SERVICES TO SUPPORT CONSISTENCY IN MOBILE COLLABORATIVE APPLICATIONS

Keith Cheverst, Nigel Davies, Adrian Friday and Gordon S. Blair

Distributed Multimedia Research Group,
Lancaster University,
Lancaster,
U.K.
kc, nigel, adrian, gordon@comp.lancs.ac.uk

Abstract

This paper describes the design of services to support consistency in collaborative mobile applications. The requirements for application level consistency in groupware applications are discussed and it is argued that existing consistency services for mobile environments fail to address these requirements because they assume infrequent (write) sharing of information. We present the design of a group execution service called G-QEX which is designed to support consistency in mobile applications and includes features necessary to cope with the fluctuations in QoS which characterise mobile environments.

1: Introduction

Collaborative groupware applications require support for maintaining application level consistency between group members. To achieve such consistency requires the implementation of appropriate concurrency control and synchronisation methods [1].

The difficulties of providing effective support for consistency are exacerbated when operating in a mobile environment owing to the potential for rapid fluctuations in the quality of service (QoS) of the underlying network. In particular, it is possible for group members to become completely disconnected from the rest of the collaborating group. This makes the selection of an appropriate consistency policy non-trivial. For example, if a policy was selected which guaranteed consistent views between all group members and one or more group members subsequently became disconnected, no group member could receive any group updates. If, however, a policy was specified such that certain members of a group need not be guaranteed consistent views then if those members became disconnected, group updates could still occur.

This paper describes the design of a set of services to enable application programmers to specify consistency guarantees for collaborative groupware applications. A key aspect of the services' design is their ability to enable the application programmer to specify temporal consistency parameters. For example, the application programmer can stipulate that all group members should receive group updates within one second of the update being sent. If at any point during the lifetime of the collaboration this synchronisation guarantee can not be achieved the services are designed to provide appropriate feedback to the application. This feedback would allow the programmer to take an appropriate course of action. The services are supported by a protocol called QEX [2] (Quality of service driven remote EXecution protocol) developed at Lancaster which provides the necessary feedback on the state of the underlying network to enable consistency guarantees with temporal parameters to be policed.

Section 2 of this paper summarises the requirements for consistency support within groupware applications. The section discusses a number of the CSCW (Computer Supported Cooperative Work) issues relating to groupware applications and focuses on the extensive requirements capture performed during the MOST (Mobile Open Systems Technologies for the Utilities Industries) project [3]. Section 3 reviews the related work on supporting consistency in groupware applications for mobile and fixed networking environments. It is argued that the currently available solutions which address the problems of consistency within mobile environments focus on the support of non-collaborative applications in which shared data is rare. There are, however, numerous applications which manage consistency for groupware applications designed to operate in fixed networking environments and these are described in some detail. Section 4 presents our design for an ISO RM-ODP [4] (Reference Model for Open Distributed Processing) based service to support consistency in distributed mobile groupware applications. The design includes details of both the application programmer's interface (API) and the engineering support required to

0-8186-7499-7/96 $05.00 © 1996 IEEE

enable the service to support consistency in highly heterogeneous networking environments. Section 5 contains examples of how the service could be used to support consistency in a real application scenario. Finally, section 6 presents some concluding remarks.

2: Requirements Analysis

To date relatively little work has been carried out on determining the requirements for collaborative applications operating in a mobile environment. A notable exception was the requirements capture performed during work on the MOST project which focused on providing mobile computing support for field engineers in the electricity distribution industry. MOST developed a trial mobile collaborative multimedia application which took the form of a toolkit to support field engineers. The application allowed engineers to exploit the functionality of a GIS (Geographical Information System) in a conference setting, i.e. to show, manipulate and highlight maps and diagrams to all or a subset of the conference participants via a shared *public view*. Engineers were given the functionality to easily switch between synchronous and asynchronous styles of working through the inclusion of an electronic mail based application designed for issuing job instructions which enabled annotated diagrams to be sent in addition to plain text. The application was supported by APM's ANSAware software suite [5] which was extended to support operation in a mobile environment [6].

Central to the design of the MOST collaborative application was the end-user requirement that field engineers would usually be contactable only via an analogue PMR (Private Mobile Radio) channel. Such a channel offers low bandwidth (approximately 2.4 kbits/sec), high latency and, most challenging of all, long periods of complete disconnection. As a result of this requirement the application's interface was designed to provide appropriate feedback to collaborating group members regarding the state of connectivity within the group. For example, if a group member became disconnected then that member's icon would be displayed with a red background. This feedback provides group members with sufficient information to enable them to adapt their style of working to changes in overall group connectivity. For example, if an engineer realised that a certain cable fitter was currently disconnected from the group then he might choose to delay performing a shared hi-lighting operation.

The MOST application was demonstrated to several utilities companies and a number of shortcomings in its support for data consistency were identified:

(i) *Lack of automatic system support for bringing latecomers to a collaboration up to state.* The current version of the GIS requires latecomers to enter into a dialogue with another group member and explicitly request them to transfer the state of their public view. There should be system support for allowing a latecomer to automatically receive an up-to-date public view. There should also be a facility to allow the latecomer to have the option to observe graphically the creation of the public view.

(ii) *Lack of system support for bringing group members who have experienced an extended period of disconnection up to state.* This set of circumstances can be treated similarly to the above.

(iii) *Lack of system support for checking the consistency of public views between group members.* The system currently does not provide support guaranteeing consistency of public views. Instead, the application relies on voice communications between collaborating engineers to identify inconsistencies.

(iv) *Lack of system support for guaranteeing the synchronisation of group updates.* There is a need for system support to allow guarantees to be made regarding the maximum delay that any group member should have to wait before receiving a group update. This is of particular importance in the MOST application since engineers are collaborating not only via the computer based application but over a separate, real-time, physical network, i.e. the power distribution network. There are a number of scenarios (for example the switching of power supplies) were it is critical that group updates should be propagated in a timely fashion in order to ensure that the application accurately reflects the state of the physical network.

During our work on MOST we studied the way in which group members interact when operating over an unreliable mobile network. In particular, we examined the form of interaction between group members on the basis of the collaborating group's co-location and temporal synchronicity. We observed an interesting anomaly in the standard time/space matrix [7] produced by the CSCW community. The standard time/space matrix is shown in figure 1(a) and reveals clear divisions between the form of interaction and position in the time/space matrix. Using this standard matrix one would expect geographically distributed group members operating at the same time to operate in a synchronous way e.g. by using a real-time conferencing system. However, we discovered that the form of interaction would often vary between synchronous and asynchronous as the available network QoS changed. This led us to produce a modified time/space matrix that does not assume a completely reliable network; this matrix is shown in figure 1(b).

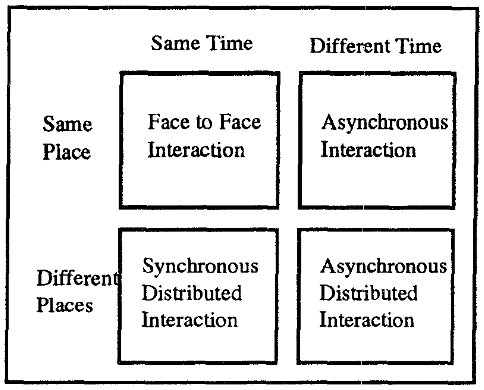

Figure 1(a) : Standard space/time matrix

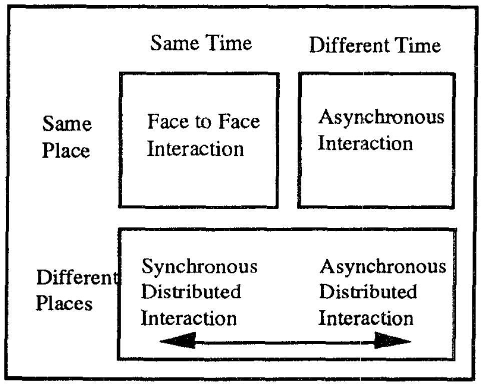

Figure 1(b) : Modified space/time matrix

3: Current Services to Support Consistency

The database community has for a long time considered the tension between providing data availability whilst maintaining the consistency of data. In general pessimistic locking is used in situations where the consistency of data takes priority and optimistic rollback strategies are used where data availability takes priority. One approach to increasing the availability of data when using pessimistic locking is the use of flexible locking techniques. An example of a flexible locking technique is the tickle lock [8] as used in CES to enable effective collaborative editing. Tickle locks automatically release an author's lock to a requesting author after a period of inactivity and so help prevent the improper or accidental hoarding of locks. A second example of flexible locks are soft locks [9]. Soft locks are released in the event of a conflicting hard lock request and each lock request is stored in a log. This log can be interrogated by users to ascertain who has accessed the data item and so enable negotiation for the release of locks.

However, in situations where data availability must be maintained despite network partition then an optimistic approach based on the replication of data [10] needs to be used. The CODA [11] file system is based on such an approach, offering high availability through the use of file replication and client caching. Clients cache files locally and hence when a network or server fails, the client is able to maintain availability using its cached files. CODA relies on the assumption that in normal operation there will be a very small percentage of cache write clashes and this has been proven to be the case for normal usage.

A second example of a system offering high availability in a mobile environment is the Bayou [12] storage system. Unlike CODA the design of Bayou supports applications in which write operations are expected to conflict on a regular basis. To provide high availability Bayou replicates files which are kept consistent using a transaction based approach. A degree of optimism is built into the system by allowing writes to be regarded as tentative until committed and allowing committed writes to be rolled back when write conflicts are detected.

In fixed networking environments there are numerous examples of groupware applications which manage consistency . Such applications may be categorised as using either *centralised* or *replicated* architectures [13]. Groupware designed using a centralised architecture is simplest to implement because there is a single coordinating application whose output is propagated to group members. Systems based on this architecture (e.g. WSCRAWL 2.0 [14]) suffer from a central point of failure and a large amount of network traffic (because communications between the application and the group members is typically at the windowing system level rather than at the application level). Groupware designed using a replicated architecture has an application program replicated on every group member's machine (e.g. GROUPSKETCH [15] and GROUPDRAW [16]). The advantages of this approach are reduced network traffic and greater resilience to machine and network failure. Implementing groupware based on the replicated architecture is more difficult because appropriate synchronisation schemes must be employed in order to ensure that members' views remain consistent.

In addition to the specific applications cited above a number of toolkits have been written to simplify the task of implementing certain classes of groupware application. GroupKit [17] and DistEdit [18] are examples of such toolkits and each is based on a fully replicated architecture and uses the atomic broadcasting facilities provided by ISIS [19] to communicate between replicas. ISIS provides two main atomic multicast protocols which allow group updates to be received by replica objects in the correct order. The first protocol, *abcast* is a totally-ordered multicast protocol which requires the sender to block until acknowledgements are received from the entire group. The problem with using this protocol is that if there is a network failure then no further updates can proceed until full connectivity is restored. The second protocol, *cbcast* is perhaps the most useful for groupware applications and is used by GroupKit and DistEdit. This protocol provides a non-blocking causally-ordered multicast protocol and achieves this by using the concept of virtual synchrony [19]. If a network failure occurs when using cbcast to propagate an update, the sender will not be forced to block. However, those group members partitioned from the sender will have an inconsistent view to the rest of the group until full network connectivity is restored.

4: An ODP Compatible Service to Support Group Consistency

4.1: Overview of RM-ODP and ANSAware

The implementation of our service is based on APM Ltd.'s ANSAware software suite. This software suite is itself based on the ANSA architecture which has had a profound influence on the RM-ODP. The ANSA programming model is a location-independent object model where all interacting entities are treated uniformly as encapsulated objects. Objects are accessed through operational interfaces which define named operations together with constraints on their invocation. Interfaces may be to single objects or to groups of objects in which case the model provides *group transparency*. Objects are made available for access by exporting interfaces to a special object known as the *trader*. An object wishing to interact with this interface must then import the interface from the trader by specifying a set of requirements in terms of a interface type and attribute values. This will be matched against the available services and a suitable candidate selected. At this stage, an implicit *binding* is created to the object supporting the interface, i.e. a communication path is established to the object. Invocation of operations can then proceed.

To provide a platform conformant with the above programming model the ANSAware suite augments a general purpose programming language (usually C) with two additional languages. The first of these is IDL (Interface Definition Language), which allows interfaces to be precisely defined in terms of operations, arguments and results. The second language, DPL (Distributed Processing Language) is embedded in a host language, such as C, and allows interactions to be specified between programs which implement the behaviour defined by these interfaces. Specifically, DPL statements allow the programmer to import and export interfaces, and to invoke operations in those interfaces.

In the engineering infrastructure, the binding necessary for invocations is provided by a remote procedure call protocol known as REX (Remote EXecution protocol) or a group execution protocol known as GEX (Group EXecution Protocol). These are layered on top of a generic transport layer interface known as a *message passing service* (MPS). A number of additional protocols may be included at both the MPS and the execution protocol levels and these may be combined in a number of different configurations. The infrastructure also supports lightweight threads within objects so that multiple concurrent invocations can be dealt with.

All the above engineering functionality is collected into a single library, and an instance of this library is linked with application code to form a *capsule*. Each capsule may implement one or more computational objects. In the UNIX operating system, a capsule corresponds to a single UNIX process. Computational objects always communicate via invocation at the conceptual level but, as may be expected, invocation between objects in the same capsule is actually implemented by straightforward procedure calls rather than by execution protocols. ANSAware currently runs on a variety of operating systems platforms including various flavours of UNIX, VMS and MS-DOS/Windows.

4.2: Service Design and API

In a fixed networking environment with group members collaborating in close synchrony, the notion of transparent group invocation as provided by ANSAware works well because network failures are rare. However, in a mobile environment when network failure, and hence invocation failure, occurs more frequently, group transparency must be broken if the client is to be able to ascertain the cause of the failure and take appropriate action. For example, by breaking group transparency applications can provide information to the users regarding the state of the underlying network and hence they can make an informed decision on how to continue their collaboration.

Our services are designed to allow the group transparency paradigm to be selectively maintained, partially broken or completely broken at the application level. In order to retain complete group transparency a client of the group sends a message in the normal way and the group execution protocol will perform the standard message propagation to the group using a default policy. Transparency can be partially or totally discarded by establishing an *explicit binding* between the client and the group interface with an associated binding control interface. Through the binding control interface clients will be able to choose to partially discard group transparency by specifying the quorum to be used for deciding whether or not an invocation on the group has been successful, or to totally discard group transparency by specifying a *message profile* stipulating the required QoS to be used when propagating their next group invocation. In more detail, the binding control interface enables programmers to:

- Obtain the QoS of the binding, i.e. return the current values of the quorum and message profile.
- Set the QoS of the binding. This operation allows programmers to specify the desired quorum and optionally the message profile for group invocations.
- Set the collation policy to be associated with the

group. This operation allows the client to specify the function to be used for implementing the collation policy. The collation policy is responsible for determining which invocation result to return to the client from the set of invocation results obtained from the group.

- Register for QoS violations. Programmers can register for notification of violations in any of the QoS parameters irrespective of whether they led to overall invocation failure.

- Delete the binding (unbind).

The message profile is a matrix associating group members with a set of QoS (Quality Of Service) parameters. The parameters we currently envisage supporting are:

i) Temporal constraints which stipulate the time out period within which the group member must acknowledge receipt of the group invocation.

ii) Ordering requirements which stipulate whether or not the group member must receive group invocations in sequence.

iii) Reliability requirements which stipulate whether or not the group member must receive the group invocation.

iv) Cost requirements which stipulate the cost which the client is prepared to pay in order to have the group member receive the group invocation.

An example of the message profile structure is shown in figure 2 where the first column represents a group member's id, the second column represents the time constraint, the third column represents the required ordering, the fourth column represents the reliability guarantee and the last column represents the cost in an appropriate unit.

There will be occasions when group membership is increased before the client is able to update the message profile to take account of the new group members. In this situation, the current message profile will be updated automatically by using a set of default QoS parameters. The client will be able to stipulate these default parameters by entering them in the first row of the message profile.

MemberId	Time	Ordering	Reliability	Cost
0,	2,	ordered,	yes,	20,
1,	2,	ordered,	yes,	20,
2,	2,	unordered,	no,	20,
3,	2,	unordered,	no,	20

Figure 2: An example message profile

If a client's group invocation fails due to a QoS violation then this information will be reported back to the client via an appropriate error code. The client can then interrogate the binding to establish the cause of the failure. Note that not all QoS violations will result in overall failure of the invocation due to factors such as low quorums. However, clients will be able to register an interest in QoS violations which will enable them to be informed of all QoS violations.

Further consistency services which are needed to address the requirements described in section 2, e.g. state-transfer to new group members, will be layered on top of the group execution protocol.

4.3: Engineering Issues

Our services are based on a QoS driven group execution protocol called G-QEX. We are engineering G-QEX as part of our ongoing work to enhance the ANSAware distributed systems platform to enable it to operate efficiently in a mobile environment. As described in section 4.1, ANSAware currently supports the concept of the *interface group* [20] which is defined as a collection of interfaces which provide a service at a single interface. Interface groups use a transparent group execution protocol called GEX for object invocation. In ANSAware 4.1 only *active replica groups* are provided which means that when an invocation is made upon a group, each member of that group is required to service the invocation. GEX uses the standard ANSAware remote execution protocol REX to provide point to point links between the client and the group.

GEX is currently an unsuitable protocol for providing the various group consistency guarantees required to support mobile collaborative applications and hence we are replacing GEX with a QoS driven protocol G-QEX. In addition, earlier work has shown that the REX protocol is unsuitable for use in a mobile environment and hence we will use a new protocol QEX (which has been reported previously in [2], [21]) to provide the point-to-point links where necessary. QEX can operate over a diverse range of networks by adapting its behaviour to match the quality of the underlying network. This adaptation is achieved by gathering information on the number of retries and average delay time experienced over a given channel. Using this information QEX is able to adjust retry intervals and alter transmission rates to make the best use of the channel. QEX is also able to pass QoS information regarding the state of a channel to interested clients.

G-QEX will use the QoS information provided by QEX to provide an insight into the viability of group consistency guarantees. For example, suppose a client requires a guarantee that a particular group member should receive a group invocation within five seconds. If QEX has estimated that communication to that group member involves a ten second call set-up time then G-QEX can fail the group invocation immediately rather that cause the client to wait for five seconds before being notified of the failure. G-QEX will also be able to use channel throughput information obtained from QEX to detect whether an invocation of a given size can reach a particular group member in time.

It is intended that future versions of QEX will enable clients to receive information regarding the cost of using a given channel. This information will be used by G-QEX to enable it to take appropriate action when clients require a group consistency guarantee that involves cost. For example, suppose a client requires a guarantee that the cost of sending an invocation to a particular group member should not exceed twenty units and that the group member has network connection via a cellular phone. QEX will be able to supply information on the tariff charges for the group member's cellular connection and this cost information will enable G-QEX to calculate whether the invocation can be made without exceeding the stipulated cost. G-QEX will also be capable of making optimisations based on cost information when a client stipulates the quorum to be used when invoking a group operation. For example, if a majority quorum is specified then G-QEX will send the invocation to those group members connected by the lower cost channels until the quorum is reached.

5: Example Usage

As an example of how the consistency services might be used we shall use a scenario based around the requirements identified during the MOST project. Consider the scenario in which a group comprising two mobile field engineers and a control centre is collaborating using a shared GIS application. Initially the control centre wishes to use the shared GIS to hi-light to the first engineer the point at which they should expect to find damage to an electricity cable. To keep the second field engineer informed of the situation the control centre determines that the second engineer should also observe the suspected point of damage via their shared GIS. Because both field engineers are mobile each has an unreliable network connection. Before performing the hi-light operation, the control centre can use the consistency services to set the appropriate consistency guarantees for the propagation of the hi-lighting operation to the group. The control centre would require a guarantee that the hi-lighting operation performed on its shared GIS would be received by itself and the first engineer. The control centre would not in this case require a guarantee that the second engineer would receive the update. The profile in figure 3 could be used in this scenario (note that cost would not be an issue since the utilities use free wireless data services).

MemberId	Time	Ordering	Reliability	Cost
0,	5,	unordered,	no,	na,
1,	5,	ordered,	yes,	na,
2,	5,	ordered,	yes,	na,
3,	5,	unordered,	no,	na

Figure 3 : Message profile for a single important recipient

When propagation of the update is made to the group, the first field engineer is located in an area offering coverage for their PMR (Private Mobile Radio) service but the second field engineer is not. Given the profile used, despite the second engineer suffering a network disconnection, the propagation of the hilight operation would still be regarded as successful provided that the first engineer receives the update within five seconds.

The first engineer visits the location of the damaged cable and requests permission to switch power from the current circuit to an alternative one. The shared GIS can now be used by the first engineer to hilight which new distribution circuit he wishes to use. Performing such a switch could affect the work of the second engineer and therefore a guarantee is required that each member of the group should receive the highlight update. The first engineer would use the profile in figure 4.

MemberId	Time	Ordering	Reliability	Cost
0,	5,	unordered,	no,	na,
1,	5,	ordered,	yes,	na,
2,	5,	ordered,	yes,	na,
3,	5,	unordered,	no,	na

Figure 4 : Message profile for two important recipients

When the attempt is made to propagate the update to the group it will fail because the second field engineer is outside service coverage and so cannot receive the update. The first engineer can now make an informed decision regarding whether to delay until the second engineer enters an area of coverage or whether to continue with due caution.

The first engineer and the control centre make the decision to continue without the second engineer and decide to remove them from the group. The next stage for the engineer and control centre is to collaboratively produce a plan for fixing the damaged cable. This stage of close collaboration requires the use of telepointers in addition to a separate reliable voice channel to enable each collaborator to quickly see and respond to the other's ideas. In order to maintain some form of synchrony between the telepointers and the voice channel a policy is required which ensures that a telepointer update is only made if it is received within one second otherwise it is ignored. To achieve such a policy the engineer and control centre would each use the following profile:

MemberId	Time	Ordering	Reliability	Cost
0,	1,	unordered,	no,	na,
1,	1,	unordered,	no,	na,
2,	1,	unordered,	no,	na

Figure 5 : Message profile for time-critical message transmission

As the above scenario demonstrates, users may wish to change consistency requirements (and hence message profiles) during the life-time of a collaboration. This may be achieved implicitly by the application when, for example, users change their role within a collaboration or may be the result of an explicit user request via the application's user interface

6: Concluding Remarks

The MOST project identified a number of requirements for application level consistency in collaborative mobile applications. This paper has described the design of a group execution service (G-QEX) which is aimed at addressing these requirements. The service builds on our earlier work on QoS driven point-to-point remote execution protocols and allows application programmers to selectively break group transparency by specifying a range of QoS parameters for group invocation. These QoS parameters include temporal constraints on message delivery, ordering and reliability requirements and an overall cost figure which the client application is prepared to pay in order to ensure delivery to a given group member.

While there has been considerable research work on QoS driven group communication for continuous media types [22] there has been relatively little on QoS driven group execution protocols which we attribute to the relatively uniform levels of service found in most fixed networks. Consequently, many of the existing group execution protocols are unsuitable for use in a mobile environment. We are currently developing a prototype implementation of G-QEX and modifying the MOST application to exploit the benefits accruing from the use of this service.

7: References

[1] Greenberg, S., D. Marwood, "Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface", *Proc. of CSCW 94,* Chapel Hill, NC, U.S., October 1994.

[2] Friday, A., N. Davies, G.S. Blair, and K. Cheverst, "Extensions to ANSAware for advanced mobile applications.", *Proc. International Conference on Distributed Platforms,* Dresden, Germany, February 27-March 1, 1996

[3] Davies, N., G.S. Blair, A. Friday, A.D. Cross, and P.F. Raven, "Mobile Open Systems Technologies For The Utilities Industries", *Remote Cooperation - CSCW for Mobile and Tele-workers,* Editor: A. Dix, Springer Verlag (to be published).

[4] ISO, "Draft Recommendation X.901: Basic Reference Model of Open Distributed Processing - Part1: Overview and Guide to Use", *Draft Report* , 1992.

[5] APM Ltd., "An Introduction to ANSAware 4.0", APM Ltd., Cambridge, U.K., February 1992.

[6] Davies, N., G.S. Blair, K. Cheverst, and A. Friday, "Experiences of Using RM-ODP to Build Advanced Mobile Applications", *Distributed Systems Engineering Journal,* Vol. 2 No. 3, Pages 142-151, 1995.

[7] Ellis, C.A., S.J. Gibbs, G.L. Rein, "Groupware: Some issues and experiences", *Communication of the ACM,* Pages 38-58, January 1991.

[8] Grief, I., S. Sarin, "Data Sharing in Group Work", *ACM Transactions on Office Information Systems,* Vol. 5, Pages 187-211, April 1987.

[9] Ege, A., C.A. Ellis, "Design and Implementation of GORDION, an Object Base Management System", *Proc. 3rd International Conference on Data Engineering,* February 1987.

[10] Ceri, S., G. Pelagatti, "Distributed Databases - Principles and Systems", *McGraw Hill,* 1984.

[11] Satyanarayanan, M., J.J. Kistler, P. Kumar, M.E. Okasaki, E.H. Siegel, and D.C. Steere, "Coda: A Highly Available File System for a Distributed Workstation Environment", *IEEE Transactions on Computers,* Vol. 39 No. 4, Pages 447-459.

[12] Demers, A., K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and B. Welch, "The Bayou Architecture: Support for Data Sharing Among Mobile Users", *Proc. Workshop on Mobile Computing Systems and Applications,* Santa Cruz, CA, U.S., December 1994.

[13] Crowley T., E. Baker, H. Forsdick, P. Milazzo, and R. Tomlinson, "MMConf: An infrastructure for building shared applications", *Proc. Conference on Computer Supported Cooperative Work,* Los Angeles, California, U.S., October 1990.

[14] Wilson, B., "WSCRAWL 2.0: A Shared whiteboard based on X-Windows.", In *Designing Groupware for Real Time Drawing,* Editors: S. Greenberg, S. Hayne and R. Rada, McGraw Hill, 1994.

[15] Greenberg, S., R. Bohnet, "GroupSketch: A multi-user sketchpad for geographically-distributed small groups", *Proc. of Graphics Interface 91,* Calgary, Alberta, Canada, July 1991.

[16] Greenberg, S. M. Webster, R. Bohnet, "Issues and experiences designing and implementing two group drawing tools", *Proc. of the 25th Hawaii International Conference on System Sciences,* Kuwaii, Hawaii, IEEE Computer Society Press, January 1992.

[17] Roseman, M., S. Greenberg, "GroupKit: A groupware toolkit for building real-time conferencing applications", *Proc. of ACM Conference on Computer Supported Cooperative Work,* Toronto, Ontario, ACM Press, November 1992.

[18] Knister, M., A. Prakash, "DistEdit: A Distributed Toolkit for Supporting Multiple Group Editors", *Proc. of Conference on Computer Supported Cooperative Work,* Los Angeles, California, U.S., October 1990.

[19] Birman, K.P., "The Process Group Approach to Reliable Distributed Computing", *Technical Report,* Dept. of Computer Science, Cornell University, U.S., July 1991.

[20] Oskiewiez, E., J. Warne and M. Olsen, "A Model for Interface Groups", APM/TR.009.00, Advanced Networked Systems Architecture, Cambridge, U.K., 1990.

[21] Davies, N., G.S. Blair, K. Cheverst, and A. Friday, "Supporting Adaptive Services in a Heterogeneous Mobile Environment", *Proc. Workshop on Mobile Computing Systems and Applications,* Santa Cruz, CA, U.S., December 1994.

[22] García, F., D. Hutchison, A. Mauthe, and N. Yeadon, "QoS Support for Distributed Multimedia Communications", *Proc. International Conference on Distributed Platforms,* Dresden, Germany, February 27th - March 1st, 1996.