Using Tailored Failure Suspectors to Support Distributed Cooperative Applications[*†]

François J.N. Cosquer
fjnc@inesc.pt

Luís Rodrigues
ler@inesc.pt

Paulo Veríssimo
paulov@inesc.pt

INESC[‡]- IST[§]

Abstract

This paper presents an approach to effectively support cooperative applications using tailored failure suspectors. Using a group communication subsystem, it is shown how failure suspectors can be configured to model the requirements/semantics of cooperative applications thus avoiding ad-hoc system decisions. This approach is highly relevant in the context of large scale distributed systems like the Internet, where communication high variance and unpredictable delays increase the probability of incorrect failure detection. Applications are presented illustrating how failure suspectors are configured and possibly combined with new feedback techniques in order to implement more powerful cooperative environments.

Keywords: *Fault-tolerance, Interconnected networks, Cooperative applications, Failure Suspicion, Large Scale.*

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. These applications rely heavily on fast and reliable dissemination of information to all users and thus, put strong requirements on the supporting platform, especially on the communication subsystem. Unfortunately, asynchronous message-passing computing environments suffer from the FLP impossibility results [11] i.e. it is impossible to differentiate a slow node or communication link from a crashed node.

A solution, known as failure detection, has been developed to encompass the FLP problem. We can see failure detection being composed of two distinct functional blocks:

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

[†]Selected portions of this report were published in the Proceedings of the 7th International Conference on Parallel and Distributed Computing and Systems, Washington D.C., Oct 18-21, 1995

[‡]Instituto de Engenharia de Sistemas e Computadores, R. Alves Redol, 9 - 6º - 1000 Lisboa - Portugal, Tel.+351-1-310000.

[§]Instituto Superior Técnico - Technical University of Lisbon

a failure suspector (FS), which maintains information about which processes may have crashed; and an agreement protocol (often referred to as membership protocol) which forces the system to agree with the failure suspector. This agreed *view* of the system state allows safe progress in the event of process crashes. Implementations of these mechanisms / services have proven to give satisfactory results in LAN environments leading to the emergence of what one might refer to as *group-based systems* [3, 1, 14, 15].

However, in the context of Large Scale Distributed Computing Systems (LSDCS), current solutions are ill-suited 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 FS, thus in LSDCS environment, enforcing agreement in bounded-time is more likely to lead to erroneous suspicion [17][1].

Cooperative applications represent a high demand for large scale systems because they lessen the need for a collection of people to commute to a single location. These applications have different liveness and accuracy requirements than more traditional distributed applications. When a cooperative session is taking place, it is not acceptable to remove processes prematurely. This is because cooperative applications exhibit attributes such as involvement of end-users, complex event scheduling and possibly tedious start-up procedure.

Surprisingly, having surveyed a number of groupware applications and platforms [8], we found little or no concern for scaling and fault-tolerance issues. This is clearly not acceptable if cooperative applications are to become widely used. We also have highlighted the relationship between the requirements of groupware applications and the functionality provided by group-based systems. These high-level considerations have led us to believe that this class of applications would highly benefit from the underlying support if it could:

- Model application connectivity requirements.
- Minimize erroneous process exclusion.
- Provide connectivity clues to users.

The paper describes a pragmatic approach to reach the objectives listed above. It focuses on the practical issues linked with the configuration and usage of the Failure Suspectors (FS). Section 2 briefly presents the motivation for failure suspectors and their role in group-based systems. Section 3 describes how current systems allow FS to be tailored for cooperative application needs. Section 4 illustrates the applications and our current experience. Section 5 includes the concluding remarks.

[1] The strategy adopted after we reach multiple partitions is beyond the scope of this paper. The overall framework is described in [9].

2 Failure Detection and Group-Based Systems

Failure detectors are one of the key components of group-based systems. Group-based systems can be seen as providing two main services. First, the *membership service* which maintains a consistent *view* of the nodes involved in a distributed computation. The membership protocol forces the system to conform with connectivity information provided by the failure detector. Second, the *multicast service* provides various semantics of multicast communication, such as for example causal and total ordering, for information dissemination within a group [19]. A possible combination of the two services implements what is usually known as Virtual Synchrony (VS) [4].

ISIS is one early example of a system providing VS, which failure detection has been tuned to make few mistakes in LAN environments. Processes suspected to have crashed are removed from the system by the membership protocol. This leads to irrevocable exclusion of a process (i.e. a user in the case of cooperative applications) and thus, should be accurate. In large scale distributed systems, where the unpredictability and the variance of communication are high, it becomes much harder to tune the system in a generic way. A pessimistic approach risks forcing too many live processes out while an optimistic approach has the inconvenience of reporting process crashes with a latency which may compromise safe progress of the application. Properties of FS are known as completeness and accuracy [6]. Completeness captures the fact that crashed processes will be suspected by every or some processes while accuracy states that all or some correct processes are never suspected.

We are developing a group infrastructure for large-scale networks, NAVTECH [20], which constitutes the support framework for the work presented in this paper. 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 and activity-support services such as reliable group communication, flexible failure suspector, and group membership. The NAVTECH Failure Suspector (FS) module supports two kinds of liveness/connectivity tests:

- **Standard tests -** plain *I'm alive* test, whose semantics is, informally: "I have heard about this site, directly or indirectly, in the last x seconds"; x is a baseline system parameter, whose setting may be adaptive;

- **Performability tests -** ad hoc test, superimposed on the standard one, requested for a set of sites, normally a site-group; the objective of the test is to assess whether members of a given site-group are capable of meeting a given performability goal.

Performability, is a combined measure of performance and reliability of systems.

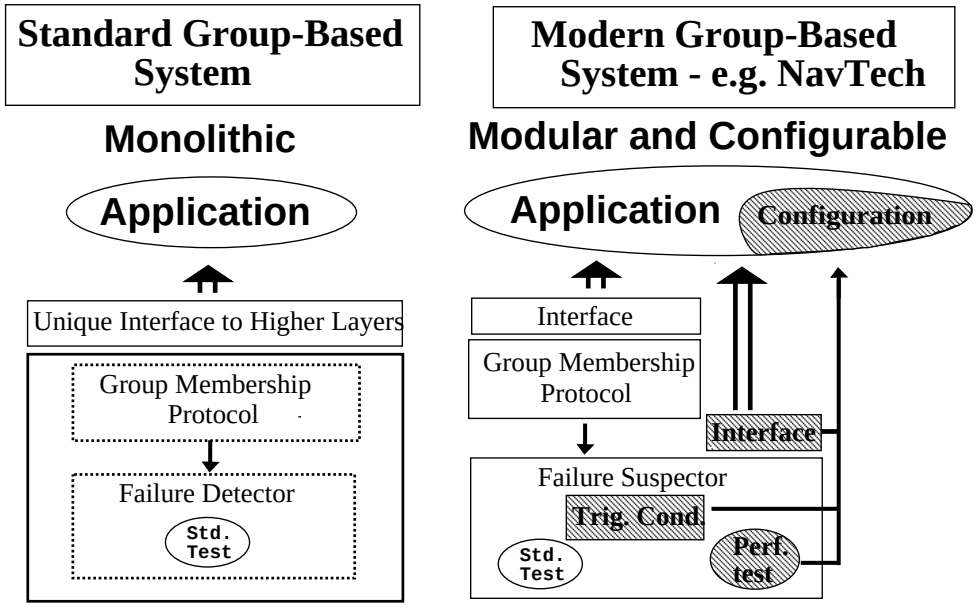

Figure 1: The Evolution of Group-Based Systems

The performability test semantics is defined by external requests, through variable types exported by the FS service, their values being set in a user-dependent fashion. The on-line performability test is, to our belief, an innovative feature. Its main utility is to map different failure semantics to different site-groups. Figure 1 depicts the functional architecture of the NAVTECH FS emphasizing the difference in approach from earlier generations of group-based systems. [2]

We now discuss how we can tailor the FS module in order to model application requirements.

3 Configurable Failure Suspectors

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.

3.1 Specifying Connectivity

Each parameter is characterized by a set of variables, as illustrated in Table 1, namely:

SR (Sample Rate): the maximum interval at which the parameter needs to be evaluated.

TH (Threshold): the parameter maximum acceptable value.

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

[2] Configuration and modularity are also a prime concern in other systems [16, 13].

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

Value (V): the parameter current value.

Operational Parameters Set-Up				**node**$_1$		**node**$_2$		...	**node**$_i$		...	**node**$_n$	
Parameter	**SR**	**TH**	**W**	**TE**	V_i^1	**TE**	V_i^2	...	**TE**	V_i^i	...	**TE**	V_i^n
roundtrip (s)	1m	10	2	1	12	2	25	...	-	-	...	0	0.3
throughput (Mbits/s)	5m	1	1	0	2	1	0.5	...	-	-	...	0	4
...	.	.	0	.	.	.	.	...	-	-	...	.	.
Disturbance Index (DI)	1m	3	-	(1*2)+(0*1)=2		(2*2)+(1*1)=5		...	-		...	(0*2)+(0*1)=0	
				↓		↓			↓			↓	
LCV$_{node_i} \rightarrow$				**1**		**0**		**...**	**-**		**...**	**1**	

Table 1: Variables associated with operational parameters at $node_i$

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, 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 a index, named *Disturbance Index* (DI), which is a sum of products of TE variables with their corresponding weight. That is:

$$\text{DI}_l^r = \sum_j \text{Weight}_l^r(j) * \text{TE}_l^r(j)$$

The DI is computed periodically, as specified by its sample rate. It indicates how often operational 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 satisfactory. 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 LCV$_l[r]$ is set to false.

3.2 Triggering Agreement Conditions

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, can be piggybacked in normal traffic with negligible overhead.

The GCM indicates how each node perceives the connectivity to every other node. If node i has good connectivity to j then GCM$[i, j]$ is true, and false otherwise. The activation of the membership module depends on a condition which is a function of GCM values. This condition is evaluated on each LCV receive and is 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 also known as "social role" in the context of cooperative applications. Weights are also defined and used for partition processing strategies purposes [10]. Another integer, called the *Global Suspicion Quorum* (GSQ), 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:

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

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 [12].

3.3 Runtime Connectivity Feedback

On top of the specification of FS and of the triggering conditions, NAVTECH allows the programmer to read current values of the relevant operational parameters. This is used for cooperative applications to provide feedback to the users, a function usually referred to as collaboration awareness. As described below, NAVTECH allows to implement such module with little extra cost.

4 Application and Experience

4.1 Cooperative Personalities Library

Cooperative applications often use *social roles* or *roles* as a way to model the relative importance of each user. These roles are application level entities used to enforce access control over the shared workspace. We extend this concept by using NAVTECH FS to create a library of *cooperative personalities* as a way to model different users connectivity requirements. The advantages are:

- It simplifies the interface to the FS.
- It allows FS to be tuned for a whole range of applications.

Typical default threshold values, weights and sample rates are defined using known specifications of responsiveness and estimated bandwidth for the target applications [18]. For example, in [5] it is recommended that notification and response times should

be comparable. We consider high responsiveness typically below *2 seconds* while low requirement means accommodating delay values above *10 seconds*. Note that local computing overhead is not taken into account since the response time of the system is mainly due to the communication over the network. Excluding continuous media transmission which lies outside the scope of this work, throughput requirements can be expected to reach *1 Mbit/s* when many workspace entities (files, graphics) are exchanged amongst participants. When a participant is mostly receiving we can consider that half the minimum requirement is sufficient.

In a first phase we have tried to model cooperative personalities based on their responsiveness and talkativeness requirements. We defined four cooperative personalities using roundtrip delay and throughput NAVTECH FS parameters. The classification is presented presented in Table 2.

Characteristics	**Responsiveness** (roundtrip)	**Talkativeness** (throughput)
Extravert	High	High
Demanding	High	Low
Verbose	Low	High
Laconic	Low	Low

Table 2: The Cooperative Personalities Library

Choosing a personality connectivity requirement is made through a selectable programming parameter which greatly simplifies the task of the programmer. For example, in a presentation-like tool, the "speaker" should select personality *extravert* while eager attendees would select personality library *demanding*. However, attendees with remote interest would select personality *laconic* etc. The requirements of each parameter are quantified using High and Low as to reflect their importance in value and weight. The actual values were pre-set using the recommendations presented above. The library can be further extended using other FS parameters.

4.2 Enhanced Feedback

Another application of Failure Suspectors is collaboration awareness enhancement. This is achieved by intercepting the output of the FS and providing connectivity clues to the users using new feedback techniques.

We are experimenting in incorporating feedback in various building blocks such as audience monitoring, telepointer facility and concurrency control modules. Those first experiments of Tailored Failure Suspectors gave satisfactory results. An example is given in Figure 2 where sampled values $V_i^{1..n}$(roundtrip) are filtered and used to represent some "distance" between users. Values are then displayed for audience monitoring purposes. The User Interface (UI) analogy for the audience monitoring mechanism is inspired by a hill, where people go up and down. Participants are aligned by increasing magnitude from left to right as they get more distant from the viewer.

A percentage value is also displayed on the top left corner of the UI. This number represents a global distance measurement using a single value and, therefore, is a function

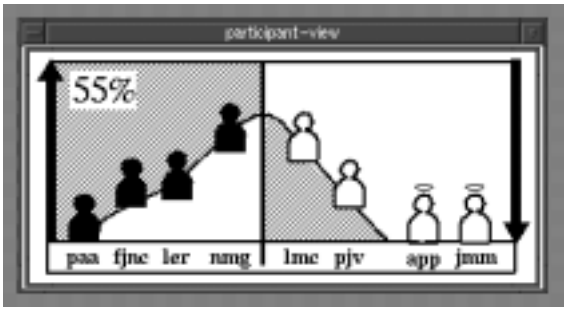

Figure 2: Adaptable Audience Monitoring

of the feedback delays of all participants in the current view. It is implemented using the GCM, the Global Connectivity Matrix, local to each node, described earlier. The unreachable participants are excluded from the measurement since they are automatically excluded from synchronous interactions by the membership protocol.

The support presented in this paper has been partially incorporated in the NGTool application [2]. NGTool is a group decision support system (GDSS) which runs on Unix workstations over the Internet and uses X Windows and the Motif toolkit. Further details of our experiments are given in [7].

5 Concluding Remarks

This paper has shown that tailored failure suspectors could be used to better support cooperative applications. The approach is based on NAVTECH, a modern configurable group-based system. The configuration of the failure suspector and the specification of the agreement triggering conditions were described. A cooperative personalities library was created in order to simplify the configuration process. The library offers various connectivity requirements using pre-set values. Furthermore, initial experiments showed that low level connectivity information can enhance collaboration awareness by combining tailored FS outputs with new feedback techniques. Further use of the NGTool will demonstrate whether erroneous membership decisions can significantly be reduced using the triggering conditions. The proposed approach is at an early stage and further experiments are underway to confirm our initial results.

Acknowledgements

We are grateful to Carlos Almeida for commenting on an early version of this document. The NAVTECH failure suspector is being implemented by Jorge Frazão. NGTool is being developed by Pedro Antunes.

References

[1] 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.

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

[3] 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.

[4] 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.

[5] S.J. Gibbs C.A. Ellis and G.L. Rein. Groupware, Some Issues and Experiences. *CACM*, 34(1):38–58, January 1991.

[6] 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.

[7] 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.

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

[9] 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).

[10] François J.N. Cosquer and Paulo Veríssimo. The Impact of Group Communication Paradigms in Groupware Support. In *Proceedings of the 5th Workshop on Future Trends of Distributed Computing Systems*, August 1995. (To appear).

[11] 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.

[12] 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 9-6º, 1000 Lisboa, Portugal, April 1995.

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

[14] 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.

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

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

[17] 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.

[18] T. Rodden, Mariani J.A., and Blair G. Supporting Cooperative Applications. *Computer Supported Cooperative Work*, 1(1-2):41–67, 1992.

[19] Luís Rodrigues and Paulo Veríssimo. Causal Separators for Large-Scale Multicast Communication. In *Proceedings of the 15th International Conference on Distributed Computing Systems*, June 1995.

[20] Paulo Veríssimo and Luís Rodrigues. The NavTech Large-Scale Distributed Computing Platform. Technical Report RT-95, Broadcast Project, INESC, Rua Alves Redol 9-6º, 1000 Lisboa, Portugal, 1995. (in preparation).