FORUM

On Consistency and Network Latency in Distributed Interactive Applications: A Survey—Part I

Abstract

This paper is the first part of a two-part paper that documents a detailed survey of the research carried out on consistency and latency in distributed interactive applications (DIAs) in recent decades. Part I reviews the terminology associated with DIAs and offers definitions for consistency and latency. Related issues such as jitter and fidelity are also discussed. Furthermore, the various consistency maintenance mechanisms that researchers have used to improve consistency and reduce latency effects are considered. These mechanisms are grouped into one of three categories, namely time management, information management, and system architectural management. This paper presents the techniques associated with the time management category. Examples of such mechanisms include time warp, lock step synchronization, and predictive time management. The remaining two categories are presented in part II of the survey.

1 Introduction

Human interaction with computers has evolved from centralized batch processing using Hollerith cards to virtual reality systems in which users can be fully immersed. The origins of virtual reality technology can be traced back to vehicle simulation research in the 1920s (Ellis, 1994) and since then, virtual environments have spread to areas such as military simulations, cooperative whiteboards, and architectural design.

The first distributed software virtual environment was SIMNET, a United States research program initiated in 1983 to train soldiers in battlefield tactics (Calvin, Dickens, Gaines, Metzger, Miller et al., 1993) and which culminated in the development of the DIS protocol standard in 1993 (IEEE, 1993). Since then numerous academic, military, and commercial distributed applications have been developed and documented (Joslin, Di Giacomo, & Magnenat-Thalmann, 2004). The diverse applications that exploit distributed virtual technology have been referred to using various terms. In this paper they will be referred to as distributed interactive applications or DIAs.

DIAs are subject to many problems. However, the authors believe that the key common objective running through most of the research in the area of DIAs is related to the issue of consistency and the maintenance of adequate consistency among all participants of the DIA in spite of demands on system resources. In addition they believe that the single greatest contributing factor to spatial and temporal inconsistencies experienced by end users in the virtual world is network latency or lag.

The issue of consistency has been the subject of much research in the field of distributed systems for the past few decades. However, no single detailed survey of these techniques has been produced, although some publications summarize a number of the issues and approaches taken (Snowdon, Greenhalgh, Benford, Bullock, & Brown, 1996; Macedonia & Zyda, 1997; Singhal & Zyda, 1999; Joslin et al., 2004; Roberts, 2004). This paper aims to bridge this gap and provide a concise summary of the issues and the approaches taken to

Declan Delaney*
Tomás Ward
Seamus McLoone
Department of Electronic Engineering
National University of Ireland, Maynooth
Maynooth, Ireland
*Correspondence to drd@eircom.net

Presence, *Vol. 15, No. 2, April 2006,* 218–234

tackle the problem of maintaining consistency in the presence of network latency. It also reviews the terminology associated with the domain of distributed interactive applications and proposes definitions for the terms consistency and latency. This survey is not intended to be comprehensive in bibliographic terms; instead references are selected to best illustrate, in our opinion, the different concepts.

The next section looks at some basic terminology. This is provided for ease of reference. Section 3 investigates what is meant by the term *distributed interactive application*. Sections 4 and 5 define and review the inherent problems of consistency and network latency in such applications. In Section 6 we provide a classification for various mechanisms that have been employed to mask network latency and improve consistency in DIAs. The first of these classifications, time management techniques, is then presented. The remaining two classifications are presented in part II of the survey.

2 Terminology

To facilitate discussion of DIAs, the following terms derived from the world of modeling and simulation are listed (Zeigler, Praehofer, & Kim, 2000). The terms listed here are used extensively in referring to DIAs, but are rarely defined. Other terms will be defined as they are encountered.

2.0.1 State. A complete description of a virtual entity at a single moment in time (Churchill, Snowdon, & Munro, 2001). In a DIA information pertaining to state variations in the virtual environment should be shared among all participants. This information is often referred to as dynamic shared state (Singhal & Zyd, 1999; Qin, 2002).

2.0.2 Entity or Object. An element of the synthetic environment that is created and controlled by a simulation application through the exchange of information (IEEE, 1993). We consider entity and object as synonymous, and we adopt the term entity throughout this paper. Entities may be *passive* or *active* (Singhal & Zyda, 1999). A passive entity is one that is either entirely stationary or moves deterministically. The behavior of an active entity is nondeterministic, although it may be predictable in the sense that its motion is a function of the constraints imposed by the environment, the objectives/abilities of the participant and the initial conditions. A similar classification describes entities as either *static* (passive) or *dynamic* (active) (Lui, 2001). The state of a static entity never changes with time whereas the state of a dynamic entity can.

2.0.3 Environment. The information needed to render an application's time-constant state (Mauve, Hilt, Kuhmunch, & Effelsberg, 2001).

2.0.4 Event. The state of a DIA may change for two reasons, as a result of the *passage of time* or the occurrence of *events*. An event causes a state change that is not a fully deterministic function of time (Mauve, 2000). Between successive events the state of the medium is a fully deterministic function of time. Events can be separated into *external events* and *internal events*. External events are caused by (user) interactions with the medium, whereas internal events are nondeterministic internal changes in the state of the application. Events may also be viewed as *deterministic* or *nondeterministic* (Roberts & Sharkey, 1997a; Sharkey, Ryan, & Roberts, 1998). Deterministic events can be fully predicted whereas nondeterministic events cannot be reliably predicted because they result from real-time interaction between the DIA and human participants.

2.0.5 Node. A node refers to a computing device connected to the communications network. A node can be a source or destination machine, a router, or any other device that processes data.

2.0.6 Dead Reckoning. Dead reckoning exploits information about current user dynamics to make a short-term prediction of user movement based on extrapolation techniques. It is formally defined in the DIS protocol (IEEE, 1993). It was first implemented as the Players and Ghosts paradigm in the VERN (Virtual Environment Realtime Network) test bed, developed by Blau, Hughes, Moshell, & Lisle (1992).

3 Distributed Interactive Application

A distributed interactive application (DIA) is best defined by considering the three terms that comprise its title:

1. A system is *distributed* if there is a message transmission delay between parts of the system that is not negligible compared to the time between events in a single process (Lamport, 1978).
2. *Interactive* refers to an input-output process involving input from a user through a human-computer-interface with an appropriate output response from the system (Broll, 1997; Natrajan & Reynolds, 1999; Manninen, 2000; Zhou, Cai, Lee, & Turner, 2001). Interaction is also defined as truly concurrent object manipulation (Margery, Arnaldi, & Plouzeau, 1999), as distinct from the cooperative sequential object manipulation defined in Broll.
3. The *application* is a software system built for a specific purpose. It is usually a virtual environment that, in some way, reflects the physical world and interfaces to the physical world.

What this paper refers to as DIAs have been referred to using many different terms, as summarized in Table 1. These reflect the diverse applications of this paradigm and the different research emphasis of each research group: shared workspaces, networked games, distributed whiteboards, group editors, distributed architectural design, education, telemedicine and simulations (Bhola, Banavar, & Ahamad, 1998; Sun, Jia, Zhang, Yang, & Chen, 1998; Bouras, Hornig, Triantafillou, & Tsiatsos, 2001; Fujimoto, 2001; Riva & Gamberini, 2001; Tawfik & Fernando, 2001; McCoy, Delaney, & Ward, 2003; Frécon, 2004). Several research teams have developed experimental distributed virtual environment platforms; some examples include RING (Funkhouser, 1995), NPSNET (Macedonia, Zyda, Pratt, Barham, & Zeswitz 1994; Capps, McGregor, Brutzman, & Zyda 2000), MASSIVE (Greenhalgh & Benford, 1995; Greenhalgh, Purbick, & Snowdon 2000), PaRADE (Roberts, Sharkey, & Sandoz, 1995; Roberts & Sharkey, 1997a, 1997b), SPLINE (Barrus, Waters, & Anderson, 1996), CAVERNSoft (Leigh, Yu, Schonfeld, Ansari, He et al., 2001), VELVET (de Oliveira & Georganas, 2002), PARADISE (Holbrook, Singhal, & Cheriton, 1995; Singhal, 1996), DIVE (Frécon & Stenius, 1998; Frécon, 2003; Frécon, 2004), QUICK (Capps, 2000), VPark (Joslin, Molet, Thalmann, Esmerado, Thalmann et al., 2001), MOVE (Garcia, Montala, Pairot, Rallo, & Skarmeta, 2002), ATLAS (Lee, Lim, & Han, 2002), EQUATOR (MacColl, Millard, Randell, & Steed, 2002) and PING (Roberts, 2004).

For the purposes of this paper we adopt the term distributed interactive application (Kelly, 1997; Diot & Gautier, 1999; Lee, Yang, Yoon, Yu, & Hyun, 2000; Zhou et al. 2001; Qin, 2002; Vogel, Mauve, Hilt, & Effelsberg, 2003) as we consider it to be a generic term that encapsulates all aspects of networked applications

Table 1. *DIAs are referred to using various terms and acronyms*

DIAs
Distributed interactive simulation (DIS)
Networked virtual environment (NetVE; Singhal & Zyda, 1999)
Distributed virtual environment (DVE; Stytz, 1996)
Distributed interactive media (DIM; Mauve et al. 2001)
Networked interactive entertainment (NIE; Capps, McDowell, & Zyda, 2001)
Collaborative virtual environment (CVE; Park & Kenyon, 1999; Vaghi, Greenhalgh, & Benford, 1999)
Distributed synthetic environments (DSE; Worthington & Roberts, 2000)
Shared virtual environment (SVE; Waters & Barrus, 1997)
Computer supported cooperative work (CSCW; Greif, 1988)
Groupware systems (Ellis & Gibbs, 1989)

that involve interaction, collaboration, cooperation and communication. We define a DIA to be:

> a networked software system that seeks to maintain global consistency when responding to multiple simultaneous nondeterministic inputs.

A key aspect of a DIA is the participation of humans-in-the-loop (Ellis & Gibbs, 1989). People interact with the system in real time and they expect the system to respond in real time to the actions they make. In addition they assume that other users of the system, particularly those with whom they are interacting, share the same view of the system state. In other words, users demand consistency. The next section addresses this issue in detail.

4 Consistency

The input from users of a DIA generates events that then modify an underlying database that is shared in some way by all users across a communications network. An essential requirement of the DIA is that users are informed of changes to this common database in real time if the changes affect them or if they are interested in receiving the changes. Users of DIAs expect both temporal and spatial consistency so that their experience of the virtual world can be the same as the users with whom they are interacting.

Consistency is one of the key factors in providing a truly interactive experience to users of DIAs (Vaghi et al. 1999; Bowman, Johnson, & Hodges, 2001; Gutwin, 2001; Henderson, 2003). So, what exactly is consistency? For Gautier, Diot, & Kurose (1999) it implies that at any point in time, all players should ideally see the same information at the same time independent of the network. This is referred to as absolute consistency and it has been defined mathematically by Qin (2002) and Zhou et al. (2001). Absolute consistency in distributed applications is impossible to attain because of the existence of network latency and the fact that the delay in transmitting information across the application cannot be ignored when compared to the time between events.

Consistency refers to a number of aspects in DIAs (Shneiderman, 1984; Dourish, 1995; Blakowski & Steinmetz, 1996; Roberts, 2004):

1. Synchronization The maintenance of (a) temporal relations between events so that the time of each event relative to other events across the DIA is the same for all participants and (b) spatial relations so that entity positions are the same across the DIA;
2. Causality or ordering Events cause the state of the system to change to a new state. The order in which events are received is thus important to maintain a natural cause-effect order;
3. Concurrency The simultaneous execution of events by different users on the same entity within the application. Entity ownership conflicts have to be resolved.

Linked to all of these aspects are the issues of responsiveness and fidelity. Responsiveness is the time taken for the system to register and respond to a user event. Fidelity is defined in the DIS standard as the degree to which the representation within a simulation is similar to a real-world object, feature, or condition in a measurable or perceivable manner (IEEE, 1995). Similarly the Fidelity Definition and Metrics Implementation Study Group (DM-ISG) defines fidelity as "the degree to which a model or simulation reproduces the state and behavior of a real-world object or perception of a real-world object, feature, condition, or standard in a measurable or perceivable manner." The DM-ISG identifies six components to fidelity: resolution, error, accuracy, sensitivity, precision, and capacity (Roza, Voogd, Jense, & van Gool, 1999). Stytz (1996) explores fidelity in DIAs and lists a number of types of fidelity, such as physics fidelity, time fidelity, and sensory fidelity. Capps (2000) discusses the link between visual accuracy and fidelity and believes that visual accuracy is only a "passing first-order approximation for fidelity." Fidelity is also referred to as *level-of-detail* or *resolution* (Radhakrishnan & Wilsey, 1999).

Poor consistency can lead to fidelity problems and fidelity is often sacrificed to maintain consistency and responsiveness. In an ideal DIA absolute consistency could be obtained: all user clocks would be synchronized, all events would be executed in the order they occurred, conflicts over shared entities and the system response time would satisfy normal user expectations, and fidelity would be guaranteed. However, this ideal cannot be achieved and the inconsistencies that result manifest themselves in many different ways, resulting in three noted phenomena (Schwarz & Mattern, 1994; Dourish, 1995; Sun & Ellis, 1998; Vaghi et al., 1999; Zhou et al., 2001; Zhou, Cai, Lee, & Turner, 2003):

1. Divergence This refers to the final temporal-spatial state of the environment being different for different participants. Users will thus unwittingly be interacting with an inconsistent environment so that their behavior will also be inconsistent, possibly causing the states to diverge even further;
2. Causality violation Events may be received, executed, and rendered out of their natural cause-effect order, so that users notice the effect before the cause;
3. Intention/expectation violation The actual effect of an action may differ from the intended effect because events may be generated concurrently. This refers to something that is expected to happen based on real-world experience, but doesn't happen in the virtual world.

Consistency therefore needs to be controlled within the DIA. This is achieved by using consistency maintenance mechanisms. In this paper these are defined as:

> any element employed to ensure a sufficient, uniform dynamic shared state for all participants in a DIA.

While absolute consistency is impossible to achieve in practical DIAs because of their dynamic nature, a hypothetical DIA in which the environment remains totally static and unchanging would achieve absolute consistency. At one extreme therefore is the dynamic DIA with continuous state changes that requires a high throughput of update packets and that never achieves absolute consistency; at the other is the totally static DIA where no update packets are required and so absolute consistency can be achieved. This is known as the *Consistency-Throughput Trade-off* (Singhal & Zyda, 1999). There is also a conflict between system responsiveness and consistency. High responsiveness can be achieved locally, but with the risk that the global state of the DIA becomes inconsistent because of network latency (Cheshire, 1996). This conflict has been documented as the consistency-responsiveness trade-off (Zhou et al., 2001; Mauve, Vogel, Hilt, & Effelsberg, 2002; Qin, 2002).

The principal cause of inconsistencies is ironically linked to the network that actually facilitates DIAs. Therefore the relationship that exists between the physical network and consistency needs to be explored. In particular we focus on the most important aspect of the physical network affecting consistency: latency. The next section describes this phenomenon.

5 Latency

A review of the research literature reveals that a commonly agreed definition of latency does not exist. Latency has sometimes been defined as the amount of time required to transfer a bit of data from one point to another (Singhal & Zyda, 1999). This definition leaves the word *point* open to several possible interpretations. Latency is also defined in Smed, Kaukoranta, & Hakonen (2001) as the length of time (or delay) that a message incurs during transmission from one designated node to another. Again, the definition is open to different interpretations of the word *node*. A similar problem exists in the following definition: the time taken by data to travel from the source to the destination (Dutta-Roy, 2000). Meehan et al. incorporate the delay associated with graphics rendering in their definition (Meehan, Razzzaque, Whitton, & Brooks, 2003). Tse-Au and Morreale interpret latency as the time required for one bit to propagate round-trip between two nodes, similar to a *ping* time (Tse-Au & Morreale, 2000). Pullen and Wood define it as the time delay that occurs between the output of a data packet at the application level of one simulator and the input of that data packet at the application level of another simulator (Pullen & Wood, 1995). We have taken this definition as a basis to propose the following definition for network latency based on the ISO OSI reference model:

> Network Latency is the time taken from the start of exchange of an application protocol data unit (APDU) at the application layer of one participating node to the end of the exchange of the same APDU with the application layer of a second participating node.

This definition is similar to that put forward by de Oliveira, Shirmohammadi, and Georganas (1999) and excludes latency due to any other source, for example, computations in the application, graphics rendering, or user reaction time. These other sources are also included in local latency, which is the latency involved in representing a user's actions back to that user (Roberts et al., 1995). For the rest of the paper the term latency will refer to network latency.

Latency is particularly problematic in interactive applications where the network delays are comparable to the interaction time or speed (Sharkey et al., 1998). Typical latency values to maintain real-time interaction fluctuate between 40 and 300 ms (Diot & Gautier, 1999). Cheshire (1996) suggests that the round-trip latency of 100 ms should be the maximum delay, whereas both Wloka (1995) and Diot and Gautier aim for a one-way delay of 100 ms. Conscious awareness of latency by users of a DIA depends on the application. For voice interaction a maximum latency of 100 ms is sufficient (Cheshire, 1996) and Mauve et al. report values of 120 ms for visual data (Mauve et al., 2002). In teleoperation control loops, instability occurs when total system latency exceeds 100 ms. For motor-driven tasks human reaction time is on the order of 200 ms (Krumm-Heller & Taylor, 2000).

5.1 Components of Latency

To assist in the understanding of latency, it is convenient to analyze the factors that comprise it. The main contributors to latency are:

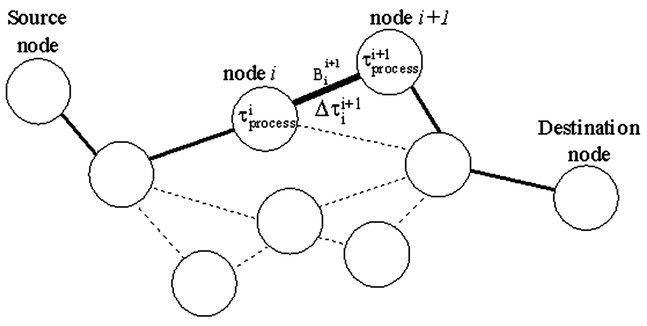

Figure 1. *Illustration of network nodes. B refers to bandwidth, τ refers to processing time and $\Delta\tau$ refers to propagation delay.*

1. Queuing and processing at routers, bridges, and gateways within the network and at source and destination nodes. Packet loss and timeouts may be considered to introduce infinite latency;
2. Transcoding delays associated with encryption/decryption of information and compression/decompression of data;
3. Propagation and transmission delays due to the finite speed of light and the speed of the communications link. (Martin, McGregor, & Cleary, 2000)

This paper decomposes latency into three types according to the source of the delay:

1. Packet processing delay;
2. Bit propagation delay;
3. Packet propagation delay.

The latency τ_{total} associated with a single packet can therefore be expressed as the sum of these three components. This is illustrated in Figure 1 and is represented by the following equation:

$$\tau_{\text{total}} = \underset{\substack{\uparrow \\ \textit{Packet} \\ \textit{Processing} \\ \textit{Delay}}}{\sum_{i=1}^{N} \tau^{i}_{\text{process}}} + \underset{\substack{\uparrow \\ \textit{Bit} \\ \textit{Propagation} \\ \textit{Delay}}}{\sum_{i=1}^{N-1} \Delta\tau_i^{i+1}} + \underset{\substack{\uparrow \\ \textit{Packet} \\ \textit{Propagation} \\ \textit{Delay}}}{\sum_{i=1}^{N-1} \frac{M}{B_i^{i+1}}} \tag{1}$$

where

$\tau^{i}_{\text{process}}$ is the time to process a packet at node i;

N is the number of nodes, including source and destination nodes;

$\Delta\tau_i^{i+1}$ is the transmission time between nodes i and $i+1$;

B_i^{i+1} is the bandwidth between nodes i and $i+1$;

M is the number of bits in the packet.

The *packet processing delay* refers to the time taken to manage and process the data as it migrates through the network hardware and to process and parse the data at both source and destination nodes. This includes compressing, decompressing, encrypting, decrypting, and any processing performed by the operating system or network hardware at the end-point computers. It also includes the time delay associated with flow control and congestion control, buffering, and packet queuing.

The *bit propagation delay* refers to the delay associated with the physical speed of transmission (about 5 μs per km).

The *packet propagation delay* refers to the time required for all bits in a packet to be transmitted across the network from source to destination node considering only the internode bandwidth.

From this division of latency the following may be noted:

1. The bit propagation delay cannot be eliminated and it has a lower theoretical limit imposed by the speed of light in a vacuum;
2. The packet processing delay can be reduced in a number of ways: by reducing the quantity of data on the network, by increasing the processing power at routers and source/destination nodes, and by using more efficient processing algorithms;
3. The packet propagation delay can be reduced by increasing the available network bandwidth and by reducing the amount of data that must be transmitted between nodes.

Points 2 and 3 explain why most of the techniques and approaches that have been explored to combat the effects of network latency focus on reducing the quantity of network traffic and improving software architecture to reduce both the packet processing delay and the packet propagation delay.

The division of latency given here facilitates a more accurate understanding of latency and the impact of latency reducing techniques and is similar to that espoused by Jehaes et al. (2003). Other divisions of latency have been presented elsewhere. Tse-Au and Morreale (2000) view latency as being tunable or untunable. *Untunable latency* refers to the sum of the signal propagation delay and the equipment processing delay in transmitting one bit across the network. *Tunable latency* is the time taken for one bit of data to propagate through all the queuing delays between the two communicating nodes. In Hecker and Simpson (2001) latency is split between *path latency* and *queuing latency*. Path latency is the time taken for a message to get from one place to another. Queuing latency is the resultant delay when the data sent exceeds the bandwidth. Two kinds of latency are also described by Mauve et al. (2004): *observation lag* and *influence lag*. Observation lag is the delay between an event's occurrence and its display, whereas influence lag is the delay between our attempt to influence the world and the time taken for the influence to actually occur.

Network latency varies with prevailing network conditions. This variation in latency results in a phenomenon called jitter, which is described below.

5.2 Jitter

All information transmitted across the network is carried in the form of data packets. Each packet may follow a different route through the network and the order of arrival of packets at the destination node is not guaranteed by the network layer. The network is heterogeneous in nature with continuous variations in router queue lengths, buffering times, and routing paths for packets. Packet sizes also vary greatly and similarly the time to process packets as they migrate through the network and reorder packets to rebuild complete packets at the destination also varies significantly (Dutta-Roy, 2000). Depending on network conditions, this may result in a different latency for each transmitted packet and gives rise to a phenomenon known as jitter (Blow, 1998; Smed et al., 2001). For the purposes of this paper the following definition will be used:

Jitter refers to the unpredictable variation in latency with time.

The effects of network latency and jitter on end-users were examined in an interesting suite of qualitative experiments performed by Vaghi et al. (1999). They examined the effects of gradually increasing network delays for one of the two participants in a networked virtual ball game to establish how delays affect the performance of the players, how performance breakdowns manifest themselves, and how players adapt to delays. Park and Kenyon (1999) assessed the effects of latency and jitter on executing a cooperative task and found that from a psycho-perceptual viewpoint, jitter has the greatest impact when latency is high and the collaborative task is difficult. Jitter has a much greater impact than latency on performance. Gutwin (2001) investigated the effects of latency and jitter on two types of user interaction: prediction of movement and moving a shared object. He found that both performance and user strategy were affected. Other work has been carried out by Henderson (2003), who investigated how users adapt to latency in distributed games, Bowman et al. (2001), who analyzed interaction in virtual environments and Mauve (2000a), who examined the effects on interaction in a 3D tele-cooperation application when local lag is introduced to offset the effects of jitter.

Latency prevents the possibility of achieving an absolute consistent dynamic shared state across a DIA. As the quantity of data to be managed by the DIA increases (an issue called scalability; Macedonia, 1995) and due, for example, to an increase in the number of active entities and participants or the spatial extent of the DIA or the complexity of the tasks involved), maintaining consistency across the DIA becomes even more difficult. The following section examines the various methods, mechanisms, and approaches for controlling consistency in DIAs in the face of this inherent network latency.

6 Consistency Maintenance Mechanisms

A DIA can be viewed as a distributed database with multiple simultaneous users modifying it in real time. The key problem then becomes that of ensuring that the database is consistent for all users, so that they are interacting with an up-to-date source of information. Consistency maintenance mechanisms refer to any element employed to ensure a sufficient, uniform dynamic shared state for all participants in a DIA. Consistency maintenance mechanisms are also referred to as latency hiding techniques (Cheshire, 1996), consistency control algorithms (Roberts & Sharkey, 1997b), distributed synchronization mechanisms (Diot & Gautier, 1999), and object synchronization (Lui, 2001). Of the many existing mechanisms, most of them are bandwidth-saving mechanisms and combat latency by reducing the number of packets being sent across the network as explained previously. Each consistency maintenance mechanism falls into one of two general categories (Jefferson, 1990; Blanchard & Lake, 1997; Fujimoto, 2001; Cronin, Filstrup, Kurc, & Jamin, 2002):

1. Optimistic/aggressive This is a mechanism that takes risks by performing speculative computation, which, if subsequently determined to be correct, saves time, but if incorrect must be rolled back and corrected;
2. Pessimistic/conservative This is a mechanism that never indulges in speculative computation and hence never has to roll back. These algorithms perform poorly in fast-paced interactive applications where a constant rate of simulation is important.

Systems can employ a hybrid approach, applying optimistic and conservative mechanisms at different points in the DIA, depending on the type of DIA. The various mechanisms may be further classified into three general classes:

1. Time management techniques These all manipulate time to mask the effects of latency. Examples include synchronization, time warp, and local perception filtering;
2. Information management techniques These all reduce the amount of data that has to be managed by the network. Examples include relevance filtering and compression;

3. System architecture techniques These all aim to improve the efficiency of processing and disseminating data. Examples include protocols and network architectures.

This rest of this paper will focus on the first of these three categories and the other two categories will be discussed in Part II of the survey. It should be noted that the techniques are not mutually exclusive and the DIA designer can mix techniques across categories to suit each particular application.

6.1 Time Management Techniques

Time is a fundamental attribute of all distributed interactive applications, although the exact meaning of time may vary from one application to another. There are two dominant concepts of time for distributed applications:

1. Absolute time or wall clock time The time is based on the concept of a periodic clock, which is synchronized to coordinated universal time (UTC) across the DIA. For example in the internet this time is maintained by a network of servers that can communicate precise UTC time using the network time protocol (NTP; Mills, 1991; Cox, Luiijf, van Kampen, & Ripley, 1996);
2. Virtual Time or Causal Time The time in this case is based on a logical, loosely synchronized clock. Time is seen as a sequence of ordered events and stands still if no new events occur (Lamport, 1978).

The link between time and consistency in DIAs is important. In a perfectly consistent DIA all participants would be rendered the same global state at the same absolute time. Network latency ensures that this cannot be achieved and inconsistencies therefore occur.

The concept of time within a DIA depends on whether the DIA is *continuous* or *discrete* (Roberts & Sharkey, 1997a, 1997b; Mauve et al., 2001; Zhou et al., 2001) and the events that are driving the state changes of the DIA. Events may be *deterministic* (e.g., a bouncing ball) or *nondeterministic* (e.g., an action by a user or network jitter). This categorization of events allows more efficient use of bandwidth by transmitting nondeterministic events to other users and allowing deterministic events to be computed locally (Roberts et al., 1995). In a discrete DIA the state changes in response to nondeterministic user events. The correct order of events is therefore important but the exact time of each event is not. In this case an asynchronous model of time such as the logical clock is sufficient. In a continuous DIA the state changes as a result of nondeterministic events caused by a user or by the passing of time, and so wall clock synchronization is important. In this case a correct order of all events must be maintained and the resulting state of the system must be the same as if all the events had been executed in the correct order at the time identified by their timestamps.

The foundations of time within DIAs were laid by Lamport (1978) who examined global time and the ordering of events in distributed systems. The underlying assumption is that events are related, that local events can be ordered sequentially, and that the future cannot influence the past. This is referred to as the *causality relation (before-after relation)* and formed the basis for the concept of a logical clock. In such a clock the notion of time is based on the order in which local events occur. Time is therefore asynchronous among all participating nodes in the DIA. Virtual time differs from real time in that it doesn't flow and it stands still if no events occur.

Lamport's concept of scalar time as a totally ordered sequence of causal events is not always correct. With scalar time, all processes share the same global logical clock. This means that two or more events generated by different processes can have an identical timestamp, with the possible loss of causal dependency information (Raynal & Singhal, 1996). For example, if a process generates an event based on a set of events it has received, but that new event is received at another node that hasn't received the same set of prior events, then a causality violation occurs. To solve this problem, a concept of time based on vector timestamps was developed independently by Mattern (1988) and Fidge (1988). With vector clocks, the time domain is represented by a set of vectors, one for the causal time of each process.

This set of vectors represents the global time and each node maintains a vector clock for all processes. In a DIA the problem is then to disseminate the local vector clock to all other nodes. Various algorithms exist to do this. For example, the vector can get piggybacked on data packets.

For continuous DIAs, there is an additional constraint—it must appear as if the events have been executed at the correct point in time. In this case absolute time is needed and events are defined as a function of a wall clock, independently of the rendering frame rate. Various mechanisms exist for managing the clocks in the participating nodes. All clocks may be completely synchronized by using the network time protocol (Mills, 1991) or GPS (Cox, Luiijf, van Kampen, & Ripley, 1996). Lui (2001) derived an optimal synchronization interval based on subgraph communication algorithms so that every participating node has a consistent view of the virtual world.

Roberts and Sharkey (1997a) combined the concepts of absolute and virtual time into a *sufficiently causal time stamp*. In doing so they modified the causality (before-after) relation, allowing the application to decide when to apply the causality relation to events. Some events may thus be ignored if a later event is absolute and the transitory state of no consequence. The ordering of events is termed *sufficient causal ordering* or *partial causal ordering* (Roberts & Sharkey).

The following sections describe the approaches that have been taken to maintain consistency in DIAs using both logical and wall clocks.

6.1.1 Lockstep Synchronization. This is the most basic way to ensure consistency and, being a conservative algorithm, avoids roll-back at the expense of system response (Funkhouser, 1995). It operates by preventing the generation of out-of-order events and achieves this by preventing any participating node from advancing its simulation clock until all other nodes have acknowledged that they have completed their computation. This scheme guarantees consistency but does not guarantee real-time consistency, as network latency results in acknowledgment delays and the DIA clock interval is not constant.

6.1.2 Imposed Global Consistency. Techniques that impose delays rely on delaying the execution or rendering of both local and remote events until an upper latency bound is reached. In this way the state information generated at remote nodes has sufficient time to be incorporated into the local computation of the global DIA state. The objective is to maintain synchronized global consistency at the expense of reduced responsiveness. This technique is a form of buffering and includes bucket synchronization and local lag.

In bucket synchronization, time is assumed to be synchronized and is divided into intervals or buckets of length T. The bucket frequency is thus $1/T$. All events are assigned to a bucket. The local view of the global state is then calculated using all the events generated locally and by remote nodes during the time interval $[i - D, i - D + T]$, where i is the current time interval and D is an added delay to compensate for network latency (Gautier et al., 1999). Events are therefore delayed for a length of time that should prevent any misorderings and hence avoid any rollbacks. The bucket synchronization algorithm, as applied to the distributed multiplayer game MiMaze, has been shown to provide acceptable consistency even in high latency situations (Gautier & Diot, 1998; Diot & Gautier, 1999).

The concept of local lag is related to the work of Cristian, Ahali, Stron, and Dolev (1985). Events initiated by the local user are not executed immediately but are delayed for a length of time dependent on network latency, thus reducing the responsiveness of the system. Each event is timestamped at a time later than the time the event actually occurred. An upper bound on the network delay is assumed so that events can be guaranteed to occur after a certain time. An event prologue can be added between the actual local event and its local occurrence, which appears natural to the local user. The trade-off between short-term global inconsistencies and responsiveness when using local lag in replicated continuous applications was examined in Mauve (2000b). A minimal value for local lag was based on the maximum of the average network delay between two users, whereas the highest acceptable response time was based on human perception and application type, ranging from 100 ms for drag operations to 400 ms for click operations.

The determination of an optimum local lag value was left to future work.

6.1.3 Delayed Global Consistency.

In contrast to imposed global consistency, here the objective is to maintain asynchronous global consistency. Users will perceive the same consistent world but at different times.

Work by Qin (2002) on distributed whiteboards is based on the fact that it is not necessary that users have the same state at exactly the same time. Different nodes can share the same view of the objects but at a time shift. Each event is therefore timestamped so that the global state can be reconstructed locally, albeit at a later time. This shift is specified at the design stage of the DIA. Qin refers to this as delayed consistency.

A similar idea was proposed by Sharkey et al. (1998). They employed the concepts of relativity to DIAs to warp time and limit the maximum speed of objects within the virtual world. Each user is at the origin of a temporal-spatial framework and all other users are at a relative temporal-spatial position. This allows causal surfaces and volumes to be defined and associated critical velocities for virtual objects to be determined. The causal volumes define the regions in which past events can affect present events and present events can affect future events. The critical velocity determines the maximum speed of information transmission between two users of the DIA, hence delimiting the causal boundaries to produce causal surfaces. Based on these causal surfaces, a 3½D perception filter is proposed to introduce a continuously differentiable delay contour over the virtual environment. Each local user perceives other (remote) users interacting smoothly and dynamically with their local environment in real time, but delayed by the particular latency between them and the local user. As users get closer and begin to interact, the delays between them are interpolated to zero. The causal volumes and the perception filter may potentially cause distortions in virtual time-space, leading to objects having apparent stiffness. Sharkey refers to this as *distributed stiffness.* In effect, a new set of consistent physics is established where users can understand delays and inconsistencies within the context of a relativistic physics paradigm, leading to constraints on time and space within the virtual world and as a result a new form of virtual physics fidelity.

6.1.4 Time Warp.

The Time Warp mechanism, first proposed by Jefferson (1985, 1989), is an optimistic synchronization mechanism that relies on general lookahead-rollback to maintain consistency in the DIA (Fujimoto, 1990). It operates by executing each message as soon as it arrives. When a message arrives that has a time stamp earlier than a message already executed, it undoes all the events back to that message (rollback) and starts execution again from that message. It must also send messages to undo any incorrect output messages that were communicated to other nodes while it was in an incorrect state, a process known as rollback propagation (Lin & Lazowska, 1991). To assist in rollback a snapshot of each state is taken after execution of each message. Jefferson extended the Time Warp mechanism to include optimal memory management and introduced the cancelback and the fossil collection protocols to facilitate this (Jefferson, 1990). Jefferson's Time Warp mechanism was described using a formal model by Leivent and Watro (1993).

The Breathing Time Warp algorithm is a variation of the Time Warp algorithm. This limits the amount of execution that can be done optimistically by restricting the rollback time to events that occur within an event horizon (Steinman, 1993).

Another optimistic variation on the Time Warp algorithm called Trailing State Synchronization was proposed in Cronin et al. (2002) for distributed first-person shooter games. This requires less memory and processor time compared to Time Warp. A complete time-delayed copy of the game state is maintained in parallel with the main game. This copy has more time to reorder events and is a more accurate version of the game state, although delayed in time. This trailing state is used to detect inconsistencies. When an inconsistency is detected in the actual game, the correct state can then be directly copied from the trailing state.

6.1.5 Predictive Time Management.

In predictive time management future events are pre-empted and both the event and its expected time of occurrence are transmitted before they actually occur. This is an optimistic technique for maintaining DIA consistency. The concept of predictive time management in DIAs was first proposed by Roberts et al. (1995) as part of the PaRADE system. A key motivation of their work was the issue of maintaining consistency among collaborating participants in the face of inherent network latency. Their prediction depends on knowledge of delays. They describe a study of round trip time (RTT) values to estimate delays (Roberts et al.). In PaRADE they use a sample set of message RTTs together with a robust line fitting algorithm to compute network latency between nodes. This technique relies on messages being timestamped, and thus a global wall clock is required. The prediction of events is limited to collision prediction (Sandoz, Sharkey, & Roberts, 1996) or predictions based on heuristics such as knowledge of intent, interest group or spatial proximity (Roberts & Sharkey, 1997a). This is because sudden non deterministic events cannot be predicted and deterministic events are computed locally as the effects of the event can be described mathematically. Although the occurrence of non deterministic events, initiated by humans-in-the-loop, are unpredictable certain long-term behaviors and strategies can be anticipated, and the user can be locked into a course of action in these cases (Delaney, Ward, & McLoone, 2003).

Events that are predicted locally are timestamped and may be sent to other nodes where they are executed at the appropriate time. When using prediction it must be possible to predict further ahead than the network delay, so that messages sent to other participants arrive before the event occurs locally. The execution of local events can also be delayed so that the same event will be executed at the same time across the entire DIA. In this case additional filler events, known as *event prologues,* can be added between the initiation and the occurrence of local events to compensate for the local reduced responsiveness (Worthington & Roberts, 2000). If the prediction is incorrect, rollback strategies are needed to return the DIA to a consistent state. To minimize the use of state rollback in PaRADE, events are sent just-in-time by using knowledge of network delays.

6.1.6 Concurrency. The existence of latency means that if high performance interaction is required within the DIA, then the application state will probably be replicated at each participating node (Greenberg & Marwood, 1994; Snowdon et al., 1996; Sun & Chen, 2002). However, the replication of states increases the possibility of inconsistencies and this is further exacerbated by network latency due to geographical separation. These inconsistencies can generally be corrected by using an appropriate synchronization algorithm such as Time Warp. However, when inconsistencies result from conflicts over ownership of objects, specific concurrency control mechanisms are required. Such mechanisms may be optimistic (continue until a conflict is found then roll back) or pessimistic (check for conflict before allowing event). Concurrency is an important aspect of consistency and it is highly susceptible to network latency, especially when closely-coupled interaction is involved. Margery et al. (1999) define interaction as truly concurrent object manipulation, such as two or more users manipulating a shared object. Interaction may also be sequential, with different users interacting with the same, shared object but at different times. In DIAs, the issue of two or more users trying to access the same object in the DIA at the same time, either to perform some joint task or to obtain sole possession, must therefore be addressed.

Traditional concurrency control mechanisms used in databases such as locking and serialization are not suited to applications that require a fast response (Barghouti & Kaiser, 1991). In groupware systems, a concurrency control mechanism known as Operational Transform is used (Ellis & Gibbs, 1989; Sun & Ellis, 1998). This is an optimistic mechanism that replicates documents at each node and allows local operations to be performed immediately. Concurrent remote operations are transformed before execution so that both local and remote editing results are preserved. To resolve conflicting events in a replicated DIA while minimizing latency and maximizing response time, Roberts et al. (1995) proposed a prediction-based concurrency control scheme. For one object to interact with another object in a DIA, the object initializing the interaction must possess a key (or token) to obtain ownership of the other object. In a

DIA there is therefore a delay in requesting and receiving this key as it must be sent across the network, which makes it difficult to interact with the object on the first attempt. To combat this delay, Roberts and Sharkey used predictive time management to request the key before the interaction actually takes place (Roberts & Sharkey, 1997a). In their scheme the key unlocks a hierarchy of related objects rather than just a single object. They also propose a number of techniques to avoid unnecessary passing of keys due to poor predictions.

In the prediction-based concurrency control scheme, ownership requests for the key are multicast to all potential owners of the object (Roberts & Sharkey, 1997a). The process associated with the entity gathers all requests and predicts who the next owner will be. Lee et al. (2000) extended this concurrency control scheme by employing entity-based multicast and AOI. The AOI defines a spatial distance around the entity that currently possesses the key. All other entities in the AOI are potential future owners of the key. As entities enter/leave this AOI they join/leave the multicast group associated with the entity. The next owner of the key is then predicted from among all entities in the multicast group based on direction and predicted collision time.

7 Concluding Remarks

This paper is Part I of a two-part paper that examines the issues of latency and consistency in the context of distributed interactive applications. Furthermore it describes various techniques and mechanisms that researchers have used to improve consistency and reduce latency effects. A classification for these mechanisms was proposed, based on three distinct categories—time management, information management, and system architecture. Part I of this paper explored the category of time management and the various forms that this has taken in much of the research over recent decades. The choice of technique depends on the application area and, in particular, on the type of interaction required by participants. Each technique has its advantages and disadvantages. Delaying the execution of events, rolling back events, or warping the physical environment all impact the experience of users participating in the DIA. It is therefore in the interest of the designers of such applications to balance consistency, responsiveness, and fidelity in the face of network latency. Part II of this paper continues with an examination of the two remaining categories of consistency maintenance mechanisms.

Acknowledgments

This material is based upon works supported by Enterprise Ireland under grants no. SC/2002/129/ and Ircset/SC/04/CS0289.

References

Barghouti, N., & Kaiser, G. (1991). Concurrency control in advanced database applications. *ACM Computing Surveys, 23*(3), 269–317.

Barrus, J. W., Waters, R. C., & Anderson, D. B. (1996). Locales: Supporting large multiuser virtual environments. *IEEE Computer Graphics and Applications, 16*(6), 50–57.

Bhola, S., Banavar, G., & Ahamad, M. (1998). Responsiveness and consistency tradeoffs in interactive groupware. *Computer Supported Cooperative Work (CSCW'98), ACM,* 79–88.

Blakowski, G., & Steinmetz, R. (1996). A media synchronization survey: Reference model, specification, and case studies. *IEEE Journal on Selected Areas in Communications, 14*(1), 5–35.

Blanchard, T. D., & Lake, T. (1997). A lightweight RTI prototype with optimistic publication. *Spring Simulation Interoperability Workshop,* 551–560.

Blau, B., Hughes, C. E., Moshell, J. M., & Lisle, C. (1992). Networked virtual environments. *Symposium on Interactive 3D Graphics, ACM,* 157–160.

Blow, J. (1998). A look at latency in networked games. *Game Developer, 5*(7), 28–40.

Bouras, C., Hornig, G., Triantafillou, V., & Tsiatsos, T. (2001). Architectures supporting e-learning through collaborative virtual environments: The case of INVITE. *International Conference on Advanced Learning Technologies,* 13–16.

Bowman, D. A., Johnson, D. B., & Hodges, L. F. (2001).

Testbed evaluation of virtual environment interaction techniques. *Presence: Teleoperators and Virtual Environments, 10*(1), 75–95.

Broll, W. (1997). Distributed virtual reality for everyone — A framework for networked VR on the internet. *Virtual Reality Annual International Symposium (VRAIS '97),* IEEE, 121–128.

Calvin, J., Dickens, A., Gaines, B., Metzger, P., Miller, D., & Owen, D. (1993). The Simnet virtual world architecture. *Virtual Reality Annual International Symposium (VRAIS '93), IEEE,* 450–455.

Capps, M. V. (2000). The QUICK framework for task-specific asset prioritization in distributed virtual environments. *Virtual Reality (VR '00), IEEE Computer Society,* 140–153.

Capps, M., McDowell, P., & Zyda, M. (2001). A future for entertainment-defense research collaboration. *IEEE Computer Graphics and Applications, 21*(1), 37–43.

Capps, M., McGregor, D., Brutzman, D. P., & Zyda, M. (2000). NPSNET-V. A new beginning for dynamically extensible virtual environments. *IEEE Computer Graphics and Applications, 20*(5), 12–15.

Cheshire, S. (1996). *Latency and the quest for interactivity.* A white paper commissioned by Volpe Welty Asset Management, L.L.C., for the synchronous person-to-person interactive computing environments meeting. Available at www.stuartcheshire.org.

Churchill, E. F., Snowdon, D. N., & Munro, A. J. (2001). *Collaborative virtual environments—Digital places and spaces for interaction.* London, UK: Springer-Verlag.

Cox, A., Luiijf, E., van Kampen, R., & Ripley, R. (1996). Time synchronization experiments. *14th Workshop on Standards for the Interoperability of Distributed Simulations,* 599–613.

Cristian, F., Ahali, H., Stron, R., & Dolev, D. (1985). Atomic broadcast: From simple message diffusion to byzantine agreement. *15th International Symposium on Fault-Tolerant Computing,* 200–206.

Cronin, E., Filstrup, B., Kurc, A. R., & Jamin, S. (2002). An efficient synchronization mechanism for mirrored game architectures. *First Workshop on Network and System Support for Games,* 66–73.

Delaney, D., Ward, T., & McLoone, S. (2003). Reducing update packets in distributed interactive applications using a hybrid model. *16th International Conference on Parallel and Distributed Computing Systems,* 417–422.

de Oliveira, J. C., & Georganas, N. D. (2002). VELVET: An adaptive hybrid architecture for very large virtual environments. *IEEE International Conference on Communications,* 2491–2495.

de Oliveira, J. C., Shirmohammadi, S., & Georganas, N. D. (1999). Collaborative virtual environment standards: A performance evaluation. *3rd International Workshop on Distributed Simulation and Real Time Applications (DS-RT '99),* 14–21.

Diot, C., & Gautier, L. (1999). A distributed architecture for multiplayer interactive applications on the internet. *IEEE Network, 13*(4), 6–15.

Dourish, P. (1995). The parting of ways: Divergence, data management and collaborative work. *Fourth European Conference on Computer Supported Cooperative Work (ECSCW '95),* 215–230.

Dutta-Roy, A. (2000). The cost of quality in internet-style networks. *IEEE Spectrum, 37*(9), 57–62.

Ellis, S. R. (1994). What are virtual environments? *IEEE Computer Graphics and Applications, 14*(1), 17–22.

Ellis, C. A., & Gibbs, S. J. (1989). Concurrency control in groupware systems. *Conference on Management of Data,* 399–407.

Fidge, C. (1988). Timestamp in message passing systems that preserves partial ordering. *11th Australian Computing Conference,* 56–66.

Frécon, E. (2003). DIVE: A generic tool for the development of virtual environments. *7th International Conference on Telecommunications (ConTEL 2003),* 345–352.

Frécon, E. (2004a). *Dive on the internet.* Doctoral dissertation, *Swedish Institute of Computer Science,* IT University of Goteborg, Goteborg, Sweden.

Frécon, E. (2004b). DIVE: Communication architecture and programming model. *IEEE Communications Magazine, 42*(4), 34–40.

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

Fujimoto, R. M. (1990). Parallel discrete event simulation. *Communications of the ACM, 33*(10), 30–53.

Fujimoto, R. M. (2001). Parallel and distributed systems. *Winter Simulation Conference,* 147–157.

Funkhouser, T. A. (1995). RING: A client-server system for multiuser virtual environments. *Symposium on Interactive 3D Graphics,* 85–92.

Garcia, P., Montala, O., Pairot, C., Rallo, R., & Skarmeta, A. G. (2002). MOVE: Component groupware foundations for collaborative virtual environments. *4th International*

Conference on Collaborative Virtual Environments (CVE02), 55–62.

Gautier, L., & Diot, C. (1998). Design and evaluation of MiMaze, a multi-player game on the internet. *International Conference on Multimedia Computing and Systems (ICMCS '98),* 233–236.

Gautier, L., Diot, C., & Kurose, J. (1999). End-to-end transmission control mechanisms for multiparty interactive applications on the internet. *18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '99),* 1470–1479.

Greenberg, S., & Marwood, D. (1994). Real time groupware as a distributed system: Concurrency control and its effect on the interface. *Computer Supported Cooperative Work (CSCW '94),* 207–218.

Greenhalgh, C., & Benford, S. (1995). MASSIVE: A collaborative virtual environment for teleconferencing. *ACM Transactions on Computer-Human Interaction (TOCHI), 2*(3), 239–261.

Greenhalgh, C., Purbick, J., & Snowdon, D. (2000). Inside MASSIVE-3: Flexible support for data consistency and world structuring. *3rd International Conference on Collaborative Virtual Environments,* 119–127.

Greif, I. (1988). Computer-supported cooperative work: A book of readings. San Mateo, CA: Morgan Kaufmann.

Gutwin, C. (2001). The effects of network delays on group work in real-time groupware. Seventh European Conference on Computer-Supported Cooperative Work (ECSCW '01), (pp. 299–318) Dordrecht, The Netherlands: Kluwer.

Hecker, C., & Simpson, Z. B. (2001). Dead reckoning a.k.a. motion prediction. *Game Developer, 8*(2), 10.

Henderson, T. N. H. (2003). *The effects of relative delay in networked games.* Doctoral dissertation, Department of Computer Science, University College, London.

Holbrook, H. W., Singhal, S. K., & Cheriton, D. R. (1995). Log-based receiver-reliable multicast for distributed interactive simulation. *ACM SIGCOMM Computer Communications Review, 25*(4), 328–334.

IEEE. (1993). *IEEE standard for distributed interactive simulation—Application protocols.* IEEE Std 1278-1993I. New York: IEEE.

IEEE. (1995). *IEEE standard for distributed interactive simulation—Application protocols.* IEEE Std 1278.1-1995 (Revision of IEEE Std 1278-1993). New York: IEEE.

Jefferson, D. R. (1985). Virtual time. *ACM Transactions on Programming Languages and Systems, 7*(3), 404–425.

Jefferson, D. R. (1989). Virtual time. *ACM Transactions on Programming Languages and Systems, 7*(1), 404–425.

Jefferson, D. R. (1990). Virtual time II: Storage management in distributed simulation. *9th Annual Symposium on Principles of Distributed Computing,* 75–89.

Jehaes, T., De Vleeschauwer, D., Coppens, T., Van Doorselaer, B., Deckers, E., Naudts W., et al. (2003). Access network delay in networked games. *2nd Workshop on Network and System Support for Networked Games (NetGames '03),* 63–71.

Joslin, C., Di Giacomo, T., & Magnenat-Thalmann, N. (2004). Collaborative virtual environments: From birth to standardization. *IEEE Communications Magazine, 44*, 28–33.

Joslin, C., Molet, T., Thalmann, N., Esmerado, J., Thalmann, D., Palmer, I., et al. (2001). Sharing magnetic attractions on the net with VPark. *IEEE Computer Graphics and Applications, 21*(1), 61–71.

Kelly, T. (1997). Dr. T: A communications infrastructure for distributed interactive applications. Available at http://ai.eecs.umich.edu/~kelly/papers/.

Krumm-Heller, A., & Taylor, S. (2000). Using determinism to improve the accuracy of dead reckoning algorithms. *SimTecT2000—The Simulation Technology and Training Conference.*

Lamport, L. (1978). Time clocks and the ordering of events in a distributed system. *Communications of the ACM, 21*(7), 558–565.

Lee, D., Lim, M., & Han, S. (2002). ATLAS—A scalable network framework for distributed virtual environments. *4th International Conference on Collaborative Virtual Environments (CVE02),* 47–54.

Lee, D., Yang, J., Yoon, H. Y., Yu, C., & Hyun, S. J. (2000). Entity-centric scalable concurrency control for distributed interactive applications. *International Performance, Computing, and Communications Conference (IPCCC '00),* 544–550.

Leigh, J., Yu, O., Schonfeld, D., Ansari, R., He, E., Nayak, A., et al. (2001). Adaptive networking for tele-immersion. *5th Immersive Projection Technology/7th Eurographics Virtual Environments Workshop (IPT/EGVE).*

Levient, J. I., & Watro, R. J. (1993). Mathematical foundations for time warp systems. *ACM Transactions on Programming Languages and Systems, 15*(5), 771–794.

Lin, Y.-B., & Lazowska, E. D. (1991). A study of time warp rollback mechanisms. *ACM Transactions on Modelling and Computer Simulation, 1*(1), 51–72.

Lui, J. C. S. (2001). Constructing communication subgraphs and deriving an optimal synchronization interval for distributed virtual environment systems. *IEEE Transactions on Knowledge and Data Engineering, 13*(5), 778–792.

MacColl, I., Millard, D., Randell, C., & Steed, A. (2002). Shared visiting in EQUATOR City. *4th International Conference on Collaborative Virtual Environments (CVE02),* 88–94.

Macedonia, M. R. (1995). *A network software architecture for large scale virtual environments.* Doctoral dissertation, Department of Computer Science, Naval Postgraduate School, Monterey, California.

Macedonia, M., & Zyda, M. (1997). A taxonomy for networked virtual environments. *IEEE Multimedia, 4*(1), 48–56.

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.

Manninen, T. (2000). Interaction in networked virtual environments as communicative action: Social theory and multiplayer games. *6th International Workshop on Groupware (CRIWG2000),* 154–157.

Margery, D., Arnaldi, B., & Plouzeau, N. (1999). A general framework for cooperative manipulation in virtual environments. *Eurographics Workshop,* 169–178.

Martin, H. S., McGregor, A. J., & Cleary, J. G. (2000). Analysis of internet delay times. *1st Passive and Active Measurements Workshop (PAM '00),* 141–149.

Mattern, F. (1988). Virtual time and global states of distributed systems. *International Workshop on Parallel and Distributed Algorithms,* 215–226.

Mauve, M. (2000a). Consistency in replicated continuous interactive media. *Computer Supported Cooperative Work (CSCW '00),* 181–190.

Mauve, M. (2000b). *Distributed interactive media.* Doctoral dissertation, University of Mannheim, Berlin.

Mauve, M., Hilt, V., Kuhmunch, C., & Effelsberg, W. (2001). RTP/I — Towards a common application level protocol for distributed interactive media. *IEEE Transactions on Multimedia, 3*(1), 152–161.

Mauve, M., Vogel, J., Hilt, V., & Effelsberg, W. (2002). Local-lag and timewarp: Providing consistency for replicated continuous applications. *IEEE Transactions on Multimedia, 2002.*

Mauve, M., Vogel, J., Hilt, V., & Effelsberg, W. (2004). Local-lag and timewarp: Providing consistency for replicated continuous applications. *IEEE Transactions on Multimedia, 6*(1), 47–57.

McCoy, A., Delaney, D., & Ward, T. (2003). Game-state fidelity across distributed interactive games. *ACM Crossroads, Networking Issue, 9*(4), 4–9.

Meehan, M., Razzzaque, S., Whitton, M. C., & Brooks, F. P. J. (2003). Effect of latency on presence in stressful virtual environments. *Virtual Reality (VR '03),* 141–148.

Mills, D. L. (1991). Internet time synchronization: The network time protocol. *IEEE Transactions on Communications, 39*(10), 1482–1493.

Natrajan, A., & Reynolds, P. F. Jr. (1999). Resolving concurrent interactions. *3rd International Workshop on Distributed Simulation and Real Time Applications (DS-RT '99)* 85–92.

Park, K. S., & Kenyon, R. V. (1999). Effects of network characteristics on human performance in a collaborative virtual environment. *Virtual Reality (VR '99),* 104–111.

Pullen, J. M., & Wood, D. C. (1995). Network technology and DIS. *Proceedings of the IEEE 83*(83), 1156–1167.

Qin, X. (2002). Delayed consistency model for distributed interactive systems with real-time continuous media. *Journal of Software, 13*(6), 1029–1039.

Radhakrishnan, R., & Wilsey, P. A. (1999). Ruminations on the implications of multi-resolution modelling on DIS/HLA. *3rd International Workshop on Distributed Simulation and Real Time Applications (DS-RT '99),* 101–108.

Raynal, M., & Singhal, M. (1996). Logical time: Capturing causality in distributed systems. *Computer, 29*(2), 49–56.

Riva, G., & Gamberini, L. (2001). Virtual reality in telemedicine. *Telemedicine Journal and e-Health, 6*(3), 327–340.

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

Roberts, D. J., & Sharkey, P. M. (1997a). Maximizing concurrency and scalability in a consistent, causal, distributed virtual reality system, whilst minimizing the effect of network delays. *6th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises (WET-ICE '97),* 161–166.

Roberts, D. J., & Sharkey, P. M. (1997b). Minimizing the latency induced by consistency control within a large scale multi-user distributed virtual reality system. *IEEE Conference on Systems, Man and Cybernetics, 5,* 4492–4497.

Roberts, D. J., Sharkey, P. M., & Sandoz, P. D. (1995). A real-time predictive architecture for distributed virtual reality. *1st Workshop on Simulation and Interaction in Virtual Environments,* 279–288.

Roza, M., Voogd, J., Jense, H., & van Gool, P. (1999). Fidelity requirements specification: A process oriented view. *Fall Simulation Interoperability Workshop,* paper 99F-SIW-032.

Sandoz, P. D., Sharkey, P. M., & Roberts, D. J. (1996). Collision prediction of a moving object within a virtual world. *Virtual Reality World 96.*

Schwarz, R., & Mattern, F. (1994). Detecting causal relationships in distributed computations: In search of the holy grail. *Distributed Computing, 7*(3) 149–174.

Sharkey, P. M., Ryan, M. D., & Roberts, D. J. (1998). A local perception filter for distributed virtual environments. *Virtual Reality Annual International Symposium (VRAIS '98),* 242–249.

Shneiderman, B. (1984). Response time and display rate in human performance with computers. *Computing Surveys, 16*(3), 625–685.

Singhal, S. K. (1996). Effective remote modeling in large-scale distributed simulation and visualization environments. Doctoral dissertation, Stanford University, Stanford, California.

Singhal, S. K., & Zyda, M. (1999). *Networked virtual environments.* New York: ACM Press.

Smed, J., Kaukoranta, T., & Hakonen, H. (2001). Aspects of networking in multiplayer computer games. *International Conference on Application and Development of Computer Games in the 21st Century,* 74–81.

Snowdon, D., Greenhalgh, C., Benford, S., Bullock, A., & Brown, C. (1996). A review of distributed architectures for networked virtual reality. *Virtual Reality: Research, Development and Applications, 2*(1), 155–175.

Steinman, J. S. (1993). Breathing Time Warp. *7th International Workshop on Parallel and Distributed Simulation (PADS93),* 109–118.

Stytz, M. R. (1996). Distributed virtual environments. *IEEE Computer Graphics and Applications, 16*(3), 19–31.

Sun, C., & Chen, D. (2002). Consistency maintenance in real-time collaborative graphics editing systems. *ACM Transactions on Computer-Human Interaction (TOCHI), 9*(1), 1–41.

Sun, C., & Ellis, C. A. (1998). Operational transformation in real-time group editors: Issues, algorithms and achievements. *Computer Supported Cooperative Work (CSCW '98),* 59–68.

Sun, C., Jia, X., Zhang, Y., Yang, Y., & Chen, D. (1998). Achieving convergence, causality preservation and intention preservation in real-time cooperative editing systems. *ACM Transactions on Computer-Human Interaction (TOCHI), 5*(1), 63–108.

Tawfik, H., & Fernando, T. (2001). A simulation environment for construction site planning. *5th International Information Visualisation Conference,* 199–204.

Tse-Au, E. S. H., & Morreale, P. A. (2000). End-to-end QoS measurement: Analytic methodology of application response time vs. tunable latency in IP neworks. *IEEE/IFIP Network Operations and Management Symposium (NOMS '00),* 129–142.

Vaghi, I., Greenhalgh, C., & Benford, S. (1999). Coping with inconsistency due to network delays in collaborative virtual environments. *Symposium on Virtual Reality Software and Technology (VSRT '99),* 42–49.

Vogel, J., Mauve, M., Hilt, V., & Effelsberg, W. (2003). Late join algorithms for distributed interactive applications. *Multimedia Systems, 9*(4), 327–336.

Waters, R. C., & Barrus, J. W. (1997). The rise of shared virtual environments. *IEEE Spectrum, 34*(3), 20–25.

Wloka, M. (1995). Lag in multiprocessor VR. *Presence: Teleoperators and Virtual Environments, 4*(1), 50–63.

Worthington, B. G., & Roberts, D. J. (2000). Encapsulating network latency compensators in VRML. *Virtual Worlds and Simulation Conference (VWSIM '00),*

Zeigler, B. P., Praehofer, H., & Kim, T. G. (2000). *Theory of modeling and simulation.* San Diego, CA: Academic Press.

Zhou, S. P., Cai, W. T., Lee, F. B. S., & Turner, S. J. (2001). Consistency in distributed interactive applications. *European Simulation Interoperability Workshop 2001 (Euro-SIW 2001),* 01E-SIW-003.

Zhou, S., Cai, W., Lee, B.-S., & Turner, S. J. (2003). Time-space consistency in large scale distributed virtual environment. *ACM Transactions on Modeling and Computer Simulation, 14*(1), 31–47.