Adaptive Synchronous Cooperation over Large Scale Networks[*]

François J.N. Cosquer
INESC[†]

Pedro Antunes
INESC-IST[‡]

Nuno Guimarães
INESC-IST

Paulo Veríssimo
INESC-IST

This paper presents an approach to effective synchronous work support in the context of large scale interconnected networks such as the Internet. We show how the current technology limits influence synchronous cooperation. Distance-Adaptation is proposed to tackle problems linked with high variance in communication delays and partial unreachability between cooperating users. Low level support information and new feedback techniques are combined to improve group awareness in synchronous large scale CSCW tools and applications. We illustrate our claim using three cases of synchronous work mechanisms. The objective is to contribute to the development of more robust and adaptative cooperative applications.

Introduction

The rapid growth and availability of interconnected networked infrastructures such as the Internet [3] have raised dramatically users expectations. Amongst the most promising applications are those oriented towards multi-user interaction and collaboration. Papers on synchronous work have shown the importance and necessity

[*]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^o - 1000 Lisboa - Portugal, Tel.+351-1-310 0000, Fax: +351-1-52 58 43, e-mail: {fjnc,paa,nmg,pjv}@inesc.pt

[‡]Instituto Superior Técnico - Technical University of Lisbon

of interactive applications such as talks, shared editing, presentation tools and teleconferencing [3,3]. Furthermore, applications classified as asynchronous often need seamless transition from asynchronous to synchronous phases [3] in order to achieve progress.

Supporting synchronous work or synchronous interaction over today's computer networks is a difficult task. Distributed synchronized applications are known to have the most stringent requirements [3], the main reason being that they rely heavily on the communication subsystem. This raises complex problems since in large scale interconnected networks, communication suffers from high variance and unpredictable transmission delays. Furthermore, such environments are also more likely to exhibit partial unreachability between users, known as partition failures [3]. This means that current mechanisms and implementation techniques are partially ill-suited to provide satisfactory functionality to the users in today's large scale networked environments.

Having surveyed a number of synchronous groupware applications and platforms [3], we found that little attention has been paid to scaling and end-to-end reliability issues. This is clearly not acceptable if synchronized cooperation applications are to become widely used over large scale networks.

Providing solutions to the design and implementation of large scale distributed computing systems (LSDCS) is the main objective of the European partners involved in the Broadcast project [3]. The work presented here represents the results of the combined efforts of two research groups at INESC involved in this project. One group has focused on the systems aspects of cooperation while the other emphasized the top-down approach to organizational computing. The common goal is to research and develop new CSCW techniques taking into account both the underlying technological restrictions and awareness of organizational and group work.

We have identified at least three possible types of approach to the problematic of interaction in LSDCS. The first approach advocates the use of asynchronous interaction between users [3] and avoids the inherent inefficiency of communication and distributed computations in large scale networks. A comparable approach by [3] showed that progress could be achieved using semi-synchronous interaction during collaborative editing sessions. However, there is always a need for intensive interaction such as, for example, the start or the end of a cooperative session. For this case, other facilities have to be provided.

The second type of approach is more dedicated to system support for continuous media. Correct handling and use of continuous data requires a complete end-to-end approach as both communication (transmission between sites) and scheduling (at the local site) guarantees are needed [3,3]. This dynamic adjustment of quality of service (QoS) enables some of the requirements of distributed multimedia applications to be met. However, it requires having (partial) control over network protocols and operating system scheduling policies, which we might not have in an Internet-like environment. Furthermore, it does not provide solutions for cases of unreachability betweens nodes.

The third relevant approach, which apparently has no direct link with cooperation and LSDCS, is to be found in presentation degradation techniques. These techniques rely on time-elastic objects [3] which can self-adjust the quality of their presentation according to the amount of system resources available. This best-effort approach does not depend on the underlying implementation platform. Furthermore, the model allows the application to specify the type of degradation to apply.

In order to tackle problems associated with synchronous cooperation in LSDCS, we have combined concepts of end-to-end requirements satisfaction with presentation degradation concerns. We have assumed minimal control over the operating system and the transport layer, as it is the case with environment offered by the Internet infrastructure. We have also tried to follow the system design argument as exposed in [3] in other words we have tried to avoid duplicating support at different layers of the system.

We have extended the concept of automatic smooth presentation to what we call the “distance-adaptation” paradigm. The underlying support allows end-to-end connectivity information to be tailored to the application’s needs. Collaborating users can perceive the “distance” between them. Combined with new feedback techniques, this changes group collaboration awareness dramatically . Furthermore, when lower level layers cannot decide on system state, feedback is immediately provided to users according to pre-defined policies (application dependent). This is an innovative aspect: because users are kept informed and no longer suffer ad-hoc system decisions, they can decide on whether to adapt to the current state, or proceed indifferently. This offers a high level of flexibility and hence satisfies one of the most important recommendations for the system design of cooperative tools.

The “distance-adaptation” paradigm is described in Section 2. We also briefly explains how a group-based communication subsystem can be extended to provide necessary support. We illustrate how this support can be used together with new adaptative feedback techniques in Section 3. It is followed in Section 4 by a discussion on the results and possible future work. Conclusions are given in Section 5.

The “Distance-Adaptation” paradigm

We explain the motivations behind the “distance-adaptation” paradigm followed by an overview of the services that are provided. The organization of the underlying system support is also briefly described.

Design Issues

The “distance-adaptation” paradigm integrates adaptation at all levels of the system. This means, we should be able to specify and detect the underlying system state. In the context of synchronous applications we are mainly concerned with the level of connectivity between users. Each participant can think of being separated from the others by a certain “distance”. Furthermore, we need to present this information

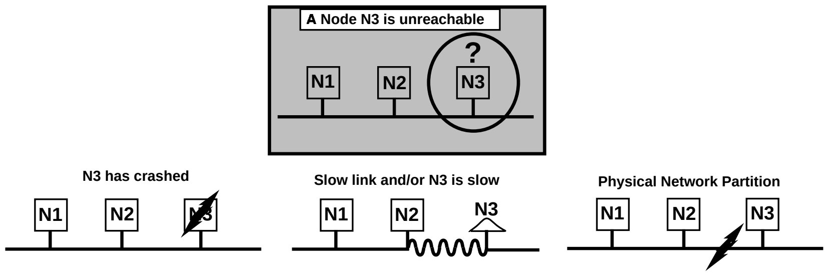

Figure 1: Unreachability: The Impossibility Result

dynamically to the cooperating users in a meaningful way. This should allow them to make decision, i.e. to adapt, based on this "distance" information thus the "distance-adaptation" paradigm.

Unfortunately, in asynchronous message-passing computing environments, it is impossible to differentiate a slow node or link, from a crashed node or broken link [3]. The situation is depicted in Figure 1. This problem of not always knowing the system's state has been a major concern of the "distance-adaptation" paradigm. This constitutes one its innovative aspect and has consequences on the degradation model. The "distance" information is used whenever available to reflect the current system state. However, dedicated service is provided during periods of unreachability between some nodes of the system.

These requirements led us to adopt a two-level approach. We first detect degradation and pass the information to the upper layers. This is what we call the *Level of Service (LoS)*[1]. If after some pre-defined time the situation is still unknown, we adopt a strategy on how to proceed with the synchronous cooperation. This is the *partition mode* what is usually known as partition processing strategy. It is programmed as part of the application. This includes defining policies for reconnection once the underlying support is able to give distance measurements again. Furthermore, for extreme cases, i.e when no improvement is detected within an application meaningful delay, we offer users the possibility to force decision to the system. A summary of each service and its parameters is given below.

(1) **Level of Service (LoS):** the following parameters are programmed by the applications designer.

- Distance measuring function: used to define the current connectivity state i.e the "distance".
- Unreliable threshold: this defines the output below which the distance is considered too big for maintaining an interactive session.
- Unreachable threshold: this defines the output below which the distance is such that it should correspond either to a partition or a crash failure.

[1]We avoid here overloading the term Quality of Service (QoS) usually used to describe adaptability to the network and computing resources through reservation and scheduling mechanisms.

(2) **Partition mode:** This provides functionality as to maintain service during unreachability periods and smooth reconnection on repair. The following parameters are also application dependent.

- Splitting and merging rules: once some node are declared unreachable (or reachable again), we need agreement on the (un)reachability status resulting in non-intersecting partitions (or rejoining).
- Partition identification: this allows to specify how to assign partition types (e.g. *main* or *secondary*). Runtime partition type information can be used by the application to discriminate the type of synchronous operations allowed.
- User-level crash decision: this feature is useful when the application has been running in partition mode for a long period due to unresolved unreachability cases. By forcing a crash to be detected by the system the application can then run again with a restricted number of participants but without the restrictions of the partition mode.

Before illustrating the benefits of the "distance-paradigm" in the context of synchronous cooperation, we now briefly describe the implementation platform.

Architectural Decisions

Research in distributed systems has led to the development of technologies which also have their role to play in groupware system support: replication management algorithms, group communication protocols and group management tools [3]. Group tools have shown their usefulness in many projects and are considered to yield simpler and more efficient implementations of a large class of distributed applications [3].

By providing facilities for the management of replication and cooperation activities, groups have the potential to ease the structuring of a number of future distributed systems. Early experiments with groupware applications have shown that a key issue is *group activities management*. This suggests a beneficial relationship between what may be provided by a group-oriented system and the groupware applications requirements. This has been confirmed by projects like COLA[3] and Duplex [3] which have based their implementation on a group-oriented platform.

Group-oriented systems can be viewed as providing two key services. First, the *membership service* which maintain a consistent view of the nodes involved in the distributed cooperation. It is based on a failure suspector component which provides information of basic connectivity situation. Second, the *multicast service* provides various semantics of multicast communication, such as for example causal and total ordering, for information dissemination within a group.

Building support for the "distance-adaptation" paradigm services requires modifying our group-oriented platform. We need to integrate requirements specific to the application domain into the system. The various modifications are listed below:

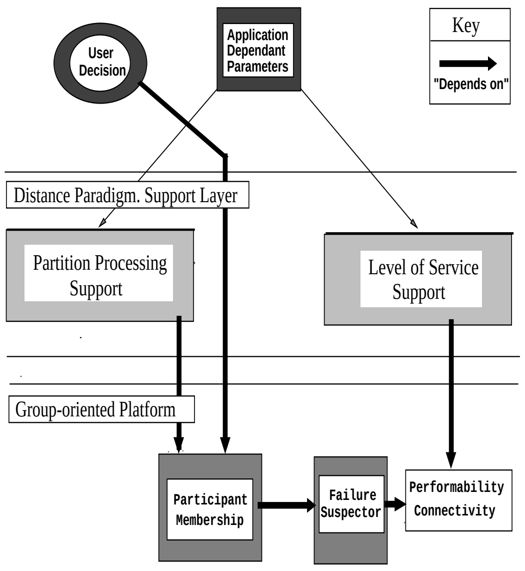
User Decision
Application Dependant Parameters
Key
"Depends on"
Distance Paradigm. Support Layer
Partition Processing Support
Level of Service Support
Group-oriented Platform
Participant Membership
Failure Suspector
Performability Connectivity

Figure 2: Group-based "distance-adaptation" support

- Extend Failure Suspector functionality; by adding application-defined performability tests to the mechanisms normally used to suspect failure. This maintains connectivity metrics to be used by higher layers like the Level of Service. Furthermore, the decision on unreachability is no longer a system parameter but the unreachable threshold parameter.
- Extend Membership Services: The group-based communication subsystem provides a *strong-partial membership service* [3]. Informally, this means that at any time it is guaranteed that concurrent views are non-intersecting. This property is useful for developing our partition functionality of splitting and merging rules. A new feature is the user-level crash decision facility. This can be implemented using an upcall mechanism from the membership component.
- Customize Membership View; by adding extra partition status information to the group membership view. This allows partitions identification.

A simplified view of the group-based support is given in Figure 2. It shows the two main services offered by the "distance-adaptation" paradigm and their integration with lower level layers. A more detailed description can be found in [3].

The Feedback during Synchronous Work Sessions

An important part of the validation path of the "distance-adaptation" paradigm was to test its benefits from the end-users point of view. The major concern is to provide sufficient awareness so that users can avoid disruptive effects on work sessions. The answer is based on several already tested solutions that rely on the feedback delivered by the infrastructure, as a counterpart to the lack of feedback delivered by end-users

[3,3]. To illustrate our claim we selected three types of mechanisms frequently used during synchronous work sessions: group monitoring, gesture support (telepointer) and a generic concurrency control mechanism.

Group Monitoring and "Distance-Adaptation"

Group monitoring refers to the component used in most synchronous applications to monitor the audience attendance. It corresponds to what is referred to as shared feedback in [3]. Usually, it uses small pictures or even video frames of participants. Our objective is to integrate the group monitoring information with the "distance-adaptation" paradigm. Our experiences and knowledge of existing systems led us to the following proposal, illustrated in Figure 3.

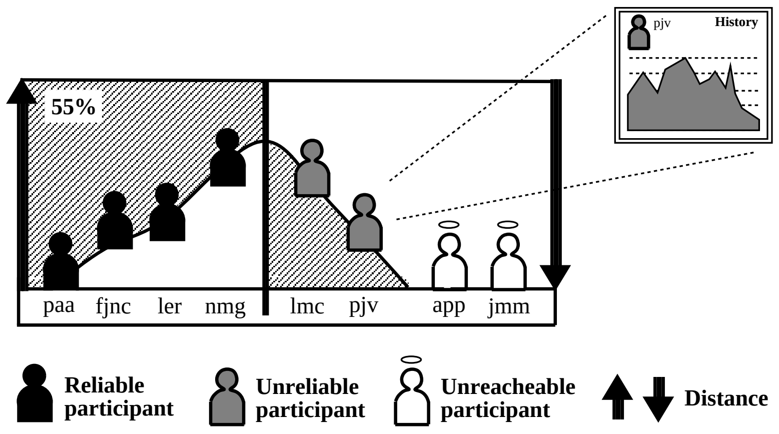

Figure 3: Group Monitoring

All participants are identified by small pictures and user names. Participants are divided in two groups: reliable and unreliable ones. The reliable participants are the ones able to provide feedback with delays below the pre-defined unreliability threshold. Unreliable participants are the ones unable to meet this threshold. A second threshold of delayed feedback is established in order to identify participants that are presumably unreachable.

The two thresholds clearly establish different effectiveness degrees for group synchronous interactions. A user will find difficulties in interacting synchronously with unreliable participants, but an estimate of the delayed feedback avoids the immediate rejection of synchronous work. Furthermore, when effectiveness is paramount, the user can request unreliable participants to leave the group. As for the unreachable participants, the long feedback delays do not allow for any practical synchronous interactions and therefore the application should smoothly switch to partition mode without user intervention.

The UI analogy for the group monitoring mechanism is inspired by an 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. The distance is measured vertically, i.e. lower participants have small feedback delays and higher participants have large feedback delays. Furthermore, 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. The unreachable participants are excluded from the measurement since they are automatically excluded from synchronous interactions.

The overall function is specified by the application developer as part of the LoS mechanism. Several strategies can then be adopted:

- *Show the worst case feedback delay* - This allows the user to perceive the worst case scenario. On the other hand, frequent changes of this value generate an overemphasized impression of feedback delays.
- *Show the average value* - Does not reflect the conditions of wide area communications when several participants are local.
- *Show the average value of the participants with longer feedback delays* - This is a compromise between the two cases above: it reflects the worst case but avoids frequent changes.

Finally, a window with the historical data of the distance for a participant can be displayed by clicking on the icon representing her/him.

We believe that the level of information provided by the mechanism described above is sufficient to enable end-users to confront the communications performance with their requirements over working sessions. In situations like network partition, are promptly detected and participants who are available for proceeding with synchronous operations are identified. The selection by the application designer of the two feedback delay thresholds described above is of critical importance. Values that are too low lead to a ripple phenomenon, which becomes excessively intrusive, while too high values result in a less informative mechanism. This issue has to be further studied.

Gesture Support Using Telepointers and “Distance-Adaptation”

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. Telepointer facilities are usually provided by many synchronous CSCW tools. In the example below we assume that the shared workspace only has one telepointer for all the participants. The issue of how one user grabs the telepointer (access control) will not be considered. We rather concentrate on the manipulation issues which relate to the “distance-adaptation” criteria.

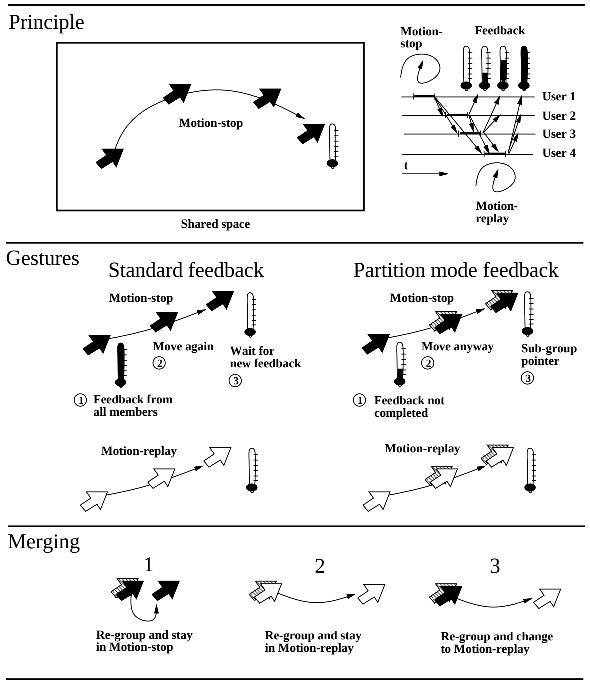

Figure 4: The Telepointing mechanism

The principles adopted for the telepointer are illustrated on the top part of Figure 4. To avoid flooding the network with too many messages, the telepointer uses a "Motion-stop" mechanism: the telepointer movement is recorded until the user stops; the recorded movement is then sent to the group. The Motion-stop is distributed to other participants using the reliable multicast primitives provided by the underlying group support.

Each participant's site, when receiving a Motion-stop will execute a "Motion-replay", i.e. it will replay the movement, and broadcast the information that the Motion-replay has been executed. A meter is displayed near the participant's telepointer after a Motion-stop or Motion-replay. 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.

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 4). 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 4). In this case, the system will automatically enter the partition mode and generate a sub-group excluding the unreliable participants. 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 enter partition mode.

The functionality of the telepointer is maintained in partition mode but the participants are notified that not all original group members are participating. Further information on sub-group membership is available through the group monitoring mechanism described above. When in partition mode the telepointer manipulation policies can use the partition identification information. This allows to decide what happens in each type of partition. For example, one can defined rules such that there is a unique *main* partition and possibility for many *secondaries*. The default policy based on this information can allow each side to possess its own telepointer so as to preserve animation during partition mode.

When the participants who were excluded from the *main* sub-group become reliable again, the system will terminate the partition mode and merge. We sketch below few cases where the owner of the telepointer in the main regain control on the others on merging (see Figure 4 bottom).

(1) *The telepointer is owned by a user in the main partition and the system decides that it will continue to own it* - The telepointer returns to its graphical aspect and stays in location.

(2) *The telepointer is not owned currently by the user* - The telepointer changes its graphical aspect. This concerns participants of both partition types. If a telepointer had been moved in a secondary partition it moves (animation) to the new owner's location.

(3) *The telepointer was owned by a user in a secondary partition and the system will give to the owner in the main* - The telepointer changes its aspect and moves (animation) to the other participant's telepointer location.

The functionality of the telepointer requires to be complemented with the group monitoring mechanism. This is mostly due to the need to analyze the monitoring information in the cases where the meter does not reach the top and a decision has to be made about how to carry on synchronous task. Providing partition identification type users can proceed, smoothly, in partition mode. One arguable design decision was whether all participants should see the meter or not. Our decision was to show the meter to all participants, such that everyone be aware of the delays in telepointer gestures.

Concurrency control and "Distance-Adaptation"

Concurrency control concerns to the problem of having several users trying to manipulate a shared resource concurrently. We will not discuss the different policies [3] but rather focus on expliciting and expressing generic concurrency control mechanisms and integrate them with the appropriate awareness of "distance-adaptation".

The first design issue to consider is that the displaying of concurrency control actions is kept separated from the graphical representation of the resource. Although this decision may result in more complex UIs, when many resources have to be displayed in a shared workspace, this decision allows for a generic solution independent from the possible resource representations.

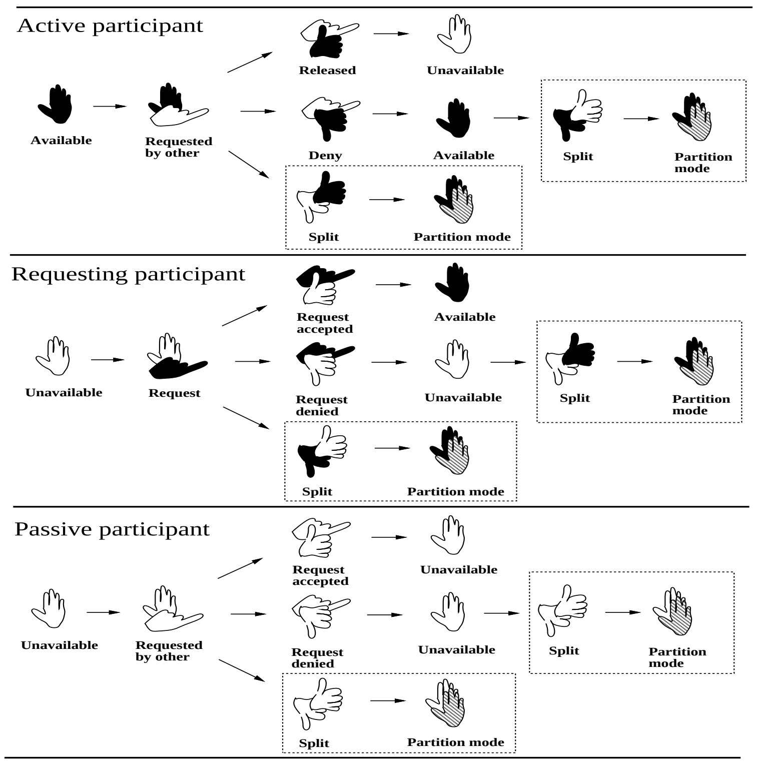

Figure 5: Concurrency control

The basic concept is to use hands language code to represent control: the open hand represents ownership; the pointing hand represents a request; the thumb up or down represents acceptance and denial. Hands do also have colors: black means "me", white means "other" and grey is a special color that means that the resource is locked by more than one participant i.e. the resource is in partition mode. Note that the convention is similar to the telepointer's case described above.

Three types of participants are then discriminated: active (locking the resource), requesting (to lock the resource) and passive (has interest in the resource but does not lock or request it). From the above design decisions, it results that a white pointing hand means that *someone* is requesting the resource, a black pointing hand means that *I* am requesting the resource, and so on. These simple representations are combined and sequenced in order to provide very expressive feedback on the concurrency control actions that are performed over a particular resource. See Figure 5 for and illustration of the the different cases as seen by each type of participant.

The "Distance-Adaptation" case to consider here is the unreachability of one or several users, detected by the system when a participant crosses the discriminated threshold of feedback delay. In this case the system does an automatic split operation into partition mode and eliminates the unreachable participants from the created sub-group. See Figure 6 for an illustration of this procedure. Any user, active, requesting or passive, or the system itself, can decide to terminate partition mode and *merge*, i.e. request a single locker for the resource (we will not discuss here the problem

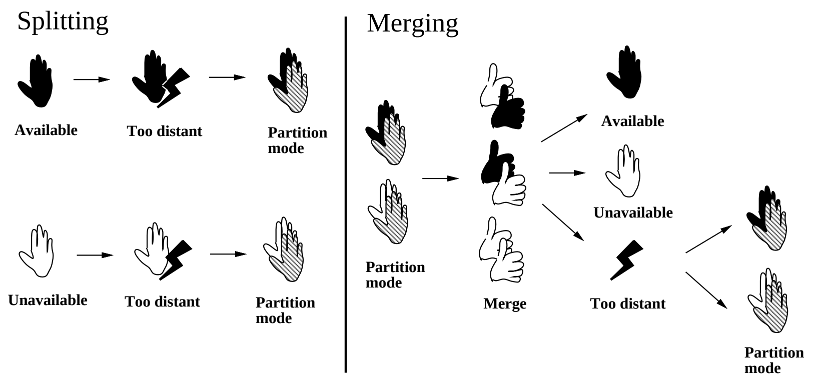

Figure 6: Concurrency control - The Partition mode

of consistency). The Figure 6 also describes the possible situations that may arise when merging. When a participant requests to merge but there are unreachable participants, the operation is aborted.

A common concurrency control policy used in asynchronous work consists in relaxing resource locking and allowing multiple modifiable copies of the resource. Such characteristic can be incorporated in the proposed concurrency control mechanism usinf the system support for the partition mode. This is illustrated in the dotted examples of Figure 5. An active user can explicitly request to *split* into the partition mode after receiving a request for locking. The requesting participant can also split after receiving a denial for locking.

The description of the concurrency control mechanism has been overly simplified. For example, the pictures do not describe the cases where unreachability follows requests for locking. Also, several policies, which may be arguable, have been kept out of the discussion. Another example, for simultaneous requests the system will operate first come first serve. This case is not shown in the illustrations but the user that does not get locking sees a *denial* followed by a *request by other* and *accepted*.

Discussion

Our first comments address the UI design of the techniques previously presented. The overall objective is to provide good feedback to users over the actions that are being executed during work sessions whether originated by the users themselves or by the system. The pertinent question is if there is sufficient end-user acceptance of the design options that were taken. In respect to this issue, we have inquired a small set of users and found several issues which have not yet been handled satisfactorily:

- *Group monitoring:* the representations of the participants in work sessions could be more expressive and, preferably, should display a live picture of the persons. Furthermore, users expected that the UI specified for group monitoring would also be used for managing the group composition. For instance, to remove users from work sessions. The current group monitoring mechanism was designed just to only provide feedback on group distance. Our idea is that these services should be detached, since they have different interaction and awareness requirements which must be fine tuned, but one must then also think of the integration of the various solutions.

- *Telepointer:* an immediate question raised by users after analyzing the telepointer specification, concerns what happens when a user does not wait enough time after a Motion-stop such that feedback from a significant number of participants has not received. As defined, the system will maintain the original group as long as member do not exceed the unreachability threshold. Should the user reiterate the operations repeatedly he will take the risk of creating his own partition. Another question concerns the telepointer design itself, and more specifically the associated meter: why is the feedback not displaying user names or at least discriminate reliable from unreliable participants. This issue will be corrected in further versions.

- *Concurrency control:* There is at least one ambiguity on the UI of the concurrency control mechanism. The colors of the split action do not conform well with the definition that was proposed, i.e. black for "me" and white for "other". This can become a problem if the information presented to the user becomes intrusive or even confusing.

We are trying to integrate the mechanisms that were presented in a unify way. An example of the required integration can be demonstrated by the usage of telepointers. Given that there is only one telepointer available in the workspace, it is a shared resource for which users must compete. A tailored version of the concurrency control mechanism could be associated to the locking of the telepointer. This integration of different mechanisms provides a consistent approach to the design of complex interaction processes and allows to seamlessly extend the notion of "distance-adaptation" throughout the whole working environment.

Regarding the underlying support needed for building the current version of the "distance-adaptation" paradigm, we can make the following three remarks:

- *Difficulty of measurement:* one issue of this work is how to define the *distance*, both in terms of its measurement, and in terms of its expression to the end-users. System support for defining *distance* in measurable terms is based on the failure suspector component of the underlying group-based system. It is hard from dynamic connectivity information to extract a stable measure to present too the end-users. The choice of the thresholds is application dependent but can also be topology dependent which complicates the programming task.

- *Assumptions and coverage:* the aim of our partition mode is more dedicated towards support for short periods of unreachability rather than disconnected operations. Processor or link crashes rarely hold for long. For example, recent statistics on the T3 backbone show a tendency of 15 minutes or below for outage duration. However, only experience will tell us if this assumption holds.

- *Finding the right model:* this initial work confirms our intuition about modern group-oriented systems in providing suitable functionality for this type of synchronous applications over large. Early experiments with Isis shown the need for a more flexible implementation platform [3]. Horus, the follow-up project [3], has been selected as one possible target for further development because its high configurability allows easier modifications.

Conclusions

The objective of the work reported in this paper is to design mechanisms that provide increased adaptiveness of synchronous and large scale CSCW applications. This adaptiveness is the result of combining system information with rich interactive feedback techniques. The described approach can be related to solutions proposed in other areas of computing, namely media degradation approaches for multimedia applications, and thus be considered as a useful contribution to solve the problem of adaptiveness and degradation in computing environments.

One innovative aspect is the concern for unreachability which has been integrated through all levels of the system. As opposed to system-level failure detection, we proposed degradation composed of a new intermediate state (*unreliable*) between reliable and unreachable. The degree of unreliability of users is calculated and forwarded to upper layers of the system. Furthermore, history information can always be used for diagnostic purposes. When participants are unreachable, we preserve potential for maximum animation by allowing multiple partitions to operate concurrently. Seamless transitions from full connection to partition operation mode and vice-versa are achieved by defining appropriate splitting and merging rules.

A great deal of work remains to be done in order to generalize the approach proposed by the “Distance-Adaptation” paradigm. We have concentrated our efforts on preserving animation without dealing with the potential conflicting concurrent updates when operating in partition mode. This is the concern of on-going work and we are studying how we could integrate our work with existing approaches like active diffs [3].

The user interface techniques that were presented are meant to be general and widely applicable to different cooperative applications. Furthermore, as mentioned earlier, the expressive power of these interface mechanisms can be further enhanced with multimedia facilities, from more sophisticated graphics to virtual reality features. In any case, the point we tried to make is that the *distance* notion can be usefully expressed and transmitted to the users of CSCW applications. The contextual clues on underlying system state allow them to adjust their expectations and define appropriate strategies throughout cooperative sessions.

Acknowledgements

We are grateful to Alexandre Lefebvre and Susan Tinniswood for commenting on an early version of this document. Many thanks to David Matos for his valuable help with LaTeX.

[1] Colin Allison and Mike Livesey. Coping with Concurrency in Real Time Groupware. In *Proceedings of the Usenix SEDMS IV*, pages 289–295, September 1993.

[2] P. Antunes and N. Guimaraes. Multiuser interface design in cscw systems. Technical Report Volume 3 - Systems Engineering, Chapter: 4 - Cooperative Working, ESPRIT Basic Research Project 6360, Broadcast, 1994.

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

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

[5] 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^o, 1000 Lisboa, Portugal, September 1994. (Also available as BROADCAST Technical Report [2nd year - Vol 1]).

[6] 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 (to appear)*, April 1995. (Also available as INESC Report AR 2/95).

[7] Geoff Coulson, Gordon S. Blair, and Philippe Robin. Micro-kernel Support for Continuous Media in Distributed Systems. Technical Report MPG-93-04, Lancaster University, Distributed Multimedia Research Group, Dpt of Computing, Lancaster University, Lancaster, LA1 4YR, U.K., 1993.

[8] T. Crowley, P. Milazzo, E. Baker, H. Forsdick, and R. Tomlinson. Mmconf: An infrastructure for building shared multimedia applications. In *CSCW 90*, pages 329–342, October 1990.

[9] Paul Dourish and Victoria Bellotti. Awareness and Coordination in Shared Workspaces. In *CSCW 92 Proceedings*, pages 107–114, November 1992.

[10] 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.

[11] S.E. Goodman, L.I. Press, S.R. Ruth, and A. M. Rutkowski. The Global Diffusion of the Internet: Paterns and Problems. *Communications of the ACM*, 37(8):27–31, August 1994.

[12] D.P Reed J.H Saltzer and D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):30–41, November 1984.

[13] Michael B. Jones. Adaptive Real-Time Resource Management Supporting Composition of Independently Authored Time-Critical Services. In *Proceedings of the Fourth Workshop of Workstation Operating Systems*, pages 135–139, October 1993.

[14] M. Frank Kaashoek, Andrew S. Tanenbaum, and Kees Verstoep. An Experimental Comparison of Remote Procedure Call and Group Communication. In *5th ACM SIGOPS Workshop on Models for Paradigms for Distributed Systems Structuring*, September 1992.

[15] Sten Minor and Boris Magnusson. A Model for Semi-(a)Synchronous Collaborative Editing. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work - ECSCW'93*, pages 219–231, September 1993.

[16] Broadcast partners. Broadcast - technical annex. Technical Report Project BRA 6360, Esprit, July 1992.

[17] F. Penz, P. Antunes, and M. Fonseca. Feedback in computer supported cooperation systems: Example of the user interface design for a talk-like tool. In *12th Schaerding International Workshop, The Design of Computer Supported Cooperative Work and Groupware Systems*, Schaerding, Austria, jun 1993. Elsevier Science.

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

[19] 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.

[20] Mark Roseman and Saul Greenberg. Groupkit: A Groupware Toolkit for Building Real-Time Conferencing Applications. In *Proceedings of the Conference on Computer-Supported Cooperative Work - CSCW'92*, pages 43–58, November 1992.

[21] 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.

[22] K. Schmidt and T. Rodden. Putting it all together: Requirements for a CSCW platform. In *12th Schaerding International Workshop, The Design of Computer Supported Cooperative Work and Groupware Systems*, Schaerding, Austria, jun 1993.

[23] 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.

[24] Steven H. Tang and Mark A. Linton. Pacers: Time-Elastic Objects. In *Proceedings of the Sixth Annual Symposium on User Interface Software and Technology - UIST'93*, pages 35–43, November 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.