The Impact of Group Communication Paradigms on Groupware Support*

François J.N. Cosquer
fjnc@inesc.pt

Paulo Veríssimo
paulov@inesc.pt

INESC†- IST‡
Lisbon, Portugal

Abstract

Group Communication Paradigms play an important role in the management of group activities, such as reliable information diffusion in support of replication and cooperation. They consequently provide an appropriate framework for groupware applications support. To meet the new challenges posed by large scale distributed settings like the Internet, a new generation of group communication systems has emerged, built on the lessons learnt from early prototypes. This paper presents new services/features proposed by this new generation of systems and shows their relevance in the context of groupware support. Experiments with a cooperative application demonstrate how the features of a group-based system like NAVTECH *can be used to enhance the functionality of groupware applications.* NAVTECH *is a modern group communication and activity support system developed at INESC.*

1 Introduction

The rapid growth of interconnected networks has led to the emergence of a computing infrastructure with enormous potential for hosting a wide range of distributed applications. Amongst the most promising are those oriented toward multi-user collaboration, usually referred to as Computer Supported Cooperative Work (CSCW) or groupware.

Early experiments with groupware applications have shown that group activities management is a key issue [11]. Cooperative applications rely heavily on fast and reliable dissemination of information among all users and, thus, put strong requirements on the supporting platform, especially on the communication subsystem. This suggests a beneficial relationship between what may be provided by a group-oriented system and the groupware applications requirements. Selecting key functionality of the groupware platform, we have shown that groups can be identified at different levels of the architecture and established a mapping to group support services [11].

Amongst the most current groupware applications requirements are the following:

- Need for multicast diffusion: multiple destinations sharing a common semantics, such as reliable delivery, causal or total order, etc.
- Need for membership protocols: dynamic joins 'n' leaves, collaboration awareness, etc.
- Need for failure detection: deciding on host failures, network partitions, too slow processes, etc.

Implementations of systems providing the group communication paradigms have proven to give satisfactory results in LAN-scale environments [5, 2, 17, 18]. Groupware systems have been implemented using those facilities [23, 25].

However, in the context of Large-Scale Distributed Computing Systems (LSDCS) such as those based on the Internet, current solutions are ill-suited. One of the reasons is because the high variance and the unpredictability of communication delays drastically increase the probability of incorrect failure detection. That is, whereas LSDCS exhibit physical partition failures, inaccurate detection may also lead to virtual partitions. In other words, there is a tradeoff between the application liveness and the accuracy of failure suspicion, thus in a LSDCS environment, enforcing agreement in bounded-time is more likely to lead to erroneous suspicion [20].

Cooperative applications for large scale systems are in high demand because they lessen the need for a collection of people to commute to a single location. Cooperative applications have however different liveness

*This work was partially supported by the CEC, through Esprit Project BR 6360 (Broadcast).

†Instituto de Engenharia de Sistemas e Computadores, R. Alves Redol, 9 - 6° - 1000 Lisboa - Portugal, Tel.+351-1-310000.

‡Instituto Superior Técnico - Technical University of Lisbon

0-8186-7125-4/95 $04.00 © 1995 IEEE

and accuracy requirements from more traditional distributed applications. When a cooperative session is taking place, it is not acceptable to remove processes too promptly. This is because cooperative applications exhibit attributes such as involvement of end-users, complex event scheduling and possibly tedious start-up procedure.

The aforementioned attributes have important implications as to how problems such as partitions, slow links and host failures are treated by the application programmer. Without adequate support, this has proven to be a hard task. In consequence, the motivation for the present paper is evident: discussing the impact of group communication paradigms in groupware support. We show how a new generation of systems propose to tackle problems linked with scale and their applications in groupware support.

2 Using the Flexibility of Modern Group Communication Systems

2.1 The NAVTECH Platform

From the wide experience learnt from use and design of group communication systems over the years, a new generation of systems is emerging [1, 4, 19]. These group communication systems are proposing new services and defining interfaces which better model the applications needs. This design and implementation effort has resulted in new structuring of systems.

For example, the Horus system is organized as a stack of layers, each implementing a certain functionality. This flexibility allows application programmers to compose the stack of layers as a way to better model the application requirements. Furthermore, all layers share a common interface, the "Horus Common Group Interface".

Relacs, proposes an alternative membership protocol. The quasi-strong membership allows two concurrent views that result from the partitioning of a common view to overlap only if one is a proper subset of the other.

Other work also includes a configurable membership framework [16]. The desired membership service is achieved by associating and configuring micro-protocols to form a composite protocol.

The Navigators group at INESC has been working for some years in the area of group communications. More recently, the group initiated, based on former experience [18], the NAVTECH project. NAVTECH is a modern group-based communication subsystem which is intended to support the development of reliable applications over large scale networks. It offers a range of communication services— such as reliable communication based on flexible failure suspectors and partition-resilient group membership management— complemented with activity support services— such as replication management protocols [22, 21]. These services are offered through a flexible interface, allowing client-server remote operations, message dissemination, and peer-to-peer conversations.

As part of the validation path for our design options, we are developing a groupware platform. The services of NAVTECH we are concerned with in this paper are the failure suspector and the membership manager. We show how the functionality they provide can be used with respect to support cooperative applications, by reporting some of the experiments we performed.

2.2 The failure detection problem

Large-scale communication infrastructures suffer from problems inherent to the nature of the current technology, topology, and, of course, sheer dimension and distance. The two most obvious are: they are prone to partitions; and they are asynchronous. Partitions mean that sites may occasionally be unable to communicate with each other. Asynchronism means that bounds cannot be defined on communication delays.

The combination of the two yields a known problem in the distributed systems community, whose foundation lies in the FLP impossibility result [14]. This result informally states that one cannot ensure progress of a distributed computation on an asynchronous system. The problem can be overcome by using what has been called failure detectors, which define an artificial threshold between a slow and a failed component. This brings us to the problem we named in the beginning: the possibility of unjustful failure detection. In what concerns communication in the presence of partitions, this is essentially because one cannot tell a slow site from a broken link. In consequence, the best we can have are *failure suspectors* (FS) of several semantics, whose reliability is inversely proportional to their precision [8]. We have proposed to configure FS in order to better model application requirements. The approach is detailed further ahead.

2.3 The membership problem

Usually, the output of failure suspectors goes to the group membership module, which ensures consistent joins and leaves for all participants with respect to communication. One such semantics is known as virtual synchrony [6]. In presence of partitions, this service has been precisely defined in [24] and termed *view-synchronous communication*: view changes are totally ordered with bags of messages which may have no mutual order. View-synchronous communication is

based on a *strong-partial membership* service. Informally, this means that at any time in the system it is guaranteed that concurrent views are non-intersecting.

Triggering membership from failure suspicion may cause instability problems in a large-scale network, since participants may have asymmetric and non-transitive connectivity, making the group toggle between unstable membership states. The effect on a collaborative application would be disastrous, with participants coming in and out at an unbearable rate. NAVTECH performs dynamic rerouting to alleviate instability, as detailed in [15].

We now elaborate on the two fundamental aspects involved in groupware support, namely: the configuration of failure suspectors and the partition service based on strong partial membership.

2.4 Configuring Failure Suspectors

Configuring a Failure Suspector to better model the application requirements means:

- specifying connectivity: having control on the kind of test used to monitor the links.
- specifying triggering decisions: having control over the decision on when to report a node crash and initiate a membership agreement protocol.

In NAVTECH, the Failure Suspector is designed to evaluate a number of relevant operational parameters. Currently supported parameters are: roundtrip delay, throughput and error rate. The application can configure the failure suspector, indicating which parameters must be measured and their acceptable values. Each parameter is characterized by a set of variables, as illustrated in Table 1, namely:

Sample rate: the maximum interval at which the parameter needs to be evaluated.

Threshold: the parameter maximum acceptable value.

Weight: a measure of the parameter's relative importance. A weight of 0 indicates that the parameter is not taken into account.

TE (Threshold Exceeded): the number of samples at which the parameter value has exceeded its threshold.

Value: the parameter's current value.

At every node, a complete set of variables for a given parameter is kept for every other reachable node in the group. We denote the variable V associated with parameter P kept at local node l about some remote node r by $V_l^r(P)$. For instance, $\text{Value}_l^r(roundtrip)$ denotes the roundtrip delay between nodes l and r as measured at l. All the variables may be set and/or read by the application, through a management interface.

The overall connectivity to a remote node is measured by an index, named *Disturbance Index* (DI), which is a weighted average of the TE of all variables. That is:

$$\text{DI}_l^r = \sum_i \text{Weight}_l^r(i) * \text{TE}_l^r(i) / \sum_i \text{Weight}_l^r(i)$$

The DI is computed periodically, as specified by its sample rate. It indicates how often the most important parameters have exceeded the defined thresholds. After the computation of the DI, TE variables are reset, and incremented from zero until the next computation of DI. A very low DI value means that the connectivity is within satisfactory thresholds. A high value means very poor connectivity.

The DI value is used to build a *Local Connectivity Vector* (LCV_l), an array of booleans that indicates whether the Disturbance Index has exceeded its threshold. Whenever the DI_l^r threshold for a given node r is exceeded, that node is suspected, and $\text{LCV}_l[r]$ is set to false.

In order to further simplify the configuration we use the personality library to model cooperative application requirements. More details are given in [10].

Specifying triggering decision: Usually once a node r is suspected by a node l, an agreement protocol is initiated by l resulting in a new view delivery. This may lead to a virtual partition if the suspicion was incorrect.

We have shown above how to tailor the FS module to perform tests in accordance to application semantics. In order to further reduce the probability of incorrect failure detection, Failure Suspectors exchange periodically their Local Connectivity Vectors to build a *Global Connectivity Matrix*, or GCM. Note that LCV is a vector of bits thus, it can be piggybacked in normal traffic with negligible overhead. The GCM indicates how each node perceives its connectivity to every other node. If node i has good connectivity to j then $\text{GCM}[i, j]$ is true, and false otherwise. The activation of the membership module is a function of GCM values. This function is also configurable using a *quorum* based approach.

An integer value, called the *Suspicion Weight* (SW_i) is associated with every node i. Suspicion weights reflect the relative importance of each user

Parameter	Sample rate	Threshold	Weight	TE	Value
roundtrip delay	1m	20s	2	0	20ms
throughput	5m	1Mbits/s	1	1	2 Mbits/s
...	.	.	.	.	.
PI	1s	3	1	0	0.4

Table 1: Variables associated with operational parameters

also known as "social role" in the context of cooperative applications. Weights are also defined and used for partition processing strategies purposes.

Another integer, called the *Global Suspicion Quorum* (GCQ), indicates when a node is said to be globally suspected. In this case only, the membership protocol is activated. More precisely, the membership is activated for a node j if the following condition is true:

$$\sum(i|\mathrm{GCM}[i,j] = \mathrm{false})\mathrm{SW}_i \geq \mathrm{GCQ}$$

The values of Suspicion Weights and of the Global Suspicion Quorum can be set by the application to define different membership triggering conditions. For instance, membership for a given node can be triggered only when this node is suspected by a majority of nodes, or by a selected node. More sophisticated triggering conditions are being studied and are reported in [15].

2.5 Partition Processing

After a node has been marked crashed, the membership protocol is initiated. As a direct consequence of the aforementioned impossibility result, the new membership may be influenced by erroneous suspicions. This may lead to "virtual" partitions which may eventually diverge resulting in a inconsistent system state.

Traditional approaches like the one found in the ISIS system [7], consist in identifying a majority partition and only allow this majority partition to make progress. A number of applications including groupware applications would benefit if multiple partitions were allowed to operate simultaneously. However, a purely optimistic behavior is hard to reconcile when the system merges again. Some form of rules that control progress of minority partitions are required. Those are normally influenced by the typical semantics of the applications in mind.

Specifying and implementing this system behavior is the topic of ongoing research. This approach can assume interesting aspects, given group-oriented support [20, 13, 16]. In the groupware area, our current thinking is towards relying on a strong-partial membership service at the communication level, which allows view-synchronous communications, as the baseline consistency paradigm. However, instead of providing primary back-up semantics on top of that, we center our support around two axis:

- A Partition Support Service (PPS) which is a generic framework for partition processing.
- A set of generic groupware services (based on the above).

PSS: A generic framework for partition processing: our approach is based on partition typing, split and merging rules. Each partition is associated a type at runtime. This type is configured by the application programmer and is based on weights. When a partition shrinks or grows, its new type is evaluated and corresponding split and merging rules are automatically invoked. This allows the application programmer to define the consistency control policy thus the behavior of the application. More details are given in [12].

Generic groupware services: synchronized applications have a number of functional modules which do not depend on the structure nor on the content of the shared data. This includes floor control policies used for mediating access to shared workspace entities such as audio or video channels, access to shared telepointers over some text or graphical windows, etc. Other modules which are turned towards collaboration awareness and accessed in an interjection-like manner, like shared console windows or subject-based bulletin boards, do not depend on the workspace content either. This is also true for collaboration awareness modules which provide feedback such as activity or state of mind status (i.e idle, unreachable, approving, puzzled, etc.). Once partioned, all these modules should remain accessible following a specified policy.

3 Application and Experience

3.1 NGTool

NGTool [3] is a groupware application that implements the Nominal Group Technique, a structured

behavioral science technique for group generation of ideas and consensus forming.

The NGTool runs on Unix workstations over the Internet and uses X Windows and the Motif toolkit. The tool is based on direct manipulations of graphic and textual objects in both private and public spaces. The differences between the non-computer supported technique and the computer counterpart are current subjects of research. NGTool is also used as a testbed for the groupware support discussed in this paper.

The NGTool does not provide face-to-face interactions neither video nor audio counterparts. Regardless of their necessity for the successful development of the group work, the lack of this continuous media channels renders the task of audience monitoring and partition handling more difficult. In order to proceed smoothly, new techniques are required. NGTool provides therefore an adequate testbed for evaluating the benefits of group communication paradigms.

We now demonstrate how, using configurable failure suspectors and the partition support service, current technological limitations can be partially overcome.

3.2 Impact On Collaboration Awareness

We are experimenting with incorporating feedback in various building blocks, such as audience monitoring, telepointer facility and concurrency control modules. The first experiments of tailored Failure Suspectors and Partition Support Service gave satisfactory results.

Tailored Failure Suspectors are used to provide collaboration awareness enhancement. This is achieved by intercepting the output of the FS and providing connectivity clues to the users using new feedback techniques. An example is given in Figure 1 where the **roundtrip delay** output is used to represent some distance between users. It is then displayed for audience monitoring purposes.

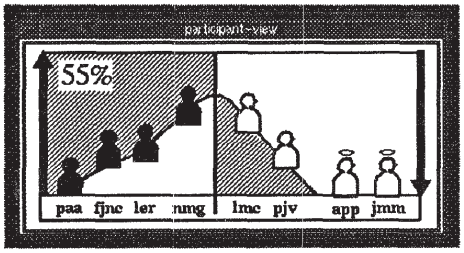

Figure 1: Adaptable Audience Monitoring

The UI analogy for the audience monitoring mechanism is inspired by a hill, where people go up and down but always getting more distant from the viewer. The distance between the viewer and the other participants is represented by a line going from left to right. Participants are aligned by increasing magnitude from left to right. Unreliable participants, which are displayed to the right of the sliding line, are also aligned by increasing magnitude but the measurement is displayed inverted, i.e. with a line that is decreasing for increasing distances. This design option also has the benefit of saving display space.

A percentage value is also displayed on the top left corner of the UI. This is intended to represent a global distance measurement using a single value and, therefore, is a function of the feedback delays of both reliable and unreliable participants. It is implemented using the GMC, the Global Connectivity Matrix, local to each node, as described earlier.

Another application which uses the GMC which also serves the audience monitoring function is the Membership Level Control Panel. The idea was to use the information provided by the GMC in order to provide hints about global connectivity. This was seen as a complementary feedback loop which complements the distance viewer. The entry corresponding to the global connectivity situation of each node is filtered to a 3 levels value. The values are then displayed as shown on Figure 2. The icons are filled using the following convention: full icon for high global connectivity, empty for low global connectivity and half for intermediate values. This allows the users have an overview of the situation at a glance.

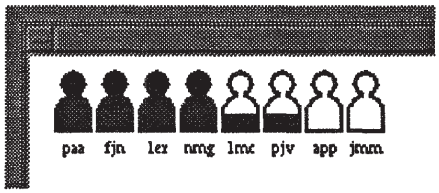

Figure 2: Membership Level Control Panel

In order to further improve the level of support we are experimenting with new feedback techniques which provide partition level information. The idea consists of combining the partition typing information provided by the PSS with standard cooperative modules. To illustrate our claim we have selected here two types of mechanisms frequently used during cooperative session: gesture support (telepointer) and a generic concurrency control mechanism.

Telepointers have been designed to be manipulated

by one participant at a time in order to point or get attention on a specific area or item of the shared workspace. The principles adopted for the telepointer are illustrated on the top part of Figure 3. It uses a "Motion-stop" mechanism. The Motion-stop is distributed to other participants using the reliable multicast primitives provided by the underlying group support. This allows each participant to execute a "Motion-replay" as show on top of Figure 3. The meter level will reflect the reception of feedback from the group participants. When feedback from all participants is received, the meter is displayed full for a short period and then vanishes.

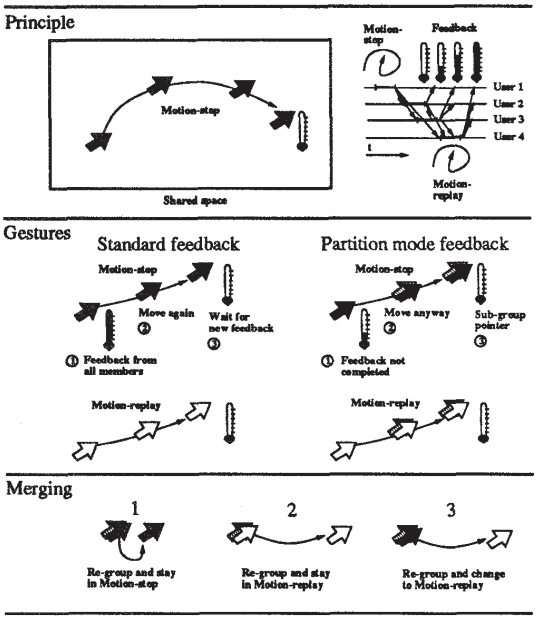

Figure 3: The Telepointing Mechanism

After a Motion-stop, the user should refrain from moving the telepointer until knowing that all participants have had the gesture replayed (case middle left on Figure 3). However, if the meter does not reach the top, due to one or several participants being unreliable, the user can still move the telepointer (case middle right on Figure 3). In this case, the system will automatically enter the partition mode. The users will be notified immediately, seeing a shadowed telepointer. Note that if excluded participants have not crashed they also have created their sub-group and entered partition mode. When the participants from two partitions are able to communicate again, the system will terminate the partition mode and merge (see Figure 3 bottom). As explained before, the PSS automatically invokes the pre-defined partition merging rules. In this case, this results in defining the owner and the position of the telepointer.

We also have implemented a generic concurrency control which keeps manipulations of objects separated from their graphical representation. Further details of our experiments are reported in [9].

4 Building a groupware platform

In parallel with the work in cooperation with the NGTool team, we are developing a groupware platform which regroups all generic services and mechanisms discussed above.

Many attempts to assist groupware applications developers have been tried. These range from models of multi-user activity to low-level system support. The services common to applications are usually grouped and made accessible through so-called groupware toolkits or platforms. Our implementation is based on the Failure Suspector Configuration module and the Partition Support Service which are both implemented on top of NAVTECH. The overall system architecture is depicted on Figure 4.

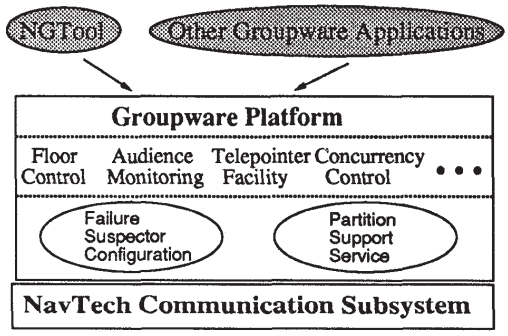

Figure 4: Overall system architecture

Using groupware platform services has several advantages, including: reducing the implementation efforts and allowing rapid prototyping; enables a clear separation between platform-provided mechanisms and application-specific policies; allowing service optimization in an application independent way.

From our experience so far, we believe that our group communication system approach presents a high potential for satisfying groupware applications requirements. We plan to continue our experiments

to further substantiate this belief.

5 Conclusion

This paper has discussed the impact of group communication paradigms on groupware support. The relevant aspects of the functionality provided by modern group communication systems were pointed out. The configuration of failure suspectors and the specification of the agreement triggering conditions were described. Furthermore, a partition processing framework was presented.

Using the NAVTECH platform, examples taken from a cooperative application were used to illustrate the potential of the group-based support. It was shown how this support, associated with new feedback techniques, can enhance cooperative environments.

We are developing a generic groupware platform which is aiming at demonstrating the suitability of group communication systems in supporting future groupware applications. Further work is underway to confirm our initial experiments. We need to ensure that the complexity inherent to the various configurations proposed can be partially hidden from the application programmer. Overlooking this issue could be prejudicial to the use of groupware support technologies.

A Acknowledgements

The authors are grateful to the NGTool team especially to Pedro Antunes and Nuno Guimarães for their excellent contribution over the last few months. We also acknowledge the contribution of Carlos Almeida and Luís Rodrigues who have greatly influenced some of the ideas presented in this paper. The work would not have been possible without the work of Jorge Frazão, who implemented the NAVTECH failure suspector module.

References

[1] Y. Amir, D. Dolev, S. Kramer, and D. Malki. A Communication Sub-System for High Availability. In *Proc. 22nd International Symposium on Fault-Tolerant Computing*, pages 76–84, July 1992.

[2] Yair Amir, Danny Dolev, Shlomo Kramer, and Dalia Malki. Transis: A Communication Sub-System for High-Availabiliy. In *Digest of Papers, The 22nd International Symposium on Fault-Tolerant Computing Systems*, pages 76–84. IEEE, 1992.

[3] Pedro Antunes and Nuno Guimarães. Structuring Elements for Group Interation. In *Second Conference on Concurrent Engineering, Research and Applications (CE95)*, August 1995. (To appear).

[4] Ozalp Babaoglu, R. Davoli, L. A. Giachini, and M. G. Baker. Relacs: A Communication Infrastructure for Constructing Reliable Applications in Large-Scale Distributed Systems. In *Proceedings of the 28th Hawaii International Conference on System Sciences*, pages 612–621, January 1995.

[5] Ken Birman and Robert Cooper. The ISIS Project: Real Experience with a Fault Tolerant Programming System. Technical Report TR90-1138, Cornell University, Ithaca, NY 14853, USA, July 1990.

[6] Kenneth P. Birman and Thomas A. Joseph. Exploiting Virtual Synchrony in Distributed Systems. In *11th Symposium on Operating Systems Principles*, pages 123–138, November 1987.

[7] Kenneth P. Birman and Robbert van Renesse. *Reliable Distributed Computing with the Isis Toolkit*. IEEE Computer Society Press, 1994.

[8] T.D. Chandra and S. Toueg. Unreliable Failure Detectors for Asynchronous Systems. Technical Report TR 91-1225, Cornell University, Department of Computer Science, Ithaca, August 1991.

[9] François J.N. Cosquer, Pedro Antunes, Nuno Guimarães, and Paulo Veríssimo. Adaptive Synchronous Cooperation over Large Scale Networks. Technical Report RT/95, INESC, Rua Alves Redol 9, March 1995.

[10] François J.N. Cosquer, Luís Rodrigues, and Paulo Veríssimo. Using Tailored Failure Suspectors to Support Distributed Cooperative Applications. Technical Report RT/95, INESC, Rua Alves Redol 9, April 1995.

[11] François J.N. Cosquer and Paulo Veríssimo. Survey of Selected Groupware Applications and Supporting Platforms. Technical Report RT-21-94, INESC, Rua Alves Redol 9-6°, 1000 Lisboa, Portugal, September 1994. (Also available as BROADCAST Technical Report [2nd year - Vol 1]).

[12] François J.N. Cosquer and Paulo Veríssimo. Large Scale Distribution Support for Cooperative Applications. In *Proceedings of the European Research Seminar on Advances in Distributed Systems - ERSADS '95*, April 1995. (Also available as INESC Report AR 2/95).

[13] Danny Dolev, Dalia Malki, and Ray Strong. A Framework for Partitionable Membership Service. Technical Report CS95-4, The Hebrew University of Jerusalem, Jerusalem, Israel, 1995.

[14] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *Journal of the Association for Computing Machinery*, 32(2):374–382, April 1985.

[15] Jorge Frazão, Paulo Veríssimo, and Luis Rodrigues. Enhancing Large-scale Communication with Failure Suspectors and Dynamic Routing. Technical Report RT-95, INESC, Rua Alves Redol 9-6°, 1000 Lisboa, Portugal, April 1995.

[16] Matti A. Hiltunen and Richard D. Schlichting. A Configurable Membership Service. Technical Report TR 94-37, University of Arizona, Tucson, AZ 85721, December 1994.

[17] Norman C. Hutchinson and Larry L. Peterson. The x-Kernel: An Architecture for Implementing Network Protocols. *IEEE Transactions on Software Engineering*, 17(1):64–76, January 1991.

[18] D. Powell, editor. *Delta-4 - A Generic Architecture for Dependable Distributed Computing.* ESPRIT Research Reports. Springer Verlag, November 1991.

[19] R. van Renesse, Ken Birman, Robert Cooper, Brad Glade, and Patrick Stephenson. The Horus System. Technical report, Cornell University, July 1993.

[20] Aleta Ricciardi, Andre Schiper, and Kenneth Birman. Understanding Partitions and the No Partition Assumption. In *Proceedings of the 4th Workshop on Future Trends of Distributed Computing Systems*, pages 354–360, September 1993.

[21] L. Rodrigues, E. Siegel, and P. Veríssimo. A Replication-Transparent Remote Invocation Protocol. In *Proceedings of the 13th Symposium on Reliable Distributed Systems*, October 1994.

[22] L. Rodrigues and P. Veríssimo. Replicated object management using group technology. In *Proceedings of the 4th Workshop on Future Trends of Distributed Computing Systems*, pages 54–61, September 1993. (Also as INESC AR/28-93).

[23] Alain Sandoz, Francois Pacull, and Andre Shiper. Duplex: A Distributed Collaborative Editing Environment in Large Scale. In *Proceedings of the Conference on Computer-Supported Cooperative Work - CSCW'94*, October 1994.

[24] A. Shiper and A. Ricciardi. Virtually Synchronous Communication based on a weak failure suspector. In *Proceedings of the 23rd Int. Conf. on Fault Tolerant Computing Systems*, June 1993.

[25] Jonathan Trevor, Tom Rodden, and Gordon Blair. COLA: A Lightweight Platform for CSCW. In *Proceedings of the 3rd European Conference on CSCW (ECSCW'93)*, pages 15–30, September 1993.