Response Times in N-user Replicated, Centralized, and Proximity-Based Hybrid Collaboration Architectures

Sasa Junuzovic
Computer Science Department
University of North Carolina at Chapel Hill
sasa@cs.unc.edu

Prasun Dewan
Computer Science Department
University of North Carolina at Chapel Hill
dewan@cs.unc.edu

ABSTRACT
We evaluate response times, in N-user collaborations, of the popular centralized (client-server) and replicated (peer-to-peer) architectures, and a hybrid architecture in which each replica serves a cluster of nearby clients. Our work consists of definitions of aspects of these architectures that have previously been unspecified but must be resolved for the analysis, a formal evaluation model, and a set of experiments. The experiments are used to define the parameters of and validate the formal analysis. In addition, they compare the performances, under the three architectures, of existing data-centric, logic-centric, and stateless shared components. We show that under realistic conditions, a small number of users, high intra-cluster network delays, and large output processing and transmission costs favor the replicated architecture, large input size favors the centralized architecture, high inter-cluster network delays favor the hybrid architecture, and high input processing and transmission costs, low think times, asymmetric processing powers, and logic-intensive applications favor both the centralized and hybrid architectures. We use our validated formal model to make useful predictions about the performance of the three kinds of architectures under realistic scenarios we could not create in lab experiments.

Categories and Subject Descriptors
C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *distributed applications, client/server.* C.4 [**Performance of Systems**] Performance Attributes.

General Terms
Algorithms, Performance, Measurement, Experimentation.

Keywords
Collaboration architecture, analytical model, response time.

1. INTRODUCTION
Two main architectures have been used to support the sharing of a program among multiple users: centralized (client-server) and replicated (peer-to-peer) [5]. In the centralized architecture, the shared program executes on a computer belonging to one of the collaborators, receiving input from and broadcasting output to all users. In the replicated architecture, a separate replica of the program executes on the computer of each user, receiving input from all users and producing output for only the local user. We call a computer which is (not) running the program a *master* (*slave*) *computer*, and the corresponding user a *master (slave) user*. Thus, in a centralized architecture, one computer is a master while the rest are slaves, and in a replicated architecture, all computers are masters. In this paper, we also consider a third type of architecture, the hybrid architecture, in which more than one computer, but not all, are masters. Such architectures are supported by a few frameworks, such as [3][8].

Given a distribution of collaborators, multiple centralized and hybrid architectures are possible. In the hybrid architectures we consider, a slave computer is always served by the replica that is nearest to it, that is, the one with the smallest network delay. We refer to such a hybrid architecture as *proximity-based* and the slaves served by a master as a *cluster*.

The choice of the architecture affects the semantics, correctness, and performance of the shared program [5]. In this paper, we focus on performance, specifically, response times, and assume that correctness issues such as externalities [2] have been addressed by the system implementing the architecture. Research shows that response times should ideally be less than 50ms [10], but in several commercial collaboration systems, this goal is not met. For example, the response times to operations made by a remote user of a centralized LiveMeeting/Webex shared application is intolerable as it can sometimes take several seconds. Replicated architectures offer the hope for better response times, but all commercial implementations of this architecture with which we are familiar do not offer the option of transmitting incremental changes to a shape as it is dragged because of performance fears. Similarly, unlike the early P2P "talk" programs, state-of-the-art IM tools do not support the option of incremental sharing of typed text partly because of the fear that IM servers would be overloaded.[1] However, several usable and useful research systems support incremental sharing of drag operations and text edits [6]. Thus, the key is to systematically characterize these scenarios based, for example, on the number of participating users and network delays between the users so that an appropriate architecture can be chosen for a particular collaboration. This goal has motivated previous work in both empirical and formal analysis of the performance of collaboration architectures.

CSCW'06, November 4–8, 2006, Banff, Alberta, Canada.

[1] Another reason is the assumption that this feature would not be useful. This assumption can be verified only by providing it as an option when it will not have an impact on the performance.

Ahuja et al. [1] performed experiments to compare the network load imposed by the centralized and replicated implementations of a shared drawing program. Junuzovic, Chung, and Dewan [7] addressed response, feedthrough, and task completion times in centralized and replicated architectures. They developed a formal model comparing the two architectures and performed experiments validating it using a checkers application. This work assumed two-user collaborations, a constant cost of processing each user operation, constant think times before each command, zero cost of transmitting inputs and outputs, and no type-ahead.

This paper extends previous work in several ways. It considers collaborations involving an arbitrary number of users and adds hybrid architectures to the mixture of architectures considered, which make sense only in collaborations involving more than two users. Moreover, the experimental evaluation considers three classes of shared applications: 1) Logic-centric, which process computationally-heavy input commands; 2) Data-centric, which distribute large amounts of data; and 3) Stateless, which do not process computationally heavy input commands or distribute large amounts of data. Like [7], it also develops a formal performance model but focuses only on response times. It relaxes all assumptions of [7] except no type-ahead. It considers several new collaboration parameters (such as transmission and output processing costs) that become important when the assumptions are relaxed. Finally, because of the relaxed assumptions and extended architecture and application set, it makes several new predictions about the optimal architecture under realistic collaboration conditions.

This paper makes its own set of assumptions, which we describe in the next section. Following this, we develop the response time equations comprising our analytical model. We then describe how we validate our model through experiments. Finally, we end with conclusions and directions for future work.

2. ASSUMPTIONS

Previous literature on the definitions of the centralized, replicated, and hybrid architectures leave unaddressed several implementation aspects that are important in this evaluation. A thorough exploration of different design choices for these aspects is beyond the scope of this evaluation work. Here we assume one approach for resolving each of the issues and defend it using primarily analysis-based arguments.

The centralized, replicated, and hybrid architectures we consider assume that an interactive application consists of a program component and a user-interface component. The user-interface component is never shared and hence always replicated. The program component is the shared component and may be replicated on one or more master computers.

One issue left undefined is whether or not a user-interface component of a master computer can directly interact with files accessible to the local program component. We allow such access because, as we show in our experiments, it can improve the response times for commands entered by master users, and we would like to be able to formally analyze the degree of the improvement. The idea of different user-interface components implementing different algorithms is not new; for instance, it has been advocated to create different users of mobile and desktop computers [9].

A related question is whether each replicated program component in a replicated or hybrid architecture has access to files needed to support the collaboration before the collaborative session begins. We do not make this assumption in order to accommodate realistic situations – in particular, a PowerPoint presentation that is continuously updated until the start of the lecture. Instead, we assume that the necessary files are sent from the computer of the first inputter to all masters as part of the first input command.

Previous work has well established the set of operations a program component is responsible for (which are processing input commands and transmitting input and/or output to other computers) but does not indicate whether the operations are carried out by a single thread or separate threads. We assume the former mainly because the latter makes it impossible to model the response times without making some platform-specific assumptions about the scheduling of the threads. This assumption does not imply worse response times as on a single-processor computer there is no context-switching overhead. On the other hand, multi-threading can improve performance by allowing a thread to be scheduled while another is blocked on a communication operation.

When all operations are carried out by a single thread, we must determine the order in which they are carried out. Although other sequences also make sense, Figure 1 gives the orders we impose in the three architectures. In this figure, by transmitting data we mean sending it to the network and not waiting for acknowledgement from the receiving computer(s). Thus, the response times in all three algorithms are independent of network delays. In all three algorithms in Figure 1, the last step is handing output to the local user. The reason is that, in general, there will be a network delay before another program (user-interface) component receives an input command (output). Our scheduling ensures that receiving program (user-interface) components can process the received input command (output) concurrently with

```
Centralized Architecture
1: Wait for next input command
2: Process input command
3: if my slave user entered input command
    transmit output to inputting user
4: Transmit output to my other slave users
5: Give output to my local user
6: Goto 1
```

```
Replicated Architecture
1: Wait for next input command
2: if input command from local user
    transmit command to other master users
3: Process input command
4: Transmit output to local user
5: Goto 1
```

```
Hybrid Architecture:
1: Wait for next input command
2: if command from local/my slave user
    transmit command to other master users
3: Process input command
4: if my slave user entered input command
    transmit output to inputting user
5: Transmit output my other slave users
6: Transmit output to local user
7: Goto 1
```

Figure 1. Program-component pseudo-code for the three architectures

the sending program component, which both increases the real concurrency of the system and reduces the "can you see it now" questions. If, in the centralized and hybrid architectures, after processing an input command, the program component first handed the output to the local user-interface, the local user-interface would complete processing the output before any of the program component's slave user-interfaces even received it, thus reducing the real concurrency of the system and increasing the divergence in the user-interfaces of the collaborators. Similarly, if, in replicated and hybrid architectures, a master computer first processed an input command and then transmitted it to other master computers, the real concurrency would be reduced and the user-interface divergence increased.

Like all published implementations of the replicated architecture [5], the replicated and hybrid algorithms above assume that the program component (a) is deterministic, that is, produces the same result given a series of input commands, and (b) does not implement atomic broadcast to ensure good response times, relying instead on floor control to prevent concurrent input or some application-specific scheme such as operation transformations [11] to do consistent real-time merging of concurrent input. Furthermore, we assume that the communication cost of distributed concurrency control is negligible.

As mentioned before, we consider only the *proximity-based* hybrid architecture. A further requirement, imposed on both centralized and hybrid architectures, is that a master computer is at least as powerful as any of its slaves. As mentioned before, we assume no type-ahead.

In our experiments, we make several additional assumptions not related to the architecture design and implementation issues. In the hybrid architectures we use for our experiments, we assume that the number of master computers, and therefore, clusters, is small. This assumption allows us to illustrate the advantages of each architecture in realistic scenarios without limiting our model. Also, we experiment with only one example of each of the three shared component categories we consider: checkers for logic-centric, PowerPoint for data-centric, and IM for stateless.

As stated above, research shows that response times should ideally be less than 50ms as humans can perceive values higher than this [10]. We assume that this implies that each 50ms increment is noticeable.

Finally, our model assumes no latecomers. In particular, it does not consider the effect of bringing a latecomer's state up-to-date on the response times of other users' commands.

3. FORMAL ANALYSIS

The centralized, replicated, and hybrid architectures can be considered special cases of a general architecture in which there are one or more master computers and each master computer has zero or more slave computers. A general architecture is (a) centralized if there is exactly one master computer, (b) replicated if each computer in the collaboration is a master, and (c) hybrid if it is not centralized or replicated, that is, if more than one computer, but not all, are masters.

We first derive response-time equations for a general architecture, which we then apply to hybrid, replicated, and centralized architectures. In our analysis of general architectures, we first consider the response time for the master and then the slave users. For slave users, the response time for the first and subsequent commands in the collaborative session must be treated differently because in the latter case, the shared program may still be processing a previous command when the next command arrives.

Response time of an input command is defined to be the time that elapses from the moment a user enters the input command to the moment that user sees the output for the input command. It depends on several factors identified in Table 1, which we motivate as we use them.

Table 1. Collaboration parameters that impact response time

Parameter	Description
p_i	Processing **p**ower (MHz) of $user_i$'s computer.
w_i^{in} (w_i^{out})	**W**ork (number of CPU cycles) required to process input (output to) command i.
x_i^{in} (x_i^{out})	**X**mission cost (number of CPU cycles) required to send input (output to) command i to one computer.
t_i	**T**hink time before input command i.
$d(i,j)_c^{in}$ ($d(i,j)_c^{out}$)	Network **d**elays between $user_i$'s and $user_j$'s computers at the moment input (output to) command c is entered (xmitted). $d(i,i)_c^{in} = d(i,i)_c^{out} = 0$ for all c.
k	Number of master computers.
s_i	Number of **s**lave user-interface components computer i has.
$b^{out}(m,i,j)$	Time that elapses from the moment computer m begins transmitting output to command i to the moment it transmits the output to $user_j$.

3.1 General Architecture: Master Users

Based on Figure 1, the response time to command i entered by master $user_m$ includes the cost of transmitting the command to other masters, processing the command, transmitting the output to slaves, and giving the output to the local user. It is given by the following equation:

$$[Eq1]\ Resp_i Mast_m = (k-1)x_i^{in}/p_m + w_i^{in}/p_m + s_m x_i^{out}/p_m + w_i^{out}/p_m$$

We denote the cost, measured in CPU cycles, of (a) transmitting command i to another computer by x_i^{in}, (b) processing input command (output) i by w_i^{in} (w_i^{out}), and (c) transmitting output to input command i to a single computer by x_i^{out}. p_m is the processing power, measured in MHz, of $user_m$'s computer. k is the number of master computers. s_m is the number of $user_m$'s slaves. The first, third, and fourth terms are needed in data-intensive applications such as PowerPoint as the cost of transferring data is proportional to its size. The first and third terms are necessary in large collaborations such as a lecture as they depend on the number of users. Finally, the second term is necessary in logic-intensive applications, such as Checkers, running on lower-end computers such as a cell phone. Thus,

depending on the nature of the collaboration, this equation can be simplified by removing one or more terms.

We allow processing and data-transfer costs to vary from command to command based on experience with the applications with which we experimented. For example, in a PowerPoint presentation, the output of a next-animation command is significantly smaller than that of a next-slide command, and in our Checkers program, the cost of processing a telepointer move is significantly smaller than that of a user's board move.

3.2 General Architecture: Slave Users

Consider the case in which the first input command in the session is entered by slave $user_j$ whose master is $user_m$. As $user_j$'s master computer m is remote, $user_j$'s computer must transmit this input command, x_1^{in}/p_j, and the input command must traverse the network to reach computer m, taking time $d(j,m)_1^{in}$. In addition, computer m must transmit the output to the command, x_1^{out}/p_m, and the output must traverse the network to reach computer j, taking time $(d(m,j)_1^{out})$. Furthermore, as a master computer transmits the output to the inputting slave before it delivers it to other users, the output transmission cost to other slaves does not contribute to the response time. Thus, we add four terms to [Eq 1] and remove one from it to derive the response time for this command:

$$[Eq2]\ Resp_i Slave_j = x_1^{in}/p_j + d(j,m)_1^{in} + (k-1)x_1^{in}/p_m + w_1^{in}/p_m + x_1^{out}/p_m + d(m,j)_1^{out} + w_1^{out}/p_j$$

The subscript 1 in the terms above denotes the fact that we are calculating the response time for the first command of $user_j$. Calculating the response times for other commands is different and more complicated. The reason is that if $user_j$ does not think for a long time after the output of an input command, computer m may not be finished transmitting the output to this command by the time the $user_j$'s next command reaches it. This happens when the time, t1, at which command i reaches computer m is less than the time, t2, at which computer m finishes processing the output to command $i-1$. Figure 2 shows this case.

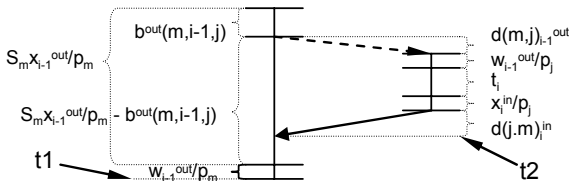

Figure 2. Master computer not ready to accept command.

Let $b^{out}(m,i,j)$ denote the time that elapses from the moment computer m begins transmitting output to command i to the moment computer m transmits the output to $user_j$. In the above figure, the time the output to command $i-1$ takes to reach $user_j$ $(d(m,j)_{i-1}^{out})$ plus the time $user_j$'s computer takes to process the output (w_{i-1}^{out}/p_j) plus the time $user_j$ thinks for about the output before entering command i (t_i) plus the time $user_j$'s computer takes to send input command i (x_i^{in}/p_j) plus the time input command i takes to reach computer m $(d(j,m)_i^{in})$ is lower than the time computer m takes to complete transmitting output to command $i-1$ to all remaining slaves after transmitting it to $user_j$, $s_m x_{i-1}^{out}/p_m - b^{out}(m,i-1,j)$, and processing the output locally, w_{i-1}^{out}/p_m. Hence, the response time equation for input commands by $user_j$ can be generalized to the following,

$$[Eq3]\ Resp_i Slave_j = \max\{0, (s_m x_{i-1}^{out}/p_m - b^{out}(m,i-1,j) + w_{i-1}^{out}/p_m) - (d(m,j)_{i-1}^{out} + w_{i-1}^{out}/p_j + t_i + x_i^{in}/p_j + d(j,m)_i^{in})\} + x_i^{in}/p_j + d(j,m)_i^{in} + (k-1)x_i^{in}/p_m + w_i^{in}/p_m + x_i^{out}/p_m + d(m,j)_i^{out} + w_i^{out}/p_j$$

As we see in the Figure 2, computer m can not accept command i immediately if

$$[Cond1]\ t_i < s_m x_{i-1}^{out}/p_m - b^{out}(m,i-1,j) + w_{i-1}^{out}/p_m - d(m,j)_{i-1}^{out} - w_i^{out}/p_j - x_i^{in}/p_j - d(j,m)_i^{in}$$

because computer m must first complete transmitting the output for command $i-1$ and processing the output locally. It is important to consider the scenario of [Cond 1] because we discovered in our user-interaction logs, even without considering telepointing actions, that the think times can be very small (less than 1 second). It is also important to consider the opposite scenario because our logs also showed that think times can be very large (several minutes).

As stated earlier, the centralized and replicated architectures are just special cases of the general architecture with 1 and n master users, respectively. We next give the response times for these two architectures by simplifying equations 1 and 3, first for replicated, and then for centralized. For hybrid architectures, these equations directly apply.

3.3 Replicated Architecture

In a replicated architecture, all computers are master computers, and receive input from all users and send output only to the local user. Thus, $k = n$ in the equations we derived above for the general architecture. Eq 1 simplifies to the following for the replicated architecture response time.

$$[Eq4]\ Resp_i Mast_m = (n-1)x_i^{in}/p_m + w_i^{in}/p_m + w_i^{out}/p_m$$

The time required to send output to slave user-interfaces, $s_m x_i^{out}/p_m$, becomes zero because all users are masters. If we assume that only two users exist, that the cost of sending an input command to a remote computer, x_i^{in}, is negligible, that the amount of work required to process command i is constant and denote it as w, and that output processing cost, w_i^{out}, is zero, which then matches the assumptions of the model presented in [7], then Eq 4 simplifies to

$$[Eq5]\ Resp_i Mast_m = w/p_m$$

which is the same as the replicated architecture response time equation in [7].

3.4 Centralized Architecture

In a centralized architecture, only one user is a master user while the rest are slave users. The general architecture master user response time equation Eq 1 thus becomes the following

$$[Eq6]\ Resp_i Mast_m = w_i/p_m + (n-1)x_i^{out}/p_m + w_i^{out}/p_m$$

Several simplifications can also be made to Eq 3 to calculate the response time for slave users. The time required to transmit the input command to other master computers, $(k-1)x_i^{in}/p_m$, becomes zero because the number of master computers, k, is one. Furthermore, all but one user is a slave users, so $s_m = n-1$. For

the slave user, $user_j$, the general architecture slave user response time equation thus becomes the following

$$[Eq7]\ Resp_i Slave_j = \max\{0, ((n-1)x_{i-1}^{out}/p_m - b^{out}(m,i-1,j) + w_{i-1}^{out}/p_m) - (d(m,j)_{i-1}^{out} + w_{i-1}^{out}/p_m + t_i + x_i^{in}/p_j + d(j,m)_i^{in}\} + x_i^{in}/p_j + d(j,m)_i^{in} + w_i^{in}/p_m + x_i^{out}/p_m + d(m,j)_i^{out} + w_i^{out}/p_j$$

If we once again match our model to the model presented in [7], by assuming that only two users exist, the transmission costs are negligible, the amount of work required to process a command is constant, the output processing costs are zero, and the delays, d, between two users are constant, then equations 6 and 7 simplify to

$$[Eq8]\ Resp_i Mast_m = w/p_m$$

$$[Eq9]\ Resp_i Slave_j = 2d + w/p_m$$

which are the same as the centralized architecture master and slave response time equations in [7]. Below we simplify our equations in other ways to cover a variety of realistic scenarios.

4. EXPERIMENTS AND PREDICTIONS

The response time equations are sufficient to compare how well the different architectures perform under different conditions such as large network delays, think times, transmission costs, and processing-power differences. However, it is important to also perform experiments to (a) validate the general equations and predictions that can be made by them and (b) get an idea of the degree by which the architecture performances would differ under realistic conditions. This, in turn, requires us to identify suitable values for the parameters of the equations, which ideally, should reflect reality. These parameters can be divided into system parameters, such as network delays and processing powers, and task parameters, such as processing cost and number of observers.

There are a number of ways to obtain task-parameter values. Under the live-interaction approach, users perform a collaborative task multiple times as the architecture and system parameters are varied between the runs. However, since the users cannot be relied upon to perform exactly the same actions and have the same think times across different collaborative sessions, this approach is impractical. Another approach is to create synthetic logs in which the task-parameter values are set using some mathematical distribution such as Poisson's. Unfortunately, synthesized parameter values are not certain to reflect reality. A third approach is to use actual collaboration logs and assume that they are independent of system parameters such as computers used and network delays. A problem with this approach is that such logs are not publicly available and a large number of them may be required to ensure that a wide range of collaboration parameters is covered. A fourth approach, which is a combination of the actual and synthetic log approaches, is to synthesize collaboration parameters using data from actual collaboration logs. Naturally, this approach is not as representative of reality as using actual logs.

We used a combined approach in which the values of all the task-parameters, except the number of observers, were those that actually occurred in collaborations. By observers we mean users who did not input any commands and thus did not influence the logs we collected. Our experiments exactly mimic reality only if a user's actions are not influenced by observers. To understand the impact of different number of observers, we varied their number from 2 to 31 in our logs. In our experiments, the number of observers was sometimes less and sometimes more than the actual observers. While 31 is less than the maximum number of observers in our recordings, it is large enough to show architecture response time differences and project response times for collaborations involving larger numbers of observers.

We analyzed two existing PowerPoint presentation recordings in which there was only one presenter and about thirty to sixty observers. In addition, we recorded two chat-room sessions and one collaborative checkers game. These recordings contain actual users' actions – PowerPoint commands, chat messages, and checkers moves – and actual think times between these actions. IM, Checkers, and PowerPoint turned out to be a good choice of applications for which to analyze actual logs for three reasons: 1) the collaboration parameters we measured in these logs were fairly wide spread, 2) they are popular and represent the kind of tasks users do on a daily basis, and 3) their shared components are fundamentally different: IM is stateless, Checkers is logic-centric, and PowerPoint is data-centric. Based on the analyzed recordings, we were able to create 36 logs for our simulations.

The checkers engine used in the actual tasks was transformed into a collaborative program using an infrastructure that has facilities for logging and replaying commands. Therefore, extracting the task parameters from the generated checker logs was relatively simple. The chat programs we logged were the ones implemented by the chat rooms we observed. We ran them under Microsoft Live Meeting 2005 and used its screen-recording capabilities. As a result, we had to use a tedious manual process to extract the think times and input command messages in the sessions – analyzing one ten-minute recording required two hours of work! The checker commands were replayed directly to the program used in the actual task using the capabilities in the infrastructure to replay stored commands. To replay the chat commands, we used the replay-supporting infrastructure to create our own version of the chat application. To replay the PowerPoint commands, we had to bridge the gap between our Java-based replay-supporting infrastructure and the PowerPoint application. We used the J-Integra library to create this bridge and relay the replayed commands to the PowerPoint application.

The system parameters, processing powers, and network delays, also have to be realistic. All machines were PCs, 7 of which were running Windows XP SP2 and 1 Windows 98. Each machine must be dedicated to a user if the response times for the user are being measured, though a machine could simulate multiple observers. We simulated 2-31 observers on 1-5 computers, maximum of 16 per computer and/or 1-7 active users on the remaining computers. The total number of users was never more than 32. When comparing the three architectures, we used the same set of computers. As mentioned above, we refer to the set of slaves served by a master as a cluster. In our experiments, we never created more than two master computers, that is, clusters in a hybrid architecture. Our 8 computers consisted of a P2 laptop (366MHz), two P3 desktops (866MHz and 1GHz), and six P4 2.4GHz desktops. These are all in-use computers in our department, though we deliberately chose ones with widely differing processing powers to observe the effect of asymmetric computing powers. Even if everyone uses state-of-the-art devices, they might use different classes of devices such as cell phones, palmtops, and different kinds of desktops. The P2 laptop was intended to simulate next-generation mobile computers.

We performed each simulation five times and reported the average performances. We show the averages using a mixture of graph and tabular forms, as each representation has its advantages. We do not show statistics other than averages to avoid cluttering the representation. We removed any "outlier" entries from the average, caused for instance, by operating system process scheduling timing issues. To reduce these issues, we removed as many active processes on each system as possible. In order to control variability in network delays in our experiments, we ran them on our local 100Mbit LAN with 7 computers and an 8th laptop with wireless communication capabilities. Based on pings done to remote computers, we added 72ms and 162ms to the LAN delays (0ms) to simulate half the round-trip time from a U.S. East Coast LAN-connected computer to German LAN-connected and modem-connected computers, respectively.

We will say that the difference between two response times is significant if it is greater than 50ms because, based on our assumption given earlier, this means users will be able to recognize that one of the response times is worse than the other. We next describe our predictions and experimental-validations. In the equations relevant to the each result, we only include terms that have a non-zero impact on response time.

4.1 Number of Users

Intuitively, as the number of users increases, the performance of hybrid and centralized architectures should improve relative to the replicated architecture because in a replicated architecture, a program component, before delivering output to the local user, must transmit input to all other users. Our equations predict this if the following conditions are met: (a) the think times are large to guarantee that a program component is always ready to accept the next input command when it arrives, (b) the number of clusters in the hybrid architecture is low and does not change as the number of users increases, (c) the network latencies between all users are low, (d) the cost of a particular output is the same for all users in all three architectures, (e) the cost of transmitting a single input command or a single output is negligible, and (f) the inputting user is a slave in the centralized and hybrid architectures. Assumption (e) does not imply that the cost of broadcasting a single input or output or sending multiple inputs or outputs is negligible. The relevant response time equations from our model simplified to account for the above assumptions are:

$$[\text{Rep}]\ \text{Resp}_i\text{Mast}_j = (n\text{-}1)x_i^{in}/p_j + w_i^{in}/p_j$$

$$[\text{Hyb}]\ \text{Resp}_i\text{Slave}_j = (k\text{-}1)x_i^{in}/p_m + w_i^{in}/p_m$$

$$[\text{Cent}]\ \text{Resp}_i\text{Slave}_j = w_i^{in}/p_m$$

The replicated response time equation contains the term $(n-1)x_i^{in}/p_j$, the time required to broadcast an input command to all the other computers. Thus, the replicated architecture response time should increase with the number of users. In the hybrid architecture equation, if the master computers are powerful, the input broadcasting costs, $(k-1)x_{i-1}^{out}/p_m$, are close to zero because we assumed that the number of clusters is low and constant. Hence, increasing the number of users does not impact the hybrid architecture response time. Thus, just like the centralized architecture response time, the hybrid architecture response time is independent of the number of users.

To verify this prediction and see if the response time differences are significant, we replayed one of the PowerPoint recordings using actual think times and the architectures shown in Figure 3. The results of measuring the response to the "next animation" commands are shown in Figure 4 (left) and confirm our analysis. As the number of observers increased from 8 to 16 to 32, the replicated architecture response time increased by 128.1-90=38.1ms, while the centralized and hybrid response times remained constant at 84.2(±1.7)ms.

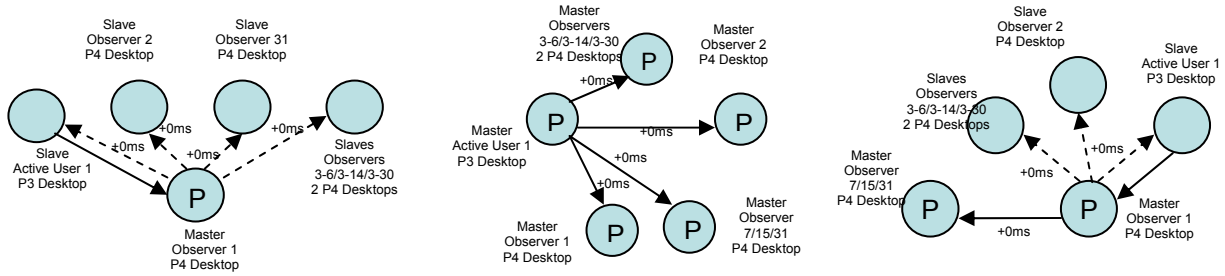

Figure 3. Centralized (left), Replicated (center), and Hybrid (right) architectures. The "+0ms" labels denote the delays added to the LAN delays. Solid lines depict inputs and dashed lines depict outputs. The letter inside the master computers circles denote if PowerPoint (P), Checkers (C), IM (I), or Telepointing (T) logs were simulated.

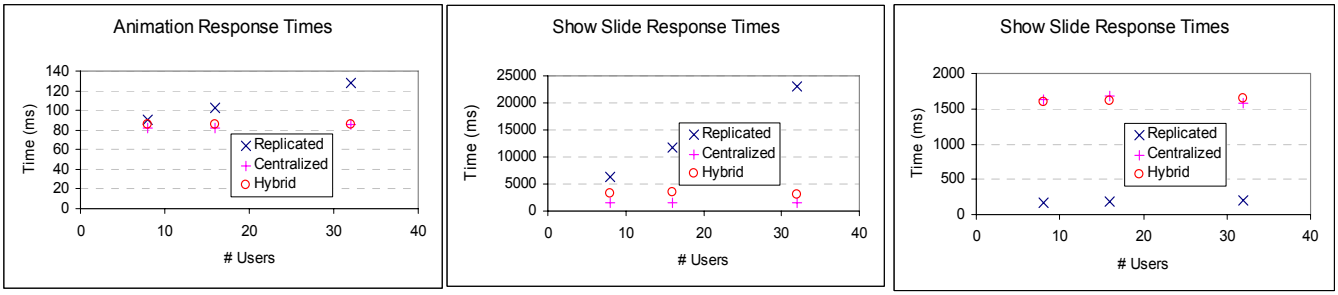

Figure 4. Next animation (right), start presentation (center), and show slide (right) response times

Thus our measurements showed no significant response time differences but confirmed the trend predicted by the equations, thereby validating them. The validated equations, in turn, can project, based on the measured values, when the difference would be significant. The 8 and 32 user measurements can be substituted into our equations as

$$[\text{Rep } 8] = (8\text{-}1)x_i^{in}/p_j + w_i^{in}/p_j = 90.0\text{ms}$$

$$[\text{Rep } 32] = (32\text{-}1)x_i^{in}/p_j + w_i^{in}/p_j = 128.1\text{ms}$$

$$[\text{Hyb } 8] = [\text{Hyb } 32] = [\text{Cent } 8] = [\text{Cent } 32] = 84.2\text{ms}$$

As mentioned before, the centralized and hybrid architecture response times are independent of the number of users. Hence, the values for their equations with 8 and 32 users are all equal and constant. The replicated architecture response time equations provide

$$[\text{Rep32}] = [\text{Rep8}] = (32\text{-}1)x_i^{in}/p_j - (8\text{-}1)x_i^{in}/p_j \Rightarrow x_i^{in}/p_j = 1.59\text{ms}$$

Thus, each additional user increases the replicated architecture transmission time, and hence response time, by 1.59ms. A response time that is significantly worse than the centralized and hybrid architecture response times of 84.2ms should be at least 134.2ms. Since the replicated architecture response time with 32 users was 128.1ms, we can project how many more users, n′, would increase the response time to 134.2ms or higher which would make the replicated architecture response time significantly worse than the centralized and hybrid architecture response times. The projection calculation is

$$n' = (134.2\text{-}128.1)/1.59 = 3.8 \Rightarrow n' = 4$$

Thus, with 36 or more users, the replicated architecture would provide a significantly worse response time than the centralized and hybrid architectures.

Our model also predicts that the higher the cost of broadcasting an input command, the faster the replicated architecture response time increases as the number of users increases. To verify this prediction, we compare the response times to the first input command in the session with the "next animation" command response times. As mentioned earlier, we assume that when a master computer receives the first input command, it must transmit the entire presentation file to all the other master computers. Transmitting an entire file has a much higher cost than transmitting a command to show the next animation. Figure 4 (center), which shows the response times to the first input command (start presentation) as the number of users increases, confirms our analysis. The replicated architectures response time to the first input command increases by 23044-6196=16484ms from 8 to 32 users. This gives is an average response time increase of 702ms per user, which is much higher than the response time increase of 1.59ms per user for next animation commands. On the other hand, the centralized architecture response time is not affected by the number of users because the output is always first transmitted to the slave user which sent the input command and only then to the remaining slave users. For the same reason, the hybrid architecture response time does not increase with the number because we assume that the number of clusters in the hybrid architecture does not increase with the number of users. The hybrid architecture response time is not as good as in the centralized architecture because before processing the input command, the master computer of the inputting user must first transmit the PowerPoint file to the other master computer.

One of the assumptions we made to make the prediction above is that think times are large. This prediction can also be made by substituting this assumption with the assumption that output processing costs are large enough to guarantee that a program component is always ready to accept the next input command when it arrives. The relevant equations from our model remain unchanged.

To verify this prediction under the substituted assumption and see if the response time differences are significant, we replayed a log of a telepointer motion that circles one of the checkers pieces. We use 0ms think times and the hybrid architecture shown in Figure 5 (top), a centralized architecture in which a Observer 1's P4 desktop hosts the application, and a replicated architecture. The response time results to telepointing commands are shown in Figure 5 (bottom) and confirm our analysis.

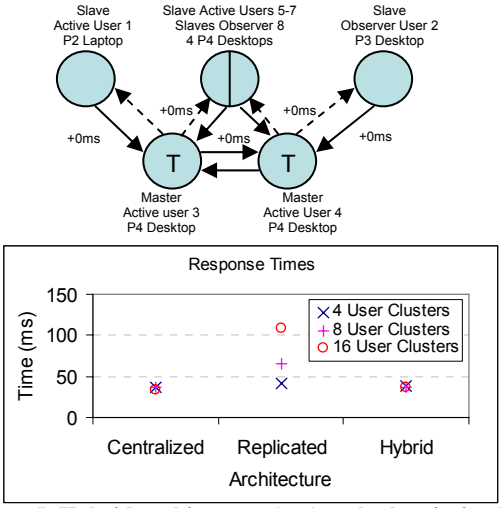

Figure 5. Hybrid architecture (top) and telepointing P2 laptop response time results (bottom).

Interestingly, with as many as 32 users, the centralized and hybrid architectures response times were less than 50ms, showing that the sharing of incremental changes can be supported by these architectures without hurting response times. Even more interestingly, with as many as 8 users, the replicated architecture response times were not significantly worse than in the centralized and hybrid cases even when the slow P2 laptop was responsible for input transmission. Thus, supporting incremental changes without hurting performance is possible in all three architectures. The design choice not to offer the option of supporting the sharing of incremental changes should be made using some other metric.

We see above the symbiotic relationship between the formal model and experimental analysis. The latter validates the former, while the former, in turn, can be used to make predictions about realistic scenarios that cannot be simulated in the laboratory. For example, we can determine what happens to the response times if the processing power increases tenfold and/or the number of users increases hundredfold.

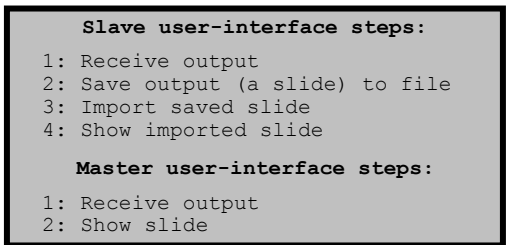

Figure 6. PowerPoint output processing steps

4.2 Output Cost Differences

As mentioned earlier, we assume that a master user-interface can directly access files available to the local program component. Such access can substantially reduce output processing costs in the PowerPoint case. The sequence of steps slave and master user-interfaces carry out when processing a PowerPoint output is given in Figure 6.[2] We found that the difference between the times required to take the two sequences of steps can be as large as two seconds. In this example, from the point of view of a slave user in the centralized or hybrid architecture, it seems better to support the replicated architecture.

We use our equations to state a general result that precisely makes the informal argument above. Assume that (a) the think times are large to guarantee that a program component is always ready to accept the next input command when it arrives, (b) the number of clusters in the hybrid architecture is low and does not change (c) the network latencies between all users are low, (d) all inputting users are slave in the centralized and hybrid architectures, (e) the number of users is low, (f) the cost of transmitting a single input command is negligible, and (g) the output transmission and processing costs to and for slave users are higher than for the master users. We can predict that these conditions favor the replicated architecture. The relevant equations from our model are

$$[\text{Rep}]\ \text{Resp}_i\text{Mast}_j = (n\text{-}1)x_i^{in}/p_j + w_i^{in}/p_j + w_i^{out}/p_j$$

$$[\text{Hyb}]\ \text{Resp}_i\text{Slave}_j = (k\text{-}1)x_i^{in}/p_m + w_i^{in}/p_m + x_i^{out}/p_m + w_i^{out}/p_j$$

$$[\text{Cent}]\ \text{Resp}_i\text{Slave}_j = w_i^{in}/p_m + x_i^{out}/p_m + w_i^{out}/p_j$$

The output processing cost, w_i^{out}, affects all three equations. Also, the hybrid and centralized equations contain an additional cost for transmitting output, x_i^{out}. Since output processing costs are high for slave user-interfaces and low for master user-interfaces, the replicated architecture will be impacted less than the centralized and hybrid slave user response times.

To verify this prediction and see if the response time differences are significant, we replayed the same PowerPoint logs as in the previous result, again using actual think times and the architectures shown in Figure 3, only this time, we consider the "show slide" input commands. The results of the experiments are shown in Figure 4 (right) and revealed that as the number of observers increased from 8 to 16 to 32, the replicated architecture response time increased by a total of 197.3-163.7=33.6ms (1.4ms per user). The centralized and hybrid architecture response times are more difficult to interpret as they varied between 1573.5ms and 1676.9ms. We explain why they varied next.

We measured, independently of our infrastructure, the standard deviation of the time required by a slave user to do steps 3 and 4 in Figure 6 (top) and found it to be as high as 100ms. The difference between the response times we measured in the centralized and hybrid architectures (1676.9ms – 1573.5ms) was small given this large standard deviation. Thus, we consider the centralized and hybrid architecture response times as constant and interpret them to be the lowest reported value of 1573.5ms.

We also found the standard deviation of the time required by a master user to take the alternative step 2 in Figure 6 (bottom) to be at most 10.7ms. As this number is small, we can say that the steady increase we see in the response time of the replicated architecture is the result of increase in number of users.

Based on these arguments, we can say that our results show that the replicated architecture gives significantly better response times than the centralized and hybrid architectures. Using the projection method from the previous PowerPoint result, we predict that with as many as 979 users, the replicated architecture gives significantly better results, while with 1051 or more users, the centralized and hybrid architectures will give significantly better results. Interestingly, this and the previous result combined show that the type of PowerPoint presentation can decide which architecture gives the best response times. For presentations dominated by animations, the centralized architecture may be the most appropriate, while for presentations with many slides and few animations, the replicated architecture may be the most appropriate even with close to a 1000 users. Even more interestingly, our model predicts that with large presentations involving thousands of employees, the centralized and hybrid architectures will give significantly better response times.

4.3 Intra-Cluster Asymmetry

Consider a logic-intensive application and a hybrid architecture in which the difference between the processing powers of the slowest and fastest computers in some cluster is large. This may occur in practice, for instance, in an IM session when one user in a cluster is using a cell phone while others are using much more powerful desktop computers. Under these conditions, our model predicts that the hybrid architecture and some centralized architectures give better response times than the replicated architecture. Intuitively, the hybrid and centralized architectures perform better in this case because a fast computer processes and transmits inputs quicker than a slow computer so it may make sense to make slow computers slaves of the faster ones.

To formalize this informal argument, assume (a) the think times are large to guarantee that a program component is always ready to accept the next input command when it arrives, (b) the number of clusters in the hybrid architecture is low and does not change, (c) the network latencies between all users are low, (d) the cost of a particular output is the same for all users in all three architectures, (e) the cost of transmitting a single input command or a single output is negligible, and (f) all inputting users are slaves in the centralized and hybrid architectures. The relevant equations from our model become:

[2] Steps 2 and 3 (save slide to file and import saved file into presentation) were necessary because of the API exposed by the J-Integra tool we used to connect our Java-based framework and COM-based PowerPoint. In particular, we were not able to import a received slide directly into an ongoing presentation.

$$[\text{Rep-Cent}]\ \text{Resp}_i\text{Mast}_j - \text{Resp}_i\text{Slave}_j = (n\text{-}1)x_i^{in}/p_j + (w_i^{in}/p_j - w_i^{in}/p_m)$$

$$[\text{Rep-Hyb}]\ \text{Resp}_i\text{Mast}_j - \text{Resp}_i\text{Slave}_j = (n\text{-}1)x_i^{in}/p_j + (w_i^{in}/p_j - w_i^{in}/p_m) - (k\text{-}1)x_i^{in}/p_m$$

We can show each user will have worse response times in the replicated architecture. Since $p_m \geq p_j$, in both equations, the term $(w_i^{in}/p_j - w_i^{in}/p_m)$, which represents the difference in the times required to process the command on the master and slave computers, is greater than or equal to 0. Moreover, the time required to broadcast an input command on computer `j`, $(n\text{-}1)x_i^{in}/p_j$, in both equations, is always positive. Thus, the `[Rep-Cent]` result is always positive, that is, a centralized architecture gives better response time than the replicated architecture. Now

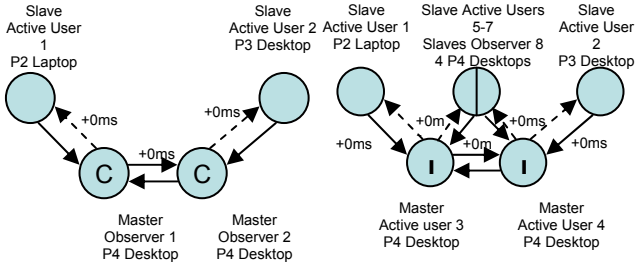

Figure 7. Checkers (left) and IM (right) hybrid architectures architectures

consider the additional term, $(k\text{-}1)x_i^{in}/p_m$, in the `[Rep-Hyb]` equation. We know that `k-1 < n-1`, that is, the number of master computers in a hybrid architecture is less than in the replicated architecture by definition. Thus, $(n\text{-}1)x_i^{in}/p_j - (k\text{-}1)x_i^{in}/p_m$ is always positive, which implies that a hybrid architecture gives better response time than the replicated architecture.

The experiments we did for this result used the hybrid architecture shown in Figure 7 (left), a centralized architecture in which the Observer 1's P4 desktop hosted the application, and a replicated architecture. We used an entire checkers log which was consistent with the assumptions above. The results are given in Table 2. As predicted, the replicated architecture gives significantly worse response time for both active users.

Table 2. P2 laptop and P3 desktop Checkers response times

Application	Centralized	Replicated	Hybrid
P2 Laptop	91.1ms	249.8ms	92.1ms
P3 Desktop	62.3ms	144ms	59.6ms

Table 3. P2 laptop and P3 desktop IM response times

Application	Centralized	Replicated	Hybrid
P2 Laptop	76.9ms	137.3ms	79.6ms
P3 Desktop	32.4ms	54.9ms	39.6ms

The experiment above used a logic-intensive application with high input processing costs. However, our prediction does not require this assumption. To verify this, we used the hybrid architecture shown in Figure 7 (right), a centralized architecture in which a P4 desktop hosted the application, and a replicated architecture. We simulated a one of our chat logs in which seven users were active. The results are shown in Table 3. The response times of the P2 laptop in the centralized and hybrid cases were significantly better than the replicated architecture response times. On the other hand, the P3 desktop response time advantage gained in the centralized and hybrid architectures for IM is not significant. Table 2 and 3 show that the advantage gained in the centralized and hybrid architectures is smaller for the chat program than for checkers. This is because this application is not logic-intensive.

Thus, we have generalized the result from previous work that low network latencies, high processing power difference, and high input processing costs together favor a centralized architecture. We show that the high input processing costs are not necessary for making this prediction. In addition, we show that these conditions also favor a hybrid architecture.

4.4 Inter-Cluster Delays

We finally make a prediction that distinguishes between hybrid and centralized architectures. A motivating scenario is groups of users in geographically-dispersed organizations chatting with each other. If the delays between the organizations is large and the master is close to one of the organizations, then users in the other organization must incur network delays they would not face in a hybrid architecture that has a master near their organization. More formally, assume that (a) the think times are large enough to not satisfy `Cond 1` so that the program component is always ready to accept the next input command when it arrives, (b) the number of clusters in the hybrid architecture is low and does not change, (c) the cost of a particular output is the same for all users in all three architectures, (d) there are multiple active users which belong to different clusters, (e) low network latencies between users in the same hybrid architecture clusters, (f) the cost of transmitting a single input command or a single output is negligible, and (g) high network latencies among users in different clusters. Under these conditions, our model predicts that the centralized architecture performance suffers for at least one active user. Assuming that $user_j$'s master computer in the centralized and hybrid architectures are `m` and `m'`, respectively, the relevant difference equations from our model are

$$[\text{Cent-Hyb}]\ \text{Resp}_i\text{Slave}_j - \text{Resp}_i\text{Slave}_j = (d(j,m)_i^{in} + d(m,j)_i^{out}) - (d(j,m')_i^{in} + d(m',j)_i^{out}) - (k\text{-}1)x_i^{in}/p_{m'} + (w_i^{in}/p_m - w_i^{in}/p_{m'})$$

$$[\text{Cent-Rep}]\ \text{Resp}_i\text{Slave}_j - \text{Resp}_i\text{Mast}_j = d(j,m)_i^{in} + d(m,j)_i^{out} - (n\text{-}1)x_i^{in}/p_m + (w_i^{in}/p_m - w_i^{in}/p_j)$$

In the `[Cent-Hyb]` difference equation, if the master computers `m` and `m'` have similar processing powers and are powerful, all the terms in the equation approach 0 except the network latency terms. For the `[Cent-Rep]` difference equation, if $user_j$'s computer and master computer `m` are in different clusters, the round-trip term dominates the equation.

Table 4. P2 laptop and P3 desktop response times

Computer	Centralized	Replicated	Hybrid
P2 Laptop	74.4ms	137.9ms	83.8ms
P3 Desktop	151.2ms	51.5ms	40.0ms

We simulated one of our chat logs in which there were seven active users. We used hybrid architecture shown in Figure 7 (right), a centralized architecture in which a P4 desktop hosted the

application, and a replicated architecture. In addition, we added 72ms to the LAN delays between clusters. Our experiment results are displayed in Table 4 and confirm our analysis. In the hybrid architecture, the P2 laptop was in one cluster while the computer used as the master in the centralized architecture was in a different cluster. As we can see, P3 user's response time was 151.2ms in the centralized architecture, while it was 40.0ms in the hybrid and 51.5ms in the replicated architecture. Meanwhile, because the P2 laptop was in the same cluster as its master, the round-trip times were just the LAN delays, so centralized and hybrid architectures performed better than the replicated architecture because the transmission costs were high.

4.5 Other Results

Our model and experiments make several additional predictions, which we do not have space to prove here:

- Inter-cluster asymmetry: A high difference in the processing powers of the most powerful computers in each cluster favors a centralized architecture.
- Intra-cluster delays: High latencies among users in a cluster favors a replicated architecture.
- High think-time impact: In all of the equations we created for realistic scenarios shown above, think times were not a factor. This is because our model predicts that after they cross a certain threshold, their values do not affect the response time. This result can be practically used in log replays by substituting actual think times with the threshold values, which would be necessary if we were replaying longer logs.
- Low think-time impact: Our model and experiments also show that there are realistic scenarios in which think times can be low enough to significantly impact the response time.

5. CONCLUSIONS AND FUTURE WORK

This paper systematically analyzes collaboration architectures to determine their response times. More specifically, it is the first to:

- bring out unresolved issues in the design of centralized, replicated and hybrid architectures that must be addressed in order to analyze their response time.
- give response-time equations for the three architectures in terms of several system and task parameters that can significantly impact the response times of the three architectures in realistic collaboration scenarios. These include processing power, network delays, processing and transmission costs, number of total users, and think times.
- make predictions about the relative performance of the three architectures under several realistic collaboration scenarios, including that 1) a small number of users, high intra-cluster network delays, and large output processing and transmission costs favor the replicated architecture, 2) large input size favors the centralized architecture, 3) high inter-cluster network delays favor the hybrid architecture, and 4) high input processing and transmission costs, low think times, asymmetric processing powers, and logic-intensive applications favor both the centralized and hybrid architectures.
- address logic-centric, data-centric, and stateless shared components in both the formal model and experiments.
- show that there are cases in which incremental graphic operations such as telepointer movements can indeed be supported with good response times, despite the fact that some commercial systems do not support them because of performance fears.

A secondary contribution is the set of logs we have gathered, which we will make public for use in future benchmarks for evaluating the performance of collaboration architectures.

We have identified only some of the parameters on which response times depend. It is important to analyze other factors such as scheduling policies and cost of displaying output. Further work is also necessary to study performance metrics other than response times such as feedthrough and task completion times [7], jitter [6], and power consumption (for mobile devices). It would also be useful to evaluate the performance of collaborations involving collaborative applications other than chat, checkers, and presentation tools such as whiteboards. Finally, future work is necessary to automatically change architectures based on the current values of collaboration parameters.

6. ACKNOWLEDGEMENTS

This research was funded in part by *Microsoft* and NSF grants ANI 0229998, EIA 03-03590, and IIS 0312328.

7. REFERENCES

[1] Ahuja, S., Ensor, J.R., Lucco, S.E. A Comparison of Application Sharing Mechanisms in Real-time Desktop Conferencing Systems. *Proc OIS 1990*. 238-248.

[2] Begole, J., Smith, R.B, Struble, C.A., and Shaffer, C.A. Resource sharing for replicated synchronous groupware. *ACM TON, 9, 6, (Dec 2001)*, 833-843.

[3] Chung, G., and Dewan. P. Towards Dynamic Collaboration Architectures. *Proc CSCW 2004*. 1-10.

[4] Correa, C. D. and Marsic, I. Software Framework for Managing Heterogeneity in Mobile Collaborative Systems. *CSCW, 14, 5-6* (2004), 603-638.

[5] Dewan, P. Architectures for Collaborative Applications. *Trends in Software: Computer Supported Cooperative Work*. 165-194.

[6] Dyck, J., Gutwin, C. Subramanian, S., Fedak, C. High-performance Telepointers. *Proc CSCW 2004*. 172-181.

[7] Junuzovic, S., Chung, G. and Dewan P. Formally Analyzing Two-user Centralized and Replicated Architectures. *ECSCW 2005*. 83-102.

[8] Litiu, R. and Prakash, A. Developing Adaptive Groupware Applications Using a Mobile Component Framework. *Proc CSCW 2000*. 107-116.

[9] Marsic, I. DISCIPLE: A framework for multimodal collaboration in heterogeneous environments. ACM Computing Surveys. 31(2es): 4 (1999).

[10] Shneiderman, B. Response Time and Display Rate. *Designing the User Interface: Strategies for Effective Human-Computer Interaction. 4th edition. Addison-Wesley Longman*. 352-369.

[11] Sun, C. and Ellis, C. Operational Transformation in Real-time Group Editors: Issues, Algorithms, and Achievements. *Proc CSCW 1998*. 59-68.