Robin Wolff
r.wolff@salford.ac.uk
The Centre for Virtual Environments
Business House
University Road
University of Salford
Salford
Manchester M5 4WT
United Kingdom

David J. Roberts
d.j.roberts@salford.ac.uk
The Centre for Virtual Environments
Business House
University Road
University of Salford
Salford
Manchester M5 4WT
United Kingdom

Oliver Otto
o.otto@salford.ac.uk
The Centre for Virtual Environments
Business House
University Road
University of Salford
Salford
Manchester M5 4WT
United Kingdom

A Study of Event Traffic During the Shared Manipulation of Objects Within a Collaborative Virtual Environment

Abstract

Event management must balance consistency and responsiveness above the requirements of shared object interaction within a Collaborative Virtual Environment (CVE) system. An understanding of the event traffic during collaborative tasks helps in the design of all aspects of a CVE system. The application, user activity, the display interface, and the network resources, all play a part in determining the characteristics of event management.

Linked cubic displays lend themselves well to supporting natural social human communication between remote users. To allow users to communicate naturally and subconsciously, continuous and detailed tracking is necessary. This, however, is hard to balance with the real-time consistency constraints of general shared object interaction.

This paper aims to explain these issues through a detailed examination of event traffic produced by a typical CVE, using both immersive and desktop displays, while supporting a variety of collaborative activities. We analyze event traffic during a highly collaborative task requiring various forms of shared object manipulation, including the concurrent manipulation of a shared object. Event sources are categorized and the influence of the form of object sharing as well as the display device interface are detailed. With the presented findings the paper wishes to aid the design of future systems.

1 Introduction

Collaborative Virtual Environments (CVE) allow people to share experience through a set of common virtual objects, regardless of geographical separation. Established applications include visualization, training, planning, and entertainment. Various forms of natural social human communication may be supported through avatars (virtual objects that represent a user), which may move around the environment, while users look at, point to, discuss, and manipulate the common objects. The CVE system is responsible for supporting such forms of communication between remote users. The distribution of a virtual environment is typically done through a shared, networked database of objects. The network introduces latencies and consistency issues which must

Presence, *Vol. 13, No. 3, June 2004,* 251–262

be addressed by the CVE system. Systems adopt a variety of distribution patterns. Centralized databases offer simple consistency control but are more affected by latency than are replicated databases, which allow users to interact with local copies of objects before communicating state changes as events. During event distribution and processing a number of mechanisms are needed for managing the balance of consistency and responsiveness. A detailed discussion would go beyond the scope of this paper and we refer the reader to Singhal and Zyda (1999) and Roberts (2004).

Interaction-rich applications may produce a scale of events that overwhelms the event management of a CVE system, reducing the quality of consistency and responsiveness. Users interface to the CVE through display systems, the capabilities of which determine the kinds of human communication that can be supported. Cubic Immersive Projection Technology (IPT), such as a CAVE, lends itself well to social human communication as the user's own body is placed within the spatial social context. This encourages natural, subconscious, nonverbal communication. Although it is trivial to capture such body movement using tracking systems, the resultant scale of event traffic can be hard to communicate across the CVE. Many teamwork tasks in the real world require the shared manipulation of objects, both sequentially and concurrently. Although many systems allow the manipulation of objects to be shared sequentially, few allow concurrent manipulation. This is because doing so in a natural manner requires complex consistency balancing to address the problem of network latency. Designing such consistency control requires a thorough understanding of event traffic.

This paper aims to increase the understanding of the level and proportions of event frequency and the effect of both the method of object sharing and the display device. In contrast to related research, we focus on cooperative manipulation as a form of closely coupled collaboration by supporting concurrent manipulation of shared objects via both the same and distinct attributes. We examine event traffic during collaboration through linked cubic IPT and desktop display devices while users share objects in a variety of ways. The main sources of events are identified and the frequency of communicated events is measured.

1.1 Related Work

Much research has been dedicated to the development of CVE systems and toolkits, including the management and optimization of network usage. Relevant examples are NPSNET (Macedonia, Zyda, Pratt, Barham, & Zeswitz, 1994), PaRADE (Roberts, Sharkey, & Sandoz, 1995), DIVE (Frécon & Stenius, 1998), MASSIVE (Greenhalgh, Purbrick, & Snowdon, 2000), CAVERNsoft (Leigh, Johnson, & DeFanti, 1997) and QUICK (Capps, 2000). Some studies have investigated the performance of communication infrastructures used to link training simulators (Wuerfel, 1998; d'Ausbourg, Bussenot, & Siron 2002), while other studies have investigated the network traffic generated by both social gatherings and teamwork activities (Roberts, Richardson, Sharkey, & Lake, 1998; Roberts, Strassner, Worthington, & Sharkey, 1999; Frécon, Smith, Steed, Stenius, & Stahl, 2001; Greenhalgh, Bullock, Frécon, Lloyd, & Steed, 2001; Leigh et al., 2001).

Although some research addresses issues of the cooperative manipulation of shared objects and the impact of the network on user performance (e.g., Ryan & Sharkey, 1998; Margery, Arnaldi, & Plouzeau, 1999; Park & Kenyon, 1999; Broll, Meier, & Schardt, 2000; Mortensen et al., 2002; Pinho, Bowman, & Freitas, 2002; Linebarger, Janneck, & Kessler, 2003), little is known about the impact of shared object manipulation through various display devices on the characteristics of the network and event handling.

2 Experimentation

The strategy for our experimentation included the measurement of event traffic during a highly collaborative application, on top of an existing CVE system, using distinct display devices. The following section describes the details of the display devices, the application and its implementation, as well as the measurements and results.

2.1 Display Devices

Our study is based on measurements of event traffic generated by two distinct display devices: an immersive cubic display and a nonimmersive desktop display. An immersive cubic display uses IPT to surround the user with projection screens showing stereo images. Common setups consist of three walls and a floor, 3 by 3 meters each. Interaction within the virtual world is achieved by motion-tracking of a user's head and hand movement. The important difference to other immersive displays, such as head-mounted displays (HMD), is that the user is still aware of his or her own body while completely immersed within computer graphics. This encourages subconscious, nonverbal social communication and allows a natural interaction within virtual environments.

The most common nonimmersive display is a desktop workstation, as it is cheap and easily available. In our configuration, a desktop user interacted on a conventional PC through a mono screen using the keyboard and mouse. Both displays were equipped with an audio connection to support verbal communication during collaboration. Throughout the measurements, the display devices were interconnected via an Ethernet LAN.

2.2 Application

Scenarios of shared object manipulation can be found in many application areas. In order to examine the event traffic during various distinct scenarios of shared object manipulation, we have designed a simple application that simulates a construction site. This application has already been a benchmark for related research, including social human communication, (Otto & Roberts, 2003; Roberts, Wolff, & Otto, 2003). The application situated remote users into a virtual construction site surrounded by a selection of building materials and tools. Their task was to build a predefined simple structure, requiring both systematic usage of tools and collaboration with remote users to gather and join materials in a number of structured subtasks. The simulation of gravity within the virtual world enforced concurrent object sharing and teamwork as in the real world. For example, a material must be held in place while it is fixed, or two users must carry large materials together.

Throughout the experiment, a user's task required both sequential and concurrent manipulation of shared objects to achieve the goal. Within the concurrent manipulation of shared objects, we can identify two major scenarios: that of distinct object attributes, and that of the same object attribute. The concurrent manipulation of distinct attributes is, for example, involved when joining building materials together. Here, one person controls the position attribute to hold a material in place, while another controls the "fixed" attribute that releases it from the effects of gravity. The concurrent manipulation of the same attribute is involved when users are carrying artificially heavy building materials, controlling the material's position attribute together. Finally, a limited number of tools in the virtual world enforced competition and sequential sharing.

2.3 Implementation

The application was implemented above a well-known CVE, DIVE (Frécon & Stenius, 1998). This was chosen because it is an established testbed for experimentation of collaboration in virtual environments and, after three major revisions, remains an effective benchmark (Frécon et al., 2001; Greenhalgh et al., 2001; Schroeder et al., 2001; Steed, Mortensen, & Frécon, 2001; Mortensen et al., 2002). DIVE version 3.3.5 was used for all display devices.

Each remote user was represented by an avatar embodied by a human-like character. A static avatar represented the nonimmersed desktop user, whereas the immersed user's avatar represented dynamic head and arm articulation, controlled through head- and hand-tracking of the cubic display device. As a construction simulation, our test application included various interactive objects that imitated the behavior of building materials and tools. Each object's behavior has been implemented by DIVE/Tcl scripts that describe a set of procedures to change an object's state in a specific way. For instance, a screwdriver tool object would make a screw object and all its intersecting material objects unmovable and thus changed to a "fixed" state. All behavior scripts were re-

active and triggered by occurrences of specific events. DIVE supports several event types. These included object transformations, such as movement or rotation; object interactions, such as grasp, release, or select; object collisions; and changes to object-specific properties and flags. Most functionality in our application was triggered by collisions of material and tool objects. For example, when a drill tool collided with a material object, the resulting collision event would trigger a procedure in the material's behavior script to increase a specific counter attribute.

The distribution of events between remote sites was handled by DIVE's distribution layer. The application can cope with the loss of certain event types. For example, users may repeat their action if an intended collision did not show the expected effect. Some specific events, however, are vital within the application. If, for example, an event that signals a change from an unfixed to a fixed state gets lost during distribution to remote sites, the object would become held in place at one site but not at the other. DIVE implements the replicated database approach above peer-to-peer communication. An event occurs first locally and is then distributed via multicast to remote replicas. The DIVE system provides a loose level of consistency, where it is assumed that objects are likely to be modified often, and therefore uses reliable multicast, relying on eventually converging states of replicas.

We have provided additional, tighter consistency management, implemented within the application-level object behavior scripts. To acknowledge the successful action of a tool, a flag is set to signal a definite state. A level of causal ordering and discarding of events was realized by constraining the order of manipulations through flags that can only be set in a certain order. For example, the flag to fix construction materials could only be set after the collision event from the drill tool and a screw occurred. If such a condition was not fulfilled the action would not be successful and the current event discarded, forcing the user to repeat the step. Allowing script procedures to set several distinct properties concurrently has enabled concurrent object manipulation of distinct attributes, avoiding exclusive object ownership. For instance, a counter attribute of an object could be set while its position was continuously updated. In contrast, concurrent manipulation of the same attribute was realized through intermediate procedures that gather and "combine" events before setting the actual attributes of the manipulated object. For example, carrying construction materials together with a remote user would be performed with the help of special tools used on each end of the material to attract a transformation. These tools send their current position to the carried object, which in turn attempts to find an average transform between them, and finally, communicates this opinion to the peer(s). One can see that this application-level consistency management adds additional events to the common traffic. However, the lack of support for the consistent shared manipulation of objects, as in many CVE systems, makes this necessary.

2.4 Measurements

To make a clear assessment of the event traffic, we extended the DIVE distribution layer for synchronized, wall-clock timed event logging. Our extension logs occurrences of outgoing event messages that have been sent over the network and maps them to a global time stamp through synchronized local clocks. The measured traffic shows the result of communicated events per second, which in turn are a result of local state changes at each site.

2.4.1 Event Types. Our measurements recorded only those events directly involved in the construction task. These included state changes of building materials, and tool and avatar objects caused by both the user and, indirectly, the object behavior scripts. Table 1 lists the various event types of DIVE that have been monitored.

Avatar movement can represent the activity and intent of the remote user in various ways. Within our application, avatar movement actions that hold particular information include change of attention, locomotion to a place, and reaching for an object or moving a hand that results in object manipulation, as well as gestures of the avatar as a form of social human communication during the collaboration. Event types in DIVE that mapped to

Table 1. *Summary of Monitored Event Types*

Category	Description	Source	Event type in DIVE
Avatar	Translation and rotation of avatar objects	User interface	Transform, Collision
Shared object	Grasping, releasing, selecting, translation, and rotation of shared objects	User interface and object behavior scripts	Transform, Interaction, Collision
Control	Application logic and application level consistency control	Object behavior scripts	Property, Flag

avatar movement were transform events caused by avatar objects and the resulting collision events.

Object manipulation included events that showed a direct effect on the shared objects caused by both the user directly and the object behavior scripts indirectly. This included positioning the materials and tools, as well as simple interactions, such as picking and selecting. Moving and rotating of shared objects generated transform events, whereas grasping, releasing, and selecting an object generated interaction events in DIVE. Again, collision events were generated when shared objects collided during transformation.

There are a number of distinctions that must be noted here between the display types. An immersed user must physically reach for an object before grasping it, whereas a desktop user can pick up even a distant object through the pointer and keyboard. Grasping an object in the cubic display results in a hierarchy change in the scenegraph where the object becomes a child of the avatar's hand. This means that manipulation does not change the relative transformation attribute of the object, as this is already mapped through hand movement events of the avatar. By default, selection in the desktop does not change the hierarchy and allows the user to directly affect the object's relative transform attribute. Thus, object manipulation through the cubic display will be communicated primarily by events describing avatar movement, while object manipulation on the desktop will be communicated primarily by those events describing object transforms.

Control comprises the application logic and application-level consistency management implemented in the object behavior scripts. Events generated by the behavior scripts included property and flag event types. Property events come from user-defined, object-specific information changes, such as counters, object references, or the level of sharing. Such events would also be generated when an object's behavior script accesses and modifies attributes within other objects, such as setting the alignment position of the target object during concurrent manipulation. Flag events were usually generated at the end of a manipulation action, signaling a definite state.

The system checks for collisions of moving objects and generates an event describing the result of a collision for each affected object. Any object within the environment, including the user's avatar, may cause collision events. As mentioned above, most functionality of our application was manged by collision triggers of certain material and tool objects. The behavior scripts must listen to and process all occurring collision events to find out whether they are relevant or not. In order to assess the load of the scripts we monitored all collision event occurrences.

In our application, vital events, such as grasping, fixing, and dropping, bound manipulation. Although these do not need to be tightly synchronized between the sites, a minimum level of causal ordering must be maintained. For example, a delayed transform event must not be allowed to affect a material once fixed. We were especially interested in the proportion of vital events to nonvital. Therefore, we monitored selected

Table 2. *Summary of Subtasks Involving Concurrent Object Manipulation*

Subtask	Description	Time stamp	Manipulated attributes	Complexity of manipulation	
				Cubic display	Desktop
ST1	Carry obj1	0.5–1.5	Same	Simple	Simple
ST2	Fix obj1	1.5–3.0	Distinct	Comprehensive	Simple
ST3	Fix obj2	3.0–4.5	Distinct	Simple	Comprehensive

event types that have been identified as vital separately from the others. The set of vital events includes most flag events, as well as some property and collision events in DIVE.

2.4.2 Timing of Subtasks. During the measurements, two remote users performed the whole construction task, structured into subtasks, for several test runs. The task included fetching tools and materials, carrying materials together, and holding materials in place while they are fixed. Within these subtasks, shared objects had to be manipulated sequentially as well as concurrently. Table 2 shows a summary of all subtasks that involved concurrent object manipulation. The table includes the approximate time stamp when a subtask was performed and how it was shared between the two users. Simple complexity of manipulation refers just to controlling the position attribute of a concurrently shared object during carrying or holding it in place, whereas comprehensive complexity refers to a number of separate manipulations of various tools and materials in a logical order to finally fix the object.

3 Results and Observations

Each test run completed construction in around four to five minutes. The process of the subtasks varied considerably over all test runs. This made a statistical analysis very difficult. However, both users always contributed with similar activity. Ten test runs were recoded. In this section, we compare the results of one typical test run. We start with analyzing the frequency of the typed event generated on each display. This is followed by a summary of the proportions of event types over the entire test run and then an overview of the influence of asymmetric displays. Throughout all tests, the average bandwidth used by the CVE system for two collaborating users within the test configuration was around 4 KBytes/s, with occasional peaks up to 20 Kbytes/s, exclusive audio streams.

3.1 Event Frequency

Figure 1 shows the measured frequency of events sent from each site. It shows graphs of events generated locally by the immersive cubic display and the nonimmersive desktop display. The values are sorted by event types, similar to Table 1, and are mapped to a common time stamp illustrating the task progressing over time in minutes. As stated in Table 2, approximately the first third of the task included concurrent manipulation of the same attribute, whereas the middle and the last third involved concurrent manipulation of distinct attributes of shared objects. Sequential sharing took place in between these subtasks.

We now closely examine the results shown in Figure 1. Graphs A and B show transform event occurrences triggered by avatar movement, while Graphs C and D illustrate the frequency of transform and interaction events caused by direct object manipulations through the users or indirectly through the object's behavior scripts. Graphs E and F display occurrences of events that were necessary for consistency control, managed by the object behavior scripts. Graphs G and H show all collision event occurrences, and finally, Graphs I and J

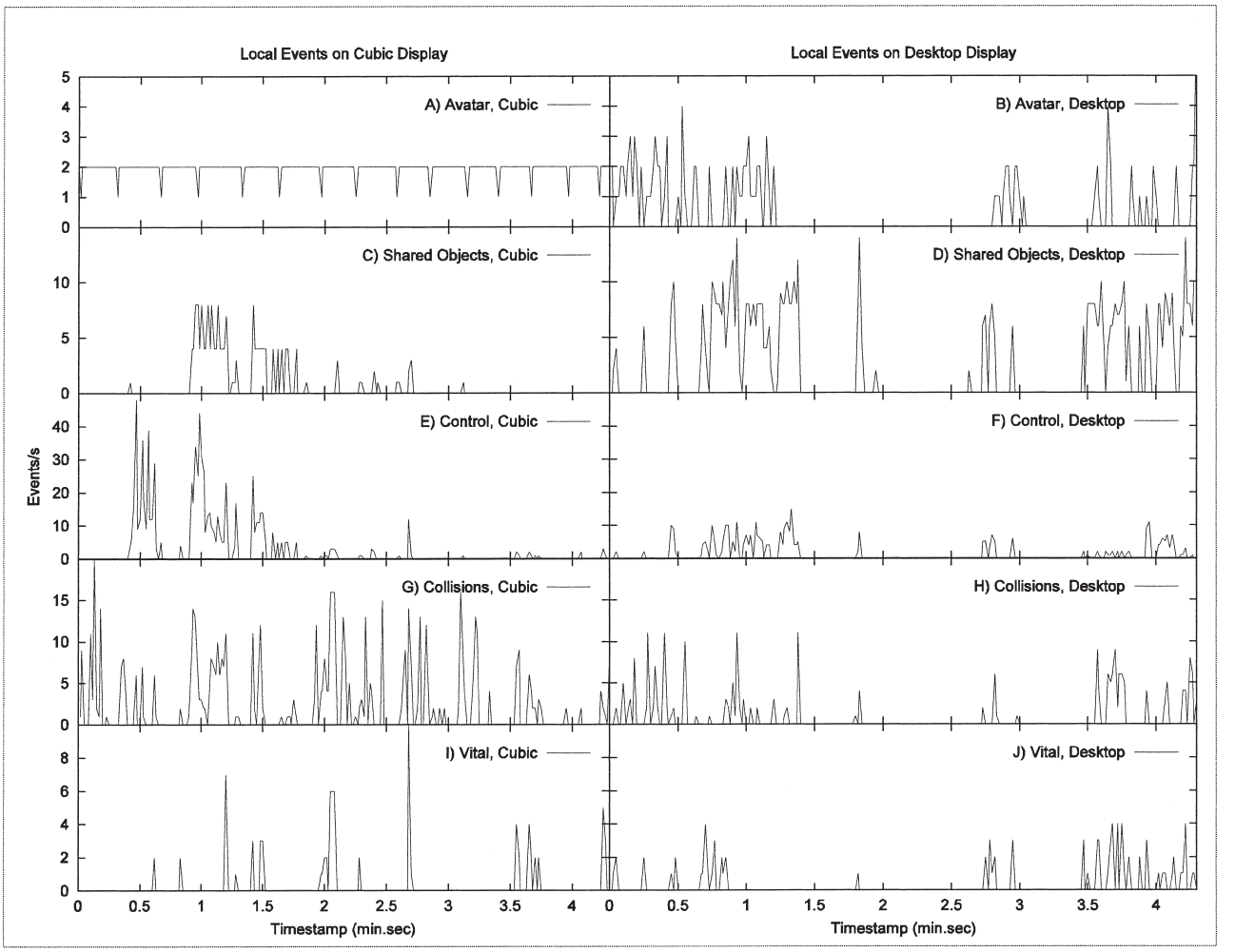

Figure 1. *Transferred events within one collaborative session.*

show the frequency of events that have been vital to the application.

In the beginning phase, between time stamp 0 and 0.5, one can see avatar movement and collision event occurrences generated by both display devices. Graph A illustrates how the tracking system of the cubic display creates events by constantly sampling the location of the user's head and hand. In our tests, the tracker update rate was configured with 500 ms, which was found to provide a reasonable compromise between accuracy and performance. All other graphs show bursts of events as a result of human interaction, similar to those found in Macedonia et al. (1994). Collisions occur when the avatar approaches the tool and material objects. The dynamic avatar of the user in immersive cubic display causes collisions in higher frequencies than the desktop user's rigid avatar. At time stamp 0.1 and 0.25, the desktop user manipulated an object, as event occurrences in graphs D, F, and J reveal. In this form, as sequential object sharing, the behavior scripts generated few events on the desktop.

At time stamp 0.4, one can see an interaction event in Graph C as the immersive user picked up a tool for concurrent manipulation and initiated subtask 1. Graph E shows frequent events with high peaks. These events are a result of the behavior scripts controlling the concur-

rent manipulation. However, an actual manipulation of the shared object has not been measured at this point in Graph C. Although both users interacted with distinct objects, no shared interaction occurred yet. The onset of concurrent manipulation of a shared object begins and is synchronized by vital events at time stamp 0.7. This is followed with a burst in nonvital events (Graphs C, D, E, and F), terminating with vital events at time stamp 1.5 when concurrent manipulation ends. Again high peaks are observed in the frequency of events for object behavior at the cubic display. One can clearly see a difference in the frequency of events for consistency management when comparing Graph E with F. The immersive display device causes the behavior scripts to generate events much more often than the desktop display.

Remember that, through motion tracking in the cubic display, no events are generated from direct transformations by the user, as these are mapped to hand movements from the dynamic avatar. What we see in Graph C is the result of transformations by the behavior scripts, whereas Graph D shows the sum of transformations by the user and behavior scripts. Both graphs show occasional peaks, rather than a smooth update rate. The users who observed a jerky movement of the material object when carrying it could confirm this. This is a result of event queuing caused by the preceding high event frequencies that must be processed by the behavior scripts before they can update the transformation of the manipulated object.

During the concurrent manipulation of distinct attributes in subtask 2 between time stamps 2.0 and 2.8, the graphs of the desktop display device show a long gap. This is because the desktop user can hold an object in the air from a distance, taking a share of the weight to allow the other user in the cubic display to manipulate it. Thus, although the desktop user does not move the mouse, all the event traffic associated with moving the object is generated from the cubic display. After the occurrence of vital events that signalized a fixed-state change of the concurrently manipulated object at 2.7 in Graph I, one can see the response of the desktop user at time stamp 2.8. The frequent events in Graph D reveal that the shared object was still manipulated before the avatar moved away, although the event for fixing should have taken effect. A delayed evaluation of vital events can lead to an erroneous application.

In the graphs of the desktop, one can see increased occurrences of events during more complex object manipulations within subtask 3. Vital event occurrences are dense at this time. Closer inspection of the log files revealed that the desktop user generated a high number of interaction events. This can be attributed to the user trying to precisely position objects within the scene, which resulted in several grasps and releases of an object. The concurrent manipulation of distinct attributes produces clearly fewer events for consistency management than manipulating the same attribute, as the object behavior graphs show.

In summary, the results show high event peaks for application-level consistency management during concurrent sharing of the same attribute. The highest frequency bursts were observed in those events generated by the behavior scripts for application control, particularly at the immersive cubic display, as shown in E. Our application tried to filter similar, superseded transform events based on thresholds for time and coordinate differences. However, the high input rate through tracking in the immersive display still resulted in event queuing. Additionally, many collision events were communicated. Only some contributed to, and few were necessary for, collaboration. Most collisions seem to occur unintentionally, often from avatars "brushing past" materials and each other during locomotion. This was partly due to the crowded nature of the environment. Although of no relevance to our application, such collisions still add to the overall event traffic within the CVE. Graphs G and H uncover that the immersive display generated collision events with higher frequency, and more densely, than the desktop display. It is interesting to see how the immersive display generates collisions during simple object manipulations during subtask 3, while the nonimmersive desktop display does not during subtask 2. Although the user in the cubic display was not navigating but just holding a material in place, the system still generated collisions by small natural head and hand movements, whereas the desktop user could rest his or her hand on a table during simple manipulations. How-

Table 3. *Event Proportions on Total Traffic*

Category	% Cubic	% Desktop	Avg. ratio cubic/ desktop
Avatar movement	13.8	3.6	4:1
Object manipulation	4.9	18.0	1:4
Object behaviour	21.3	9.8	2:1
Collision	17.3	6.4	3:1
Vital	2.5	2.3	1:1

ever, these small movements can be useful for creating a feeling of copresence and in helping to demonstrate emotion.

Peaks of vital events sometimes occur when states are changed for several objects at the same time, such as setting the fixed state for all material objects that intersect with a screw object. Vital events always occur during or around bursts of object behavior scripts, object manipulation, or collision events. Such bursts in the load for the event management can lead to a delayed evaluation of vital events and to erroneous application.

3.2 Proportions

Table 3 summarizes the proportions of the various event occurrences that had to be sent over the network. The values, given as percentages, indicate the contribution of the distinct display devices to the whole event traffic, averaged over all test runs. In summary, the immersive display device contributed 20% more to the total amount of event occurrences compared to a desktop display. Comparing the sums and ratios of avatar movement and object manipulation proportions, one can see now how continuous motion-tracking in an immersive display compensates for additional object manipulation events caused by a desktop. However, an immersive display causes considerably more collision events, as well as application-specific events for consistency control by the object behavior scripts, than the nonimmersive desktop display. The cubic display generated double the number of events than object behaviors and three times more collisions were caused. The proportions of vital events were small compared to the rest. In our highly collaborative task, only 2.5% are vital events. No significant difference in proportions of vital events can be seen between the display types, as both users showed similar activity and contribution to the collaborative task.

3.3 Comparison

Finally, we look at the effect of asymmetric displays on the total event traffic. In Figure 2, we compare the frequency of events communicated between an immersive cubic display connected to a desktop display, Graph K, and two connected cubic displays, Graph L. The former were tested across a local area network, whereas the latter were tested across the Internet, but we here assume this difference to have negligible effect on the traffic.

Comparing the frequency of event occurrences shown in the two graphs in Figure 2, one can see that in Graph L, the two immersive cubic displays together generated high peaks of up to 150 events per second, whereas in Graph K, the cubic and desktop combination showed peaks of around 60 events per second. Referring to findings of the previous results, the highest peaks occur during the concurrent manipulation of the same attribute of a shared object. During the concurrent manipulation of distinct attributes, both display configurations show similar peaks. However, the mean frequency of events, and thus the load for the CVE system, is about one third higher in Graph L than in Graph K.

4 Conclusion

This paper aimed to add to the understanding of typical event traffic in a CVE through investigating the impact of various forms of object sharing across two distinct device types. This was approached by analyzing the event traffic during a highly collaborative task, in terms of both frequency and proportions of various event types. We have presented measurements of events sent across the network during a collaborative task requiring distinct forms of object manipulation as well as

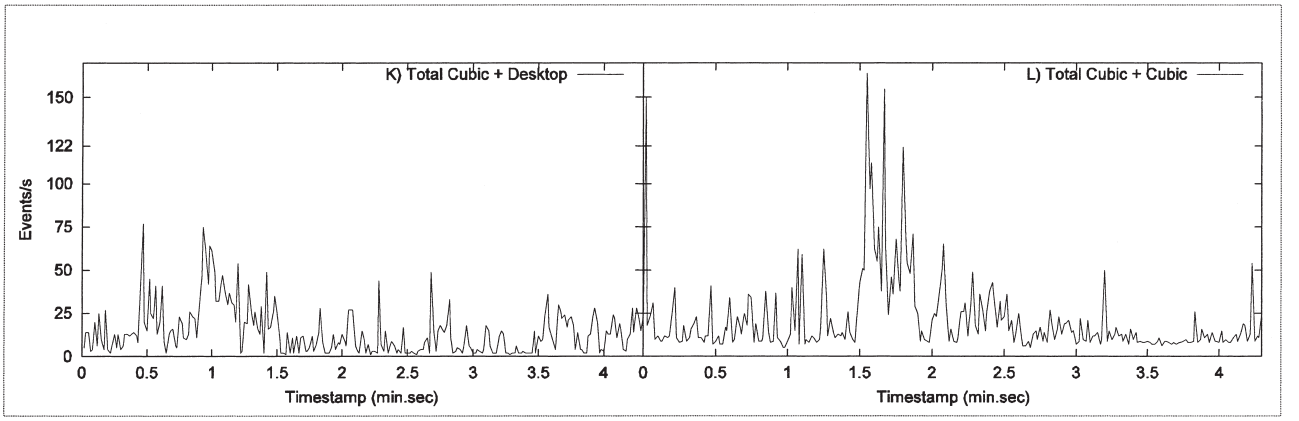

Figure 2. *Total event occurrences on different display combinations.*

nonverbal communication. Unlike previous work, this has included measurements taken during the concurrent sharing of an object, both through the same and distinct attributes. The impact on the event traffic of the type of shared object manipulation and the display device used was compared. The presented results may be of value in the design of applications, infrastructure, and input devices. For example, recognizing characteristics in collaborative actions and its resulting event traffic may lead to the development of advanced consistency control mechanisms matching the characteristics and requirements of the actual method of shared object manipulation.

Within the confines of our application and implementation, the primary findings of this work are:

1) Event bursts occur during shared object manipulation that often result in event queuing and thus the "jumping around" of the shared object.
2) The size of the burst depends on a combination of interface and type of object sharing.
3) Tight, real-time, consistency management exacerbates these bursts.
4) Erroneous results occur from the delay or loss of vital events.
5) Vital events are rare but tend to coincide or bound bursts of nonvital events.

We have shown that the interface, provided by a particular display type, has significant impact on the frequency of events. When linking several immersive displays for collaboration, the event traffic increases considerably. This should be considered when porting collaborative applications from desktop to tracked display. Furthermore, we have shown that concurrent object manipulation can result in more traffic than sequential manipulation, whereas concurrent manipulation of the same attribute has more impact than distinct attributes.

Generic support for collaboration requires both the support for various forms of object sharing and communication of natural body movements. We have shown that supporting both is achievable but nontrivial. Our results show a specific need for filtering nonvital events while maintaining some level of causal ordering for all. Strategies such as sufficient causal ordering have addressed this issue in the past (Roberts, 1995) but have not been rigorously tested with such high levels of motion-tracked data. It remains to be seen if they could offer a generic solution for supporting social human communication.

5 Future Work

We are currently developing a flexible consistency control module that supports a variety of time management strategies. Dedicated pipes distinguish between

vital and nonvital events. The time management strategies determine delivery, garbage collection, and ordering criteria.

Acknowledgments

We would like to thank Anthony Steed, and the team from the University College London (UK) as well as Christoph Anthes, Dieter Kranzlmüller, and the team from the Johannes Kepler Universität in Linz (Austria) for collaborations during this research.

References

Broll, W., Meier, E., & Schardt, T. (2000). The virtual round table—A collaborative augmented multi-user environment. *Proceedings of CVE 2000* (pp. 39–45). New York: ACM Press.

Capps, M. V. (2000). The QUICK framework for task-specific asset prioritization in Distributed Virtual Environments. *Proceedings of IEEE Virtual Reality 2000,* 143–150.

d'Ausbourg, B., Bussenot, J.-L., & Siron, P. (2002). PERFOSIM: A performance evaluation tool for HLA distributed simulations. *Proceedings of the Sixth IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT'02),* 23–31.

Frécon, E., Smith, G., Steed, A., Stenius, M., & Stahl, O. (2001). An overview of the COVEN platform. *Presence: Teleoperators and Virtual Environments, 10*(1), 109–127.

Frécon, E., & Stenius, M. (1998). DIVE: A scalable network architecture for distributed virtual environments. *Distributed Systems Engineering Journal (special issue on Distributed Virtual Environments), 5*(3), 91–100.

Greenhalgh, C., Bullock, A., Frécon, E., Lloyd, D., & Steed, A. (2001). Making networked virtual environments work. *Presence: Teleoperators and Virtual Environments, 10*(2), 142–159.

Greenhalgh, C., Purbrick, J., & Snowdon, D. (2000). Inside MASSIVE-3: Flexible support for data consistency and world structuring. *Proceedings of CVE 2000* (pp. 119–127). New York: ACM Press.

Leigh, J., Johnson, A. E., & DeFanti, T. A. (1997). CAVERN: A distributed architecture for supporting scalable persistence and interoperability in collaborative virtual environments. *Journal of Virtual Reality Research, Development and Applications 2*(2), 217–237.

Leigh, J., Yu, O., Schonfeld, D., Ansari, R., He, E., & Nayak, A. (2001). Adaptive networking for tele-immersion. *Proceedings of Immersive Projection Technology/Eurographics Virtual Environments Workshop (IPT/EGVE),* 199–208.

Linebarger, J. M., Janneck, C. D., & Kessler, G. D. (2003). Shared simple Virtual Environment: An object-oriented framework for highly-interactive group collaboration. *Proceedings of the Seventh IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'03),* 170–180.

Macedonia, M. R., Zyda, M. J., Pratt, D. R., Barham, P. T., & Zeswitz, S. (1994). NPSNET: A network software architecture for large-scale Virtual Environments. *Presence: Teleoperators and Virtual Environments, 3*(4), 265–287.

Margery, D., Arnaldi, B., Plouzeau, N. (1999). A general framework for cooperative manipulation in Virtual Environments. In M. Gervautz, A. Hildebrand, and D. Schmalstieg (Eds.), *Proceedings of Eurographics Workshop* (*Virtual Environments '99*) (pp. 169–178). Vienna: Springer Verlag.

Mortensen, J., Vinagayamoorthy, V., Slater, M., Steed, A., Lok, B., & Whitton, M. C. (2002, May). *Collaboration in Tele-Immersive Environments.* Paper presented at the Eighth Eurographics Workshop on Virtual Environments, Barcelona, Spain.

Otto, O., & Roberts, D. (2003). Importance of communication influences on a highly collaborative task. *Proceedings of the Seventh IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'03),* 195–201.

Park, K., & Kenyon, R. (1999). Effects of network characteristics on human performance in a collaborative virtual environment. *Proceedings of IEEE* Virtual Reality 1999, 104–111.

Pinho, M. S., Bowman, D., & Freitas, C. (2002). Cooperative object manipulation in immersive Virtual Environments: Framework and techniques. *Proceedings of ACM Symposium on Virtual Reality Software and Technology (VRST'02)* (pp. 171–178). *ACM SIGCHI & ACM SIGGRAPH.*

Roberts, D. J. (2004). Communication infrastructures for inhabited information spaces. In D. N. Snowdon, E. F. Churchill, & E. Frécon (Eds.), *Inhabited Information Spaces, Living with your Data* (pp. 233–267). Computer Supported Cooperative Work. London: Springer-Verlag.

Roberts, D. J., Richardson, A. T., Sharkey, P. M., & Lake, T. W. (1998). Optimising exchange of attribute ownership in the DMSO RTI. *Proceedings of the Spring Simulation Interoperability Workshop, SISO,* 379–386.

Roberts, D. J., Sharkey, P., & Sandoz, P. (1995). A real-time, predictive architecture for Distributed Virtual Reality. *Proceedings of the First ACM SIGGRAPH Workshop on Simulation & Interaction in Virtual Environments,* 279–288.

Roberts, D. J., Strassner, J., Worthington, B. G., & Sharkey, P. (1999). Influence of the supporting protocol on the latencies induced by concurrency control within a large scale multi user distributed virtual reality system. Paper presented at the International Conference on Virtual Worlds and Simulation (VWSIM), SCS Western Multi-conference '99, San Francisco, CA, 70–75.

Roberts, D. J., Wolff, R., & Otto, O. (2003). Constructing a gazebo: Supporting teamwork in a tightly coupled, distributed task in virtual reality. *Presence: Teleoperators and Virtual Environments, 12*(6), 644–660.

Ryan, M. D., & Sharkey, P. M. (1998). Distortion in distributed virtual reality. In J.-C. Heudin (Ed.), *Proceedings of the First International Conference on Virtual Worlds (VW'98), Lecture Notes in Artificial Intelligence,* Vol. 1434, 42–48. London: Springer-Verlag.

Schroeder, R., Steed, A., Axelsson, A., Heldal, I., Abelin, A., Widestom, J., et al. (2001). Collaborating in networked immersive spaces: As good as being there together? *Computers & Graphics 25*(5), 781–788.

Singhal S., & Zyda M. (1999). *Networked Virtual Environments: Design and Implementation*. SIGGRAPH Series, New York: ACM Press.

Steed, A., Mortensen, J., & Frécon, E. (2001). Spelunking: Experiences using the DIVE System on CAVE-like platforms. In H.-J. Bullinger (Ed.), *Immersive Projection Technologies and Virtual Environments* (pp. 153–164). Vienna: Springer-Verlag.

Wuerfel, R. D. (1998). A comparison of HLA and DIS real-time performance. *Proceedings of the Spring Simulation Interoperability Workshop '98.*