*, Asheville, NC, USA, December 1993.

[10] D. Clark and D. Tennenhouse. Architectural considerations for a new generation of protocols. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 200–208, Philadelphia, PA, September 1990. ACM.

[11] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *Journal of the ACM*, 32(4):841–860, October 1985.

[12] D. Gifford. Weighted voting for replicated data. In *Proc. of the 7th ACM Symposium on Operating System Principles*, pages 150–162, USA, December 1979.

[13] A. Gopal and S. Toueg. Inconsistency and contamination. In Luigi Logrippo, editor, *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing*, pages 257–272. ACM Press, August 1991.

[14] J. Gray, P. Helland, P. O'Neal, and D. Shasha. The dangers of replication and a solution. In *Proc. of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 173–182, Montreal, Quebec, Canada, June 1996.

[15] R. Guerraoui and A. Schiper. Software-based replication for fault tolerance. *IEEE Computer*, 30(4):68–74, April 1997.

[16] R. Guerraoui and A. Schiper. The generic consensus service. *IEEE Transactions on Software Engineering*, 27(1):29–41, January 2001.

[17] K. Guo. *Scalable Message Stability Detection Protocols*. PhD thesis, Cornell University, Computer Science, May 1998.

[18] V. Hadzilacos and S. Toueg. A modular approach to fault-tolerant broadcasts and related problems. Technical Report TR94-1425, Cornell University, Computer Science Department, May 1994.

[19] J. Holliday, D. Agrawal, and A. El Abbadi. Using multicast communication to reduce deadlock in replicated databases. In *Proc. of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS2000)*, Nürnberg, Germany, October 2000.

[20] Id Software Inc. Quake homepage. http://www.quake.com.

[21] I. Keidar and S. Rajsbaum. On the cost of fault-tolerant consensus when there are no faults - a tutorial. Technical report, MIT Laboratory for Computer Science, 2001.

[22] B. Kemme and G. Alonso. A suite of database replication protocols based on group communication primitives. In *Proc. of the 18th International Conference on Distributed Computing Systems (ICDCS)*, The Netherlands, May 1998.

[23] B. Kemme and G. Alonso. Don't be lazy, be consistent: Postgres-R, a new way to implement database replication. In *Proc. of the 26th International Conference on Very Large Databases*, Cairo, Egypt, September 2000.

[24] L. Lamport. Time, clocks and the ordering of events in a distributed system. *CACM*, 21(7):558–565, July 1978.

[25] H. Miranda, A. Pinto, and L. Rodrigues. Appia, a flexible protocol kernel supporting multiple coordinated channels. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, pages 707–710, Phoenix, Arizona, April 2001. IEEE.

[26] F. D. Muñoz, L. Irún, P. Galdámez, J. M. Bernabéu, J. Bataller, and M. C. Bañuls. Globdata: Consistency protocols for replicated databases. In *Proc. of the IEEE YUFORIC'2001*, pages 97–104, Spain, November 2001. ISBN 84-9705-097-5.

[27] F. Pedone and A. Schiper. Optimistic atomic broadcast. In *Proceedings of the 12th International Symposium on Distributed Computing (DISC'98)*, 1998.

[28] J. Pereira, L. Rodrigues, and R. Oliveira. Semantically reliable multicast protocols. In *Proceedings of the Nineteenth IEEE Symposium on Reliable Distributed Systems*, pages 60–69, October 2000.

[29] J. Pereira, L. Rodrigues, and R. Oliveira. Enforcing strong consistency with semantic reliability: Sustaining high throughput in reliable distributed systems. In P. Ezhilchelvan and A. Romanovsky, editors, *Concurrency in Dependable Computing*. Klywer Academic Publishers, 2002.

[30] J. Pereira, L. Rodrigues, and R. Oliveira. Reducing the cost of group communication with semantic view synchrony. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 293–302, Washington (DC), USA, June 2002.

[31] L. Rodrigues and R. Prakash (eds). Report on the ERCIM-NSF workshop on middleware for mobile systems. Technical report, ERCIM-NSF Strategic Workshops, 2002.

[32] L. Rodrigues, H. Fonseca, and P. Veríssimo. Totally ordered multicast in large-scale systems. In *Proceedings of the 16th International Conference on Distributed Computing Systems*, pages 503–510, Hong Kong, May 1996. IEEE.

[33] L. Rodrigues, R. Guerraoui, and A. Schiper. Scalable atomic multicast. In *Proceedings of the Seventh International Conference on Computer Communications and Networks (IC3N'98)*, pages 840–847, Lafayette, Louisiana, USA, October 1998. IEEE.

[34] L. Rodrigues, H. Miranda, R. Almeida, J. Martins, , and P. Vicente. The globdata fault-tolerant replicated distributed object database. In *Proceedings of the First Eurasian Conference on Advances in Information and Communication Technology*, pages 426–433, Teheran, Iran, October 2002.

[35] L. Rodrigues, H. Miranda, R. Almeida, J. Martins, and P. Vicente. Strong replication in the globdata middleware. In *Proceedings of the Workshop on Dependable Middleware-Based Systems*, pages G96–G104, Washington D.C., USA, June 2002. IEEE. (Suplemental Volume of the 2002 Dependable Systems and Networks Conference, DSN 2002).

[36] A. Sousa, J. Pereira, F. Moura, and R. Oliveira. Optimistic total order in wide area networks. In *Proceedings of the 21th IEEE Symposium on Reliable Distributed Systems (SRDS'02)*, page (to appear), Osaka, Japan, October 2002.

[37] S. Teixeira, P. Vicente, A. Pinto, H. Miranda, L. Rodrigues, and A. Martins, J. Rito-Silva. Configuring the communication middleware to support multi-user object-oriented environments. In *Proceedings of the International Symposium on Distributed Objects and Applications (DOA)*, page (to appear), Irvine (CA), USA, October 2002.

[38] R. van Renesse. Causal controversy at Le Mont St.-Michel. *ACM Operating Systems Review*, 27(2):44–53, April 1993.

[39] P. Vicente and L. Rodrigues. An indulgent uniform total order algorithm with optimistic delivery. In *Proceedings of the 21th IEEE Symposium on Reliable Distributed Systems (SRDS'02)*, pages 92–101, Osaka, Japan, October 2002.