Optimizing Consistency by Maximizing Bandwidth Usage in Distributed Interactive Applications

DAMIEN MARSHALL, SÉAMUS MCLOONE, TOMÁS WARD, National University of Ireland, Maynooth

A key factor determining the success of a Distributed Interactive Application (DIA) is the maintenance of a consistent shared virtual world. To help maintain consistency, a number of Information Management techniques have been developed. However, unless carefully tuned to the underlying network, they can negatively impact on consistency. This work presents a novel adaptive algorithm for optimizing consistency by maximizing available bandwidth usage in DIAs. This algorithm operates by estimating bandwidth from trends in network latency, and modifying data transmission rates to match the estimated value. Results presented within demonstrate that this approach can help optimise consistency levels in a DIA.

Categories and Subject Descriptors: I.6.8 [**Simulation and Modelling**]: Types of Simulation—*Gaming*

General Terms: Measurements, Performance.

Additional Key Words and Phrases: Adaptive algorithms, multiplayer games, information management techniques, consistency

ACM Reference Format:

Marshall, D., McLoone, S., and Ward, T. 2010. Optimizing consistency by maximizing bandwidth usage in distributed interactive applications. ACM Trans. Multimedia Comput. Commun. Appl. 6, 4, Article 30 (November 2010), 23 pages.
DOI = 10.1145/1865106.1865114 http://doi.acm.org/10.1145/1865106.1865114

1. INTRODUCTION

Distributed Interactive Applications (DIAs) have become extremely popular, both socially and commercially, in recent years. At their core, they involve multiple participants collaborating and competing within a virtual environment, even though those participants may be located at geographically separate locations [Macedonia and Zyda 1997]. Popular examples of DIAs include online games such as Counter Strike and World of Warcraft (www.blizzard.com), collaborative virtual environments such as DIVE [Frecon and Stenius 1998], and virtual meeting places such as Second Life and Playstation Home (www.playstationhome.com) on the Playstation 3.

In a DIA, each participant typically controls a virtual entity or avatar. This is the virtual representation of the participant, and it is described by a number of state variables. Example state variables include position and orientation. Communication between participants is achieved by sharing any changes to the value of these state variables using synchronization messages. These are transmitted periodically across the network connecting participants, and update the remote state of the virtual entity, which is the state replicated on other participants' host computers.

This work is supported by Science Foundation Ireland and Enterprise Ireland, Grant no. IRCSET/SC/04/CS0289.
Contact author: D. Marshall: email: Damien.marshall@itcarlow.ie

DOI 10.1145/1865106.1865114 http://doi.acm.org/10.1145/1865106.1865114

In order for collaboration and competition between participants within a DIA to be meaningful, it is vital that all changes to state variables are delivered to relevant participant nodes in a timely manner [Claypool and Claypool 2006]. However, each synchronization message is subjected to the limitations of the network connecting participants, meaning that this timeliness constraint can be broken. One of the most significant limitations is network bandwidth [Roehle 1997]. The bandwidth of a network link is a measure of maximum throughput of traffic on that link. A network path is typically made up of many links, each with separate bandwidth characteristics. On such a path, the maximum capacity of the network path is determined by the link with the lowest available bandwidth, known as the bottleneck link. In the current-day Internet, the bottleneck link is typically the link connecting the home user to their Internet Service Provider, also referred to as the "last mile link" [Jehaes et al. 2003; Dube et al. 2005].

If the data being transmitted by a DIA across the bottleneck link exceeds the available bandwidth of that link, then data will need to be buffered or dropped until the flow of data decreases, causing an increase in network latency, and possible loss of data. When this occurs, participants involved in the DIA will have to update their version of the virtual world with, at best, out of date state data, or, at worst, no state data at all. This will then cause the value of the world state variables on each participant host machine to diverge. This divergence is known as inconsistency [Dourish 1995; Vaghi et al. 1999; Delaney et al. 2006a]. Inconsistency can manifest itself in many ways in a DIA, depending on the nature of the data transmitted by the DIA. In general, inconsistency can be considered from either a system or user perspective. A system-centric view of consistency means that inconsistency is quantified in terms of application-specific characteristics, regardless of how the end user perceives that inconsistency. An example of this is the spatial distance between a local and remote version of an entity at the same time instance. On the other hand, user centric consistency quantifies inconsistency in terms of the end user experience [Xue et al. 2002]. An example of this type of measure includes the level of latency that has a perceptual impact on user experience within the DIA.

In a modern DIA, the individual flows of data transmitted between participants are very thin, and can typically fit within the capabilities of a 33kbit/s modem [Feng et al. 2002]. Inconsistencies then arise due to the aggregation of these streams as an application attempts to accommodate an increasing numbers of players, particularly across the last-mile upstream connection of a modern ADSL network. For example, an ADSL broadband connection with a 2Mbit/s downstream link may have only a 256kbit/s upstream link. In this case, even though a single stream may be only 33kbit/s, the aggregate data requirements of, for example, 10 players, would quickly overload the 256kbit/s upstream link. This issue manifests itself in Peer-to-Peer scenarios, where a peer must transmit data to every connected participant, and in Client/Server scenarios, where the server must transmit world state to each connected client.

In order to improve consistency, a suite of algorithms, collectively referred to as Information Management (IM) techniques, have been developed. [Singhal and Zyda 1999; Bernier 2001; Delaney et al. 2006b; McCoy et al. 2007]. These algorithms attempt to optimize the streams of data transmitted by a DIA, and thus reduce the inconsistency caused by overloaded network hardware. IM techniques operate by purposefully introducing a controlled level of inconsistency into the DIA by reducing the number of synchronization messages transmitted between participants. Although this means that state inconsistency increases, the reduction in the amount of data transmitted across the network should, in theory, reduce the load on the bottleneck link, and reduce latency. This in turn, would then reduce overall inconsistency in comparison to transmitting all synchronization messages and overloading the bottleneck link.

However, in practice, use of an IM technique does not always result in an improvement in consistency. Consider the scenario presented in Figure 1. Here, inconsistency arising within a DIA when an

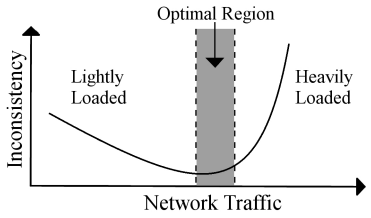

Fig. 1. Inconsistency in different network conditions.

IM technique is not employed is plotted against increasing network traffic that is being transmitted across a single node with a finite amount of bandwidth. In the heavily loaded region, an IM technique could be used to reduce the load on the bottleneck link and reduce network latency. However, if the bottleneck link is not overloaded, then using an IM technique reduces data transmission rates, thus incurring extra state inconsistency, with little or no reduction in network latency. This increases overall inconsistency [Marshall et al. 2006b].

This issue arises as the level of inconsistency introduced by an IM technique is typically governed by system or user centric control parameters, and the characteristics of the underlying network connecting participants are rarely considered. This means that the real network level impact of using an IM technique is largely unknown, so it can be difficult to ascertain whether an IM technique is truly improving consistency. In this work, we advocate a move towards using network centric control parameters for determining data transmission rates in DIAs. The value of these parameters is governed by the characteristics of the underlying network connecting participants. By doing this, the optimal region of data transmission that gives the lowest inconsistency in the current network environment can be maintained, as highlighted in Figure 1.

This article presents the design and evaluation of a novel information management technique, which we refer to as the Consistency Optimization (CO) algorithm. This application layer technique is designed to optimize the consistency of the data transmitted by a Peer-to-Peer or Client/Server DIA, where the last-mile link is the bottleneck link. It operates by unobtrusively monitoring trends in network latency, and using these trends to reduce data transmission rates only when the reduction will improve consistency. The algorithm attempts to minimize the reduction in data transmission rates by estimating the available bandwidth on the bottleneck link when congestion occurs, and adjusting the data generation rates of the DIA to match the estimated value. Results collected from a variety of live Internet trials conducted using this technique demonstrate that it can accurately modify the data generation rates of a DIA to match the available bandwidth on the bottleneck link. In doing so, it can optimize and improve the level of consistency arising in the DIA.

The rest of the article is laid out as follows. Section 2 examines the link between the management of data transmission within DIAs and inconsistency, and demonstrates the motivating factors behind the development of the CO algorithm. Section 3 details the operation of the CO algorithm. In Section 4, results are presented that demonstrate that our approach can accurately estimate the resources available on the last-mile link in a typical ADSL broadband scenario, and suitably adapt the data transmitted by the application. In doing so, the inconsistency arising during the execution of the DIA can be minimized. The impact of cross traffic on the CO Algorithm is also explored. Section 5 provides a comparative study of the CO algorithm with existing techniques. Finally, the article concludes in Section 6, with a discussion of the results, and some suggestions for future work.

2. INFORMATION MANAGEMENT AND INCONSISTENCY

In this section, the motivating factors behind the need for the Consistency Optimization algorithm are explored. To do this, a series of live Internet trials were conducted to investigate the link between data transmission rates and inconsistency using a representative DIA. All trials were conducted between Portlaoise, Ireland, and Carlow, Ireland. The former is connected to a wireless access point, which is in turn connected to the Internet via a residential ADSL broadband connection provided by BT Ireland, while the latter is connected over a corporate network. The upstream link of the broadband connection is the bottleneck link in the network. Although the advertised speed was 128kbits/s, its actual capacity was measured at 100kbits/s using PathLoad [Jain and Dovrolis 2002]. Using the "Traceroute" tool provided in Windows Vista, a total of 7 hops were measured between the two end points.

Due to the large number of players typically involved in a DIA, it was logistically unfeasible at the time of testing to employ a single node per player. To compensate for this, three separate nodes were involved in each test, regardless of the number of players in the DIA. The main player node was located behind the bottleneck link. This node transmitted state data to a varying number of nodes, and represented either a server transmitting to many clients, or a peer transmitting to many connected peers. Another node was located behind the corporate connection at Carlow. Its function was to receive and simulate the state data received from the Portlaoise node. Finally, another node was also located at another residential connection in Portlaoise, and represented a "sink node." This node received all the data relating to all other players in the DIA from the main player node, and allowed the main player node to transmit data to varying numbers of players across its upstream link. So, for example, in a 7-player Client/Server simulation, the main player node would transmit one stream of data to the node at Carlow, and six streams to the sink node. This means that, although the Portlaoise node is only transmitting to 2 physical nodes, the number of application-level connections that compete for bandwidth at the bottleneck link scales linearly with the number of players maintained. Thus, it is representative of behaviour of a typical DIA across the same link.

Over this network, a testbed application was executed. This application consisted of a simple racing game. The state of each participant in this test application is described by its position and velocity. This data was generated using a path curvature simulator developed in Matlab [Marshall et al. 2006a], and features a typical curving path found in a racing application. Clocks were synchronized between the nodes at Portlaoise and Carlow.

The control parameters of the application were configured to be representative of modern-day DIAs. First, as many DIAs aim to present a smooth and fast-paced visual feedback to the participant, their state is updated often, typically between 30Hz and 60Hz. Thus, for the racing application, the simulation was updated at 50Hz. On each simulation tick, the main player node at Portlaoise recorded the time stamped value of its own state variables, while the node at Carlow recorded the time stamped state of the Portlaoise node. This is referred to as the local and remote state respectively. Spatial inconsistency is calculated as the Euclidean difference between the local and remote positional information at each time step.

Second, to ensure that the data transmission rates within the test scenario were realistic and in line with modern DIAs, the testbed application was configured as follows. The maximum packet rate of the application was 30 Packets Per Second (PPS), which is representative of modern DIAs such as Counter-Strike [Claypool et al. 2003; Lang et al. 2004]. When a packet is available for transmission, the most recent state information is transmitted. The payload of each packet transmitted was set to 50 bytes. This value is again representative of a modern DIA [Feng et al. 2002; Lang et al. 2004]. The header of each packet, including PPPoE (Point to Point Protocol over Ethernet) and Ethernet header was 72 bytes, resulting in a total packet size of 122 bytes. As the packet rate was always lower than

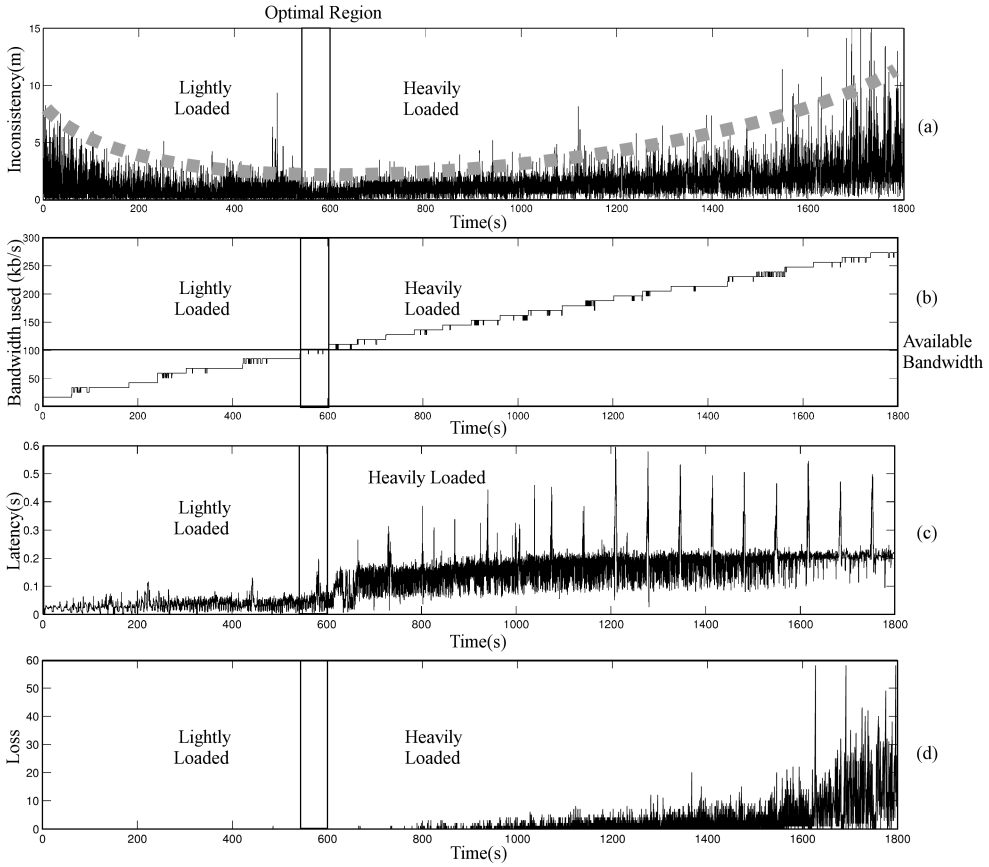

Fig. 2. (a) Inconsistency, (b) Bandwidth utilization, (c) Network Latency, (d) Packet Loss as experienced by the main player node during the initial test scenario.

the simulation rate, there are often cases where the Carlow node does not receive a packet between simulation ticks. In this case, the previously received velocity information is used to extrapolate the current position.

For test purposes, each packet was timestamped and contained a sequence number, which were used to measure one-way delay and packet loss, respectively. Latency is measured as the time between sending a particular packet from the local application, and receiving that same packet on the remote application. Packet loss is measured as the burst of packets lost between two consecutively received packets. So, for example, if a packet with sequence number 10 was received directly after a packet with sequence number 5, then loss is measured as 4 packets.

An experiment to examine the link between data transmission and inconsistency was conducted using this experimental testbed. In this experiment, the main player node located behind the ADSL connection transmitted data to 8 separate participants for 30 minutes. At the start of the DIA simulation, the packet transmission rate to each participant was 1 Packet Per Second (PPS). Every minute, the PPS was increased by 1, resulting in 30 PPS being sent to each participant by the end of the test. Results from this experiment are presented in Figure 2. Here, spatial inconsistency, bandwidth utilization, network latency, and packet loss, as experienced by the main player node, are presented. In

Figure 2(a), *m* represents virtual metres within the game world. As a reference point for this measure, the average speed of the entity was approximately 14 meters per second, and the height of the entity was approximately 2 meters.

Analyzing the spatial inconsistency in Figure 2(a) clearly shows the presence of the U-shaped trend in inconsistency, as highlighted by the dashed line, and the optimal data transmission point, as identified in Figure 1. From Figure 2(b), (c), and (d), it can be seen that up until approximately 600 seconds, the bandwidth is below 100kbit/s, during which time network latency and packet loss is minimal. This verifies the estimate taken from PathLoad. During this time period, data generation rates of the DIA can be increased to reduce inconsistency without affecting latency or loss. Using an Information Management technique to reduce data transmission rates at this point would be pointless, as it would only induce inconsistency unnecessarily. After 600 seconds, the data generated by the DIA begins overloading the bandwidth available on the link, resulting in a substantial increase in both latency and packet loss as bandwidth usage increases, and a subsequent increase in inconsistency.

As a reference point for the network latency values, consider that previous work has shown that the range of acceptable network latency values for DIAs falls between 0.06s and 0.25s [Claypool and Claypool 2006]. When latency exceeds these value ranges, the experience of the participant is negatively impacted upon. In this heavily loaded area, a reduction in data transmission rates is required in order to improve overall consistency. Overall, these results suggest that for this particular case with 8 players, a bandwidth usage of 93.2 kbits/s (or 12 PPS for each player), gives the optimal trade-off between bandwidth usage and inconsistency. The results presented in Figure 2 demonstrate the need for a scheme that can actively monitor the impact a DIA is having on the underlying network in order to minimize the inconsistency arising during runtime. In the next section, the operation of such a scheme, which we refer to as the Consistency Optimisation algorithm, is discussed.

3. OVERVIEW OF THE CONSISTENCY OPTIMIZATION ALGORITHM

The operation of our proposed algorithm is now outlined. This algorithm has been designed to operate in conjunction with existing DIA synchronization messages transported using UDP. Key to the operation of the algorithm is the "logical connection," which is a common construct in DIAs that employ UDP as a transport protocol. This is an application-layer connection that the server maintains to each client in a Client/Server scenario, or each peer maintains for each other peer involved in a Peer-to-Peer scenario. The logical connection represents a software channel of communication between a local and remote endpoint. Multiple logical connections transmit data over the same physical link, giving rise to aggregation of traffic at the bottleneck link discussed in Section 1.

Network latency is employed as a congestion indicator. A separate network latency profile and data transmission rate is maintained for each logical connection, meaning that various data transmission rates can be supported within the one application instance. The trends in the network latency profile are constantly monitored, and once congestion is detected via an increase in network latency, the available bandwidth is estimated using the measured magnitude of congestion. The data transmission rates of the appropriate logical connections are then modified to match the estimated available bandwidth.

On each logical connection, there are two key elements to the Consistency Optimization algorithm, as shown in the flowchart in Figure 3. The first element is the detection of impending congestion, which is carried out collaboratively between local and remote endpoints of a logical connection. The operation of this is highlighted in area 1 and area 2 of Figure 3. The second is the bandwidth estimation and reaction to congestion, which is carried out locally, and is highlighted in area 3 of Figure 3. The operation of each element will now be described in detail.

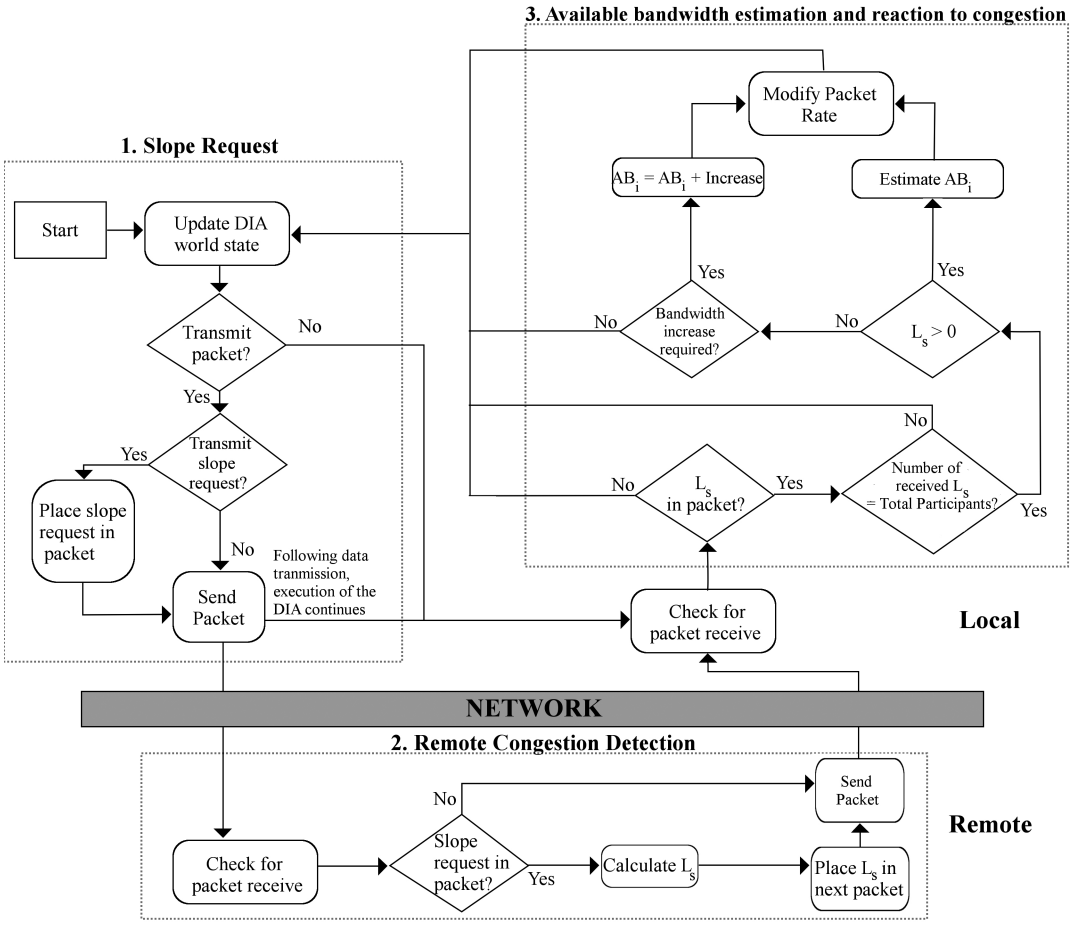

Fig. 3. Flowchart demonstrating the operation of the CO algorithm.

3.1 Congestion Detection

In the CO algorithm, one-way network latency is used as a congestion indicator. Traditionally, recording one-way network latency requires synchronized clocks. However, this requires extra time-related data to be transmitted within each packet, and is subject to clock skew effects. To deal with this issue, our scheme employs an unobtrusive method of tracking latency trends [Marshall et al. 2007]. This method operates by predicting when packets should arrive from a sender, given that the sender is transmitting packets at a constant rate. If a packet arrives before or after the predicted time, then latency has decreased or increased respectively, relative to the initial value received. This gives a measure of "relative latency." If a sender changes their sending rate, then this information is transmitted to interested recipients. This is the only information transmitted as part of the latency-monitoring scheme. In this way, the trends in latency can be monitored, without imposing extra data requirements on the application. This approach is similar in operation to that of "Phase Jitter," which has been employed to drop VOIP calls on congested last-mile links [McGovern et al. 2006].

Each remote endpoint of a logical connection maintains a relative latency trend. As can be seen in area 1 of Figure 3, at regular intervals a "slope request" message is transmitted to all connected nodes.

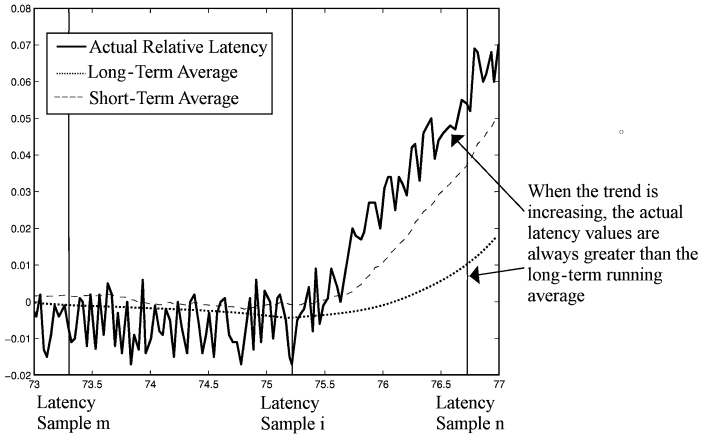

Fig. 4. Example of relative latency samples with long and short term averages. Key samples used in trend and slope calculation are highlighted.

This message indicates to all connected nodes that they should perform analysis of the trends in network latency, and return the result of this analysis to the originating host. To avoid transmission of extra data on the network, this request is piggybacked on existing application data, and not transmitted in a separate packet. Upon receipt of this request, the remote endpoint determines if congestion is occurring on the connecting link by analysing the trends in network latency occurring since the last slope request message was received.

To determine trends in network latency, three separate collections of relative latency values are maintained for each logical connection, as highlighted in Figure 4. The first, $L_{ACT,}$ contains the actual relative latency values. The second, L_{LTA}, is a long-term running average of the actual relative latency values and is used to detect the presence of trends. Lastly, L_{STA}, is a short-term running average of the actual relative latency values and is used to calculate trend magnitude. The two separate trends are employed to reduce the impact of network jitter on the operation of the trend identification scheme. In the presence of network jitter, L_{LTA} simplifies the process of increasing trend identification, while L_{STA} allows for accurate calculation of trend magnitude.

It can be seen from Figure 4 that if the short-term latency values, L_{STA}, are always greater than the long-term average, L_{LTA}, then network latency is showing an increasing trend. With this in mind, a modified version of the Pairwise Comparison Test can be used to automate increasing trend detection [Jain and Dovrolis 2002]. Given a relative latency sample, n, and an earlier sample m, the result of the Pairwise Comparison Test, PCT, can be calculated using Equation (1).

$$PCT = \frac{\sum_{s=m\ldots n} F(s)}{(n-m)}, \tag{1}$$

where $F(s) = \begin{cases} 1 & if\ L(s)_{ACT} > L(s)_{LTA} \\ 0 & if\ L(s)_{ACT} \leq L(s)_{LTA}. \end{cases}$

If the value of PCT is greater than a specified threshold, then the relative latency values are assumed to be showing an increasing trend. In practice, the value of this threshold depends on the deviation in network jitter values, as this will determine how many of the actual latency values are greater than the long-term average.

If PCT is greater than the threshold, then the slope in network latency, L_S, is calculated using two relative latency samples from the short term running average, $L(n)_{STA}$ and $L(i)_{STA}$, as shown in Equation (2). $L(n)_{STA}$ is the most recently received relative latency value, while $L(i)_{STA}$ is the value recorded when the relative latency trends first began to increase, as can be seen in Figure 4. The short-term average, L_{STA}, is employed in order to reduce erroneous calculations due to jitter in the actual relative latency values.

$$L_S = \begin{cases} \frac{L(n)_{STA} - L(i)_{STA}}{t(n)_{STA} - t(i)_{STA}} & if\ PCT \geq threshold \\ 0 & otherwise, \end{cases} \tag{2}$$

where $t(x)$is the time when sample x was received.

The value of L_S is an indicator of the magnitude of congestion on the connecting link between two participants. Once calculated, as can be seen in area 2 of Figure 4, it is then piggybacked on the next available application packet.

3.2 Bandwidth Estimation

Following transmission of the slope request, each participant waits to receive the slope value, L_S, for all logical connections before estimating available bandwidth, rather than reacting to each slope value individually. Thus, the local node can determine where the congestion, if any, is occurring. If all slope values received show congestion, then it is assumed that the upstream bandwidth is saturated. If only some are showing congestion, then other links out on the network are showing congestion. Determining where the congestion is occurring is important to our bandwidth estimation technique, as will be seen in this section.

Existing bandwidth estimation techniques, such as Self Loading Periodic Streams (SloPS) and Trains of Packet Pairs (ToPP), employ a similar latency trend analysis to estimate available link bandwidth [Cheng and Marsic 2001; Jain and Dovrolis 2002; Prasad et al. 2003; Ribeiro et al. 2003]. However, these approaches use an iterative algorithm to determine bandwidth. Due to this, Pathload, which is an implementation of SloPS, takes approximately 15 seconds to estimate bandwidth. Although this results in accurate link capacity estimates, the iterative approaches they employ are not suited to real time DIAs, which have very strict time constraints.

To deal with these limitations, a different estimation approach, which can operate in real time, is employed by the CO algorithm. In order to satisfy the real time constraint, and considering that the scheme is tailored towards the last mile link, here we make the following assumptions. First, our approach assumes that the application under our control is the only application occupying the connection. Second, the estimation approach operates by assuming that the link buffer can be modelled using a queuing system with a deterministic arrival rate and deterministic service rate, also known as a D/D/1 queuing system.

Under these assumptions, if the arrival rate, R_A, at a link is greater than the service rate, R_S, of that link, then messages will be added to a queue and delayed. The system is then said to be unstable, and the queue length and delay will grow indefinitely (assuming an infinite buffer). The increase in delay per packet, L_P is given by Equation (3).

$$L_P = \frac{1}{R_S} - \frac{1}{R_A}. \tag{3}$$

The increase in one-way latency over a second, as measured by one of the recipients connected to the bottleneck link, is given by $R_S L_P$, as the recipient receives $R_S$packets per second. However, the rate of increase in latency is already known, and is given by the value of L_S received from the remote node. The value of R_A is also known, as this is the amount of data the node is transmitting over the

bottleneck link. Therefore, the capacity of the bottleneck node on a link, R_S, can be calculated in terms of $L_S$and R_A, as shown in Equation (4).

$$R_S = R_A - L_S R_A. \tag{4}$$

In order to gain an accurate value of R_S, it is crucial that R_A is set equal to the amount of data being transmitted over the bottleneck link. This is why each node waits to receive all values of L_S from all participants. If all connections on the same link are found to be showing increasing trends, then it is assumed that the upload bandwidth is saturated. In this case, R_A is set equal to the sum of data transmitted to all participants, and $L_S$is set to the maximum L_S value received. Otherwise, R_A is set to the packet transmission rate of the connection showing the increasing trend. These values are then employed along with the slope value in Equation 4 to estimate R_S, the available bandwidth. The available bandwidth per logical connection, AB_i, given N logical connections from the DIA transmitting across bottleneck connection is then:

$$AB_i = \frac{R_s}{N}. \tag{5}$$

3.3 Reaction to Congestion

When congestion occurs, the remote node returns a positive slope value, and the value of $AB_i$is determined. A suitable packet transmission rate is then calculated by dividing the estimated bandwidth value by the required packet size. Under the CO Algorithm design, a minimum packet transmission rate can also be specified. This is the minimum rate below which the consistency of the DIA cannot be maintained to acceptable standards. Once a reduction occurs, latency will begin to decrease for a short period, as the excess data within the buffer at the bottleneck link is scheduled for transmission. Once this data is transmitted, the latency value will return to the normal operating value for that particular network link.

The transmission rate is decoupled from the rate of generation of events within the DIA. This requires that events generated between the transmission of packets be buffered until a packet is available for transmission, at which time multiple events can be aggregated into a single packet. Such an approach is necessary so that the bandwidth requirements of the DIA can be managed by the application, and is not susceptible to the possibly irregular and bursty nature of event generation.

When a new participant joins the DIA, a check is first made to see if the available bandwidth is known for the link on which the new connection is formed. If not, then the participant transmits at their default maximum packet rate. This means that when the DIA first begins, each participant transmits at the maximum transmission rate. However, if the bandwidth value has been previously calculated, then the capacity of the bottleneck link is known. Thus, the bandwidth can be further subdivided between all participants using Equation (5), which further reduces the data generation rates of each connection. Similarly, when a participant leaves the DIA, and the bandwidth value of the link is known, then the data generation rates along each logical connection can be increased.

Optimally, the CO Algorithm should perform the bandwidth estimation once throughout the entire execution of the DIA, with the only modification to packet transmission rates occurring when a participant joins or leaves the DIA. This would reduce the network latency induced by the algorithm, whilst still maximising bandwidth usage. However, a key issue is the behaviour of the algorithm in the presence of transient external congestion. For example, we have noted during our live Internet trials that network latency values are prone to infrequent increases for short periods of time, which typically passes after a few seconds without any algorithm intervention. This can lead to situations where the CO algorithm will interpret the increase in latency as a persistent congestion, and react accordingly. Although judicious choice of control parameters, particularly the long- and short-term window

sizes, can minimize the impact of the transient congestion periods, this is not an ideal solution, as the observed patterns in latency values can vary slightly throughout the day, and across different links.

A more suitable solution is the development of an appropriate recovery scheme, which allows the application to increase its data transmission rates following a reaction to a transient congestion period. The proposed recovery scheme of the CO algorithm acts as follows. If the slope value received, L_S, is zero, then no congestion has been detected on the link. In this case, if the bandwidth value is known, then the bandwidth being utilized by the DIA can be increased, if required. The estimated bandwidth value is increased by fixed amounts (for example, 1 kilobyte), and this value can be evenly distributed between all connections on that link using Equation (5). The rate at which bandwidth utilization increases determines the aggressiveness of the algorithm. A highly aggressive algorithm will maximize bandwidth usage, at the risk of increased network latency. A minimally aggressive algorithm will minimize latency, but may not promptly react to a congestion period, and thus may under utilize available bandwidth. Ideally, the aggressiveness of the recovery scheme for a DIA should be tailored to minimize latency, while at the same time allowing for relatively prompt reaction following a decrease in bandwidth usage. Under our scheme, the level of aggressiveness is controlled by varying the number of slope values received where L_S is equal to 0, before data transmission rates are increased.

In the next section, the performance of the Consistency Optimization algorithm in a series of live and simulation trials is evaluated.

4. ALGORITHM EVALUATION

To test the algorithm, the same test scenario outlined in Section 2 was again employed. The CO Algorithm was implemented in the test DIA. Following a series of initial tests to examine the patterns in network latency and jitter, the parameters of the CO algorithm were set as follows. The threshold value for the PCT threshold was set 0.7, combined with a long-term and short-term running average of 10 and 5 latency samples respectively. A slope request message was transmitted in every 10^{th} packet. If 10 slope values were received showing an absence of congestion, the bandwidth usage of the DIA was increased by 8 kbits/s. These values were found to give the best trade-off between compensating for network jitter values and trend detection. Finally, a minimum packet rate of 5 Packets per Second was specified, meaning that the CO Algorithm will not reduce its transmission rates below this value for each logical connection.

Several tests were carried out at different times of the day throughout the week. Representative results are presented here for afternoon tests (between 12pm and 3pm) carried out on a weekday. Each test lasted 80 minutes, during which players randomly joined and left the simulation (simulated by the main player node creating and deleting logical connections with the sink node), resulting in varying bandwidth requirements across the bottleneck link.

4.1 Impact on Network Conditions

For comparison purposes, the first set of experiments tested the application without the CO algorithm. In this case, a constant packet rate of 30 PPS was transmitted along each logical connection. The varying bandwidth requirements arose due to the varying number of participants. Figure 5 shows the number of players, bandwidth usage, network latency and packet loss, as experienced between the Portlaoise and Carlow node throughout the test scenario.

It is evident from the Figure 5(a) and (b) that the upstream bandwidth of the ADSL link can only support 4 players at 30 PPS, without exceeding the available bandwidth. The impact of exceeding the available bandwidth is clear from the latency and loss values in Figure 5(c) and (d), as both increase dramatically each time available bandwidth is exceeded. Comparing the loss and available bandwidth graphs, bursts of up to 60 packets, or 2 second worth of state data, can be lost when the link is heavily

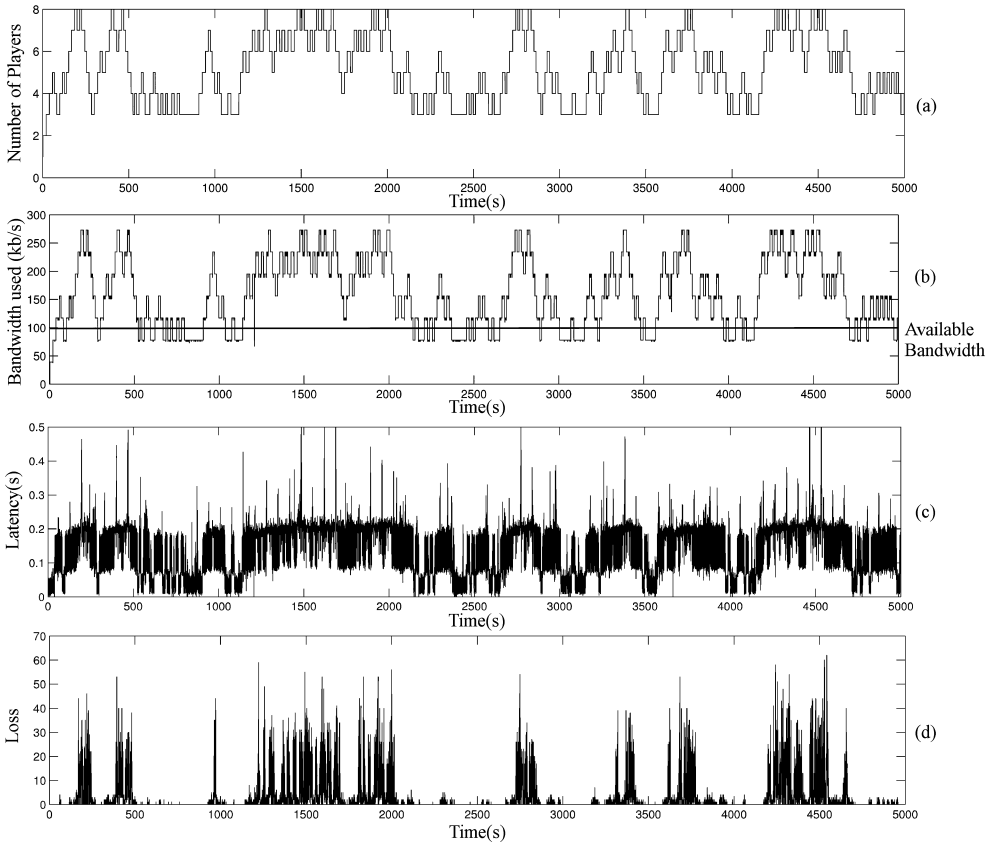

Fig. 5. (a) Number of players (b) Bandwidth utilisation (c) Network Latency (d) Packet Loss as experienced by the main player node when the CO algorithm is not employed.

overloaded. Examination of the latency values demonstrates that during the heavily loaded periods, latency falls between 100 and 400ms. This range exceeds the latency values where the player experience is negatively affected, as identified in other work [Beigbeder et al. 2004; Nichols and Claypool 2004; Yasui et al. 2005].

The next set of experiments conducted employed the same player join and leave pattern. In this case the CO algorithm is employed. The results from this test are presented in Figure 6, and again show the number of players, bandwidth usage, network latency and packet loss, as experienced by the node throughout the 80-minute simulation. Comparing the bandwidth usage results in Figure 5(b) to those in Figure 6(b), the impact of using the CO algorithm is immediately obvious. Throughout the entire 80 minutes of the live trial, bandwidth usage stays below the available bandwidth value, regardless of the number of players involved in the simulation, as seen in Figure 6(a). However, as the available bandwidth value is being estimated from the trends in network latency, the bandwidth usage is close to that of the available bandwidth, with an average bandwidth usage of approximately 85kbit/s. This result also provides validation for the assumptions made when developing the bandwidth estimation scheme as cross traffic from external links has little effect on the performance of the algorithm throughout the 80 minutes of the simulation.

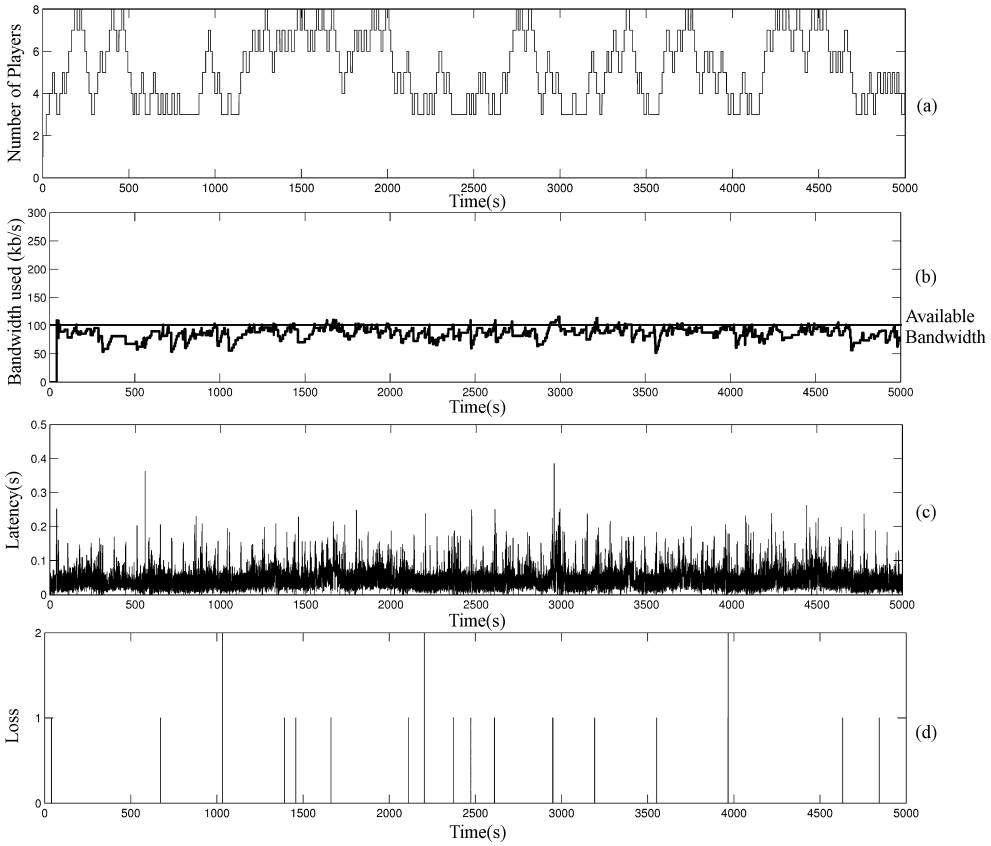

Fig. 6. (a) Number of players (b) Bandwidth utilisation (c) Network Latency (d) Packet Loss as experienced by the main player node when the CO algorithm is employed.

The operation and accuracy of the impending congestion detection scheme used within the CO algorithm is evident from Figure 6(b). It can be seen that every time the actual bandwidth usage of the DIA rises above the available bandwidth, a reduction in the bandwidth usage will occur. A period of gradual increases in bandwidth usage will then follow, as the recovery algorithm operates. The usefulness of the recovery algorithm is also evident at the time period of approximately 500 seconds. Here it can be seen that the algorithm has responded to a rare transient congestion period, resulting in a reduction in bandwidth usage even though the actual bandwidth usage was not above the available value. However, the recovery algorithm allows the application to increase bandwidth usage, and maximize overall bandwidth usage.

The main benefit of using the CO algorithm is evident from the latency and loss graphs presented in Figures 6(c) and (d). Unlike the results presented in Figure 5(c), network latency now remains below 100ms, on average. Throughout the test scenario, the latency values match that of the unloaded period before 600 seconds in the results presented in Figure 2(c). From Figure 6(d), it can be seen how packet loss bursts are now reduced to a maximum of 2, compared to the packet loss burst of up to 60 packets seen in Figure 5(d).

Figure 7 further highlights the operation and performance of the CO algorithm during the live Internet trials. Here, a sample of bandwidth utilization and network latency for the 100-second period

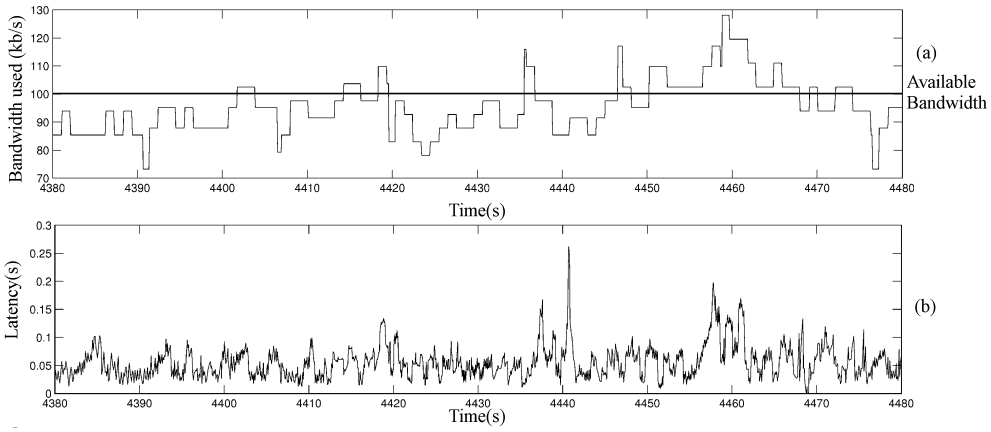

Fig. 7. (a) Bandwidth utilisation (b) Network Latency as experienced by the main player node when the CO algorithm is employed.

between 4380 and 4480 seconds from Figure 6 are highlighted. Between the period of 4380 and 4450 seconds, the recovery algorithm is gradually increasing the bandwidth being utilized by the DIA. As this is less than the available bandwidth in general, network latency remains constant on average. However, between the times of 4450 and 4460, the bandwidth being used by the application is persistently greater than the available bandwidth for an extended period. This causes queuing at the bottleneck link, resulting in an increasing trend in network latency over the same period. At this point, reducing the data transmission rates would improve inconsistency. This increasing trend is detected by the CO algorithm, resulting in a reduction in network latency via a reduction in the data transmission rates of the DIA.

4.2 Impact on Inconsistency

Now, the impact of the varying network latency, packet loss, and packet transmission values shown in the previous section on inconsistency are presented. Figure 8 shows the inconsistency arising for the two scenarios with varying participant numbers, as presented in the previous section. Figure 8(a) shows the number of players throughout the two test scenarios. Figure 8(b) shows spatial inconsistency for the case when the CO algorithm is not used, while Figure 8(c) shows spatial inconsistency for the case where the CO algorithm is used.

The benefits of using the CO algorithm are immediately obvious from the results. In the case of Figure 8(b), as the number of players increases and packet loss and network latency also increase, spatial inconsistency is seen to grow accordingly. Clearly such a scenario would impact negatively on player experience in a DIA. On the other hand, in Figure 8(c), the application can gracefully accommodate numerous players, while at the same time minimizing inconsistency. As the number of participants in the DIA varies, the amount of inconsistency also varies as the algorithm constantly adjusts the packet transmission rates. However, as bandwidth usage is maximized, the consistency values are optimised, given the available bandwidth resources on the bottleneck link.

The variable inconsistency due to varying number of users is further highlighted in Figure 9, which presents a magnified view of 300 seconds of the inconsistency values between 1800 and 2100 seconds. For clarity, the packet transmission rate per logical connection and the number of simulated players involved in the DIA during the same time period are also presented. From these results, it can be seen how the packet transmission rate is constantly modified to accommodate the varying

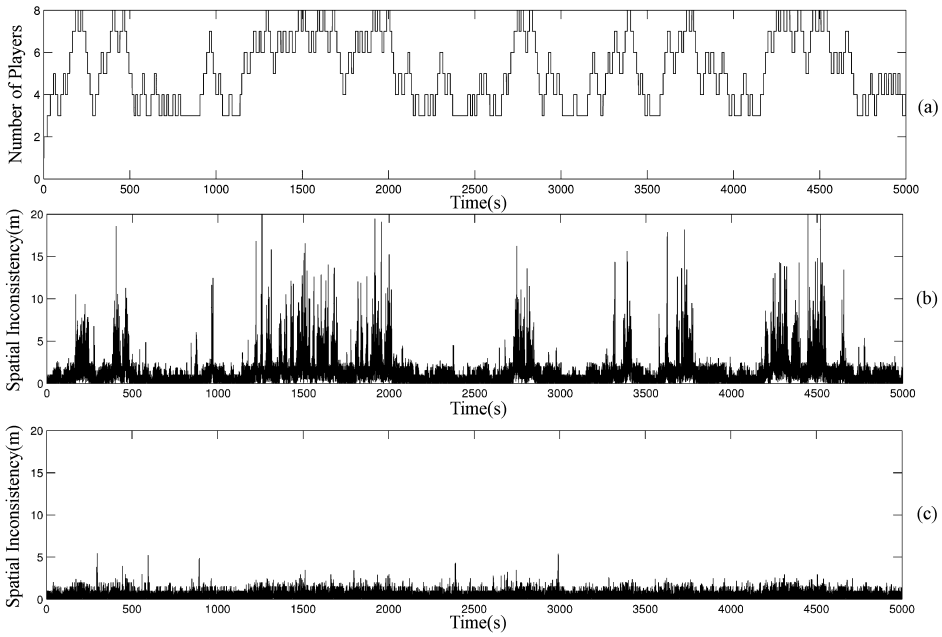

Fig. 8. (a) Number of players (b) Spatial Inconsistency without the CO algorithm (c) Spatial Inconsistency with the CO algorithm.

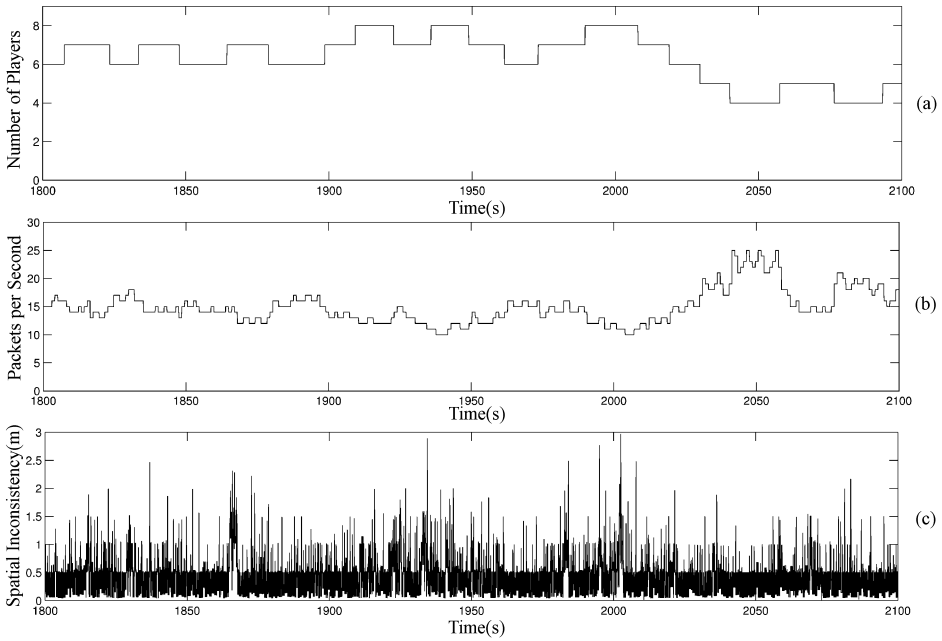

Fig. 9. (a) Number of Players (b) Packet Transmission rate (b) Spatial Inconsistency with the CO algorithm.

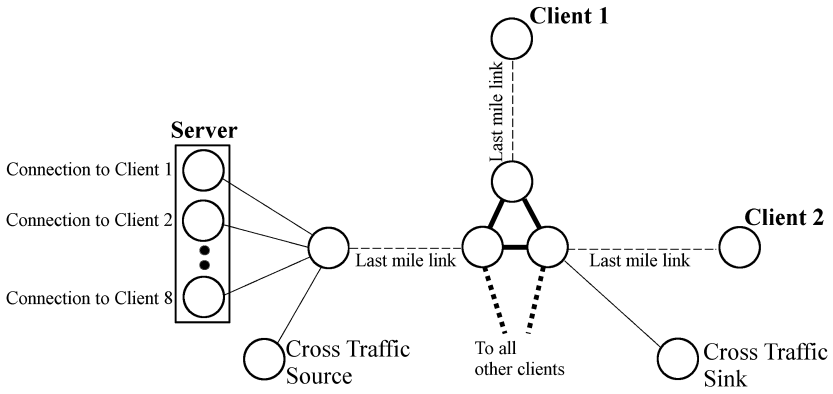

Fig. 10. Network Topology used in NS2 simulations.

number of users. Although this results in worse inconsistency as the number of players increases and the packet rate decreases, it is still an improvement on the consistency experienced without the CO algorithm. In this case, the algorithm is only reducing data reduction rates when the reduction will provide an actual reduction in inconsistency. It is also of interest to note that when 8 players are involved in the DIA, the packet transmission rate is between 10 and 12 PPS. This value is approximately equal to the optimal packet transmission rate that was identified in the results presented in Figure 2.

4.3 Impact of Cross Traffic

While the previous results have demonstrated that the assumptions made in the development of the CO Algorithm are reasonable, given the practical usage scenarios of modern day DIAs, it is still important to consider the effect of competing external flows, particularly TCP flows, on the performance of the CO Algorithm. In general, if the CO Algorithm and a constantly transmitting TCP flow are operating on the same bottleneck link, then the TCP flow will increase its bandwidth share at the expense of the flows managed by the CO Algorithm. This occurs as the CO Algorithm responds to latency, while TCP will respond primarily to loss. Therefore, once congestion begins to arise, the CO Algorithm will be the first to respond to the congestion, and thus will reduce its data transmission rates first. The TCP flow will then have more available bandwidth, so will further increase its transmission rates, leading to a negative feedback loop situation for the CO Algorithm. This is an issue for all purely delay-based congestion control mechanisms [Leith et al. 2007].

However, consider again the practical usage scenarios of a modern DIA. In general, if other TCP traffic is present on the same link as a DIA employing the CO Algorithm, it is likely to be "bursty" Web-browsing traffic employing the Hyper Text Transfer Protocol (HTTP). Therefore, the performance of the CO Algorithm in the presence of this type of traffic is now considered. To carry out this study, the NS2 network simulation tool was employed. Within NS2, a network simulating that of the live test environment outlined in Section 2 was created. An overview of this simulated network is shown in Figure 10. Within this simulated network, the Server maintained a separate logical connection to eight clients. The data transmission rates along of each logical connection were controlled by the CO Algorithm. As with the live trial scenarios, the maximum and minimum packet transmission rate was set to 30 and 5 packets per second respectively, and packet size was set to 122 bytes. Similarly, the bandwidth of the last mile links were set to 100 kbit/s and 1024 kbit/s in the upstream and downstream direction, respectively. The queue size at each last mile link was set to 50 packets. To ensure realistic

Table I. Latency Trends Used within the NS2 Simulations

Site Name	Mean Latency
google.ie(morning)	27.39 ms
google.ie(evening)	27.75 ms
google.fr(morning)	58.43 ms
google.fr(afternoon)	55.2 ms
google.fr(evening)	55.4 ms
yahoo.com(morning)	113.87 ms
yahoo.com(afternoon)	114.38 ms
yahoo.com(evening)	113.78 ms

network latency values for each logical connection, eight separate latency trends were collected at different times of day to live locations from the link at Portlaoise, Ireland. The characteristics of these latency trends are highlighted in Table I. Each logical connection employed one of these latency trends throughout the simulation.

To simulate cross traffic across the last mile link from the Server, a Cross Traffic Source and Sink node were added to the simulated network. In order to generate realistic network traffic typical of normal web browsing activity, the PackMIME traffic generator module available within the NS2 simulation framework was employed [Cao et al. 2001]. The PackMIME module generates traffic typical of browsing behaviour due to the sending and receiving of HTTP requests and responses. To use the PackMIME generator, the user specifies the average number of new requests to generate per second and a random number generator seed. The PackMIME module then uses these input parameters to randomly generate requests, and the responses to those requests, at intervals with a mean equal to the user specified value.

A PackMIME client was attached to the Cross Traffic Source node. Its responsibility was to generate HTTP requests at the specified intervals. These intervals were varied per simulation. A PackMIME server was attached to the Cross Traffic Sink node. Its responsibility was to generate HTTP responses as it received HTTP requests. The Maximum Segment Size of the TCP segments transmitted by both Source and Sink nodes was set to 1500 bytes including header information.

A number of experiments were conducted using this simulation environment to investigate the impact of cross traffic on the performance of the CO Algorithm. Within each experiment, a new participant joined the simulation every 3 seconds, up to a maximum of 8 participants. Each experiment lasted 200 seconds. The PackMIME nodes generated cross traffic throughout the entire simulation. The amount of cross traffic generated was varied per experiment. In particular, request intervals of 0.1, 0.3, 0.5, 1, and 5 HTTP requests per second were simulated. These represent light to heavy web browsing activity. The aggressiveness of the CO Algorithm was also varied per simulation. As discussed in Section 3.3, aggressiveness is determined by varying the number of slope values received that show an absence of congestion before data transmission rates are increased. Intervals of 10, 5, and 1 received slope values per increase were simulated. For clarity, these are referred to as low, medium and high aggressiveness, respectively, in the results presented below.

Each experiment was repeated five times for each HTTP request interval and CO Algorithm aggressiveness value. In each experiment, a different seed was employed for the random number generator used by the PackMIME nodes. The average upstream bandwidth collectively utilized by the Server nodes in each test scenario was then determined, and is presented in Table II.

The impact of cross traffic on the performance of the CO Algorithm is clear from Table II. For the low level of aggressiveness used within the live test scenarios, the CO Algorithm can maintain a reasonably high level of bandwidth usage in the presence of a light to medium amount of cross traffic. However,

Table II. Impact of Aggressiveness on Upstream Bandwidth Usage in Varying Cross Traffic Conditions

HTTP Requests/s	CO Algorithm Aggressivness		
	Low	Medium	High
0.1 (Light)	80.46 kbit/s	80.79 kbit/s	85.25 kbit/s
0.3 (Light)	72.51 kbit/s	79.54 kbit/s	85.65 kbit/s
0.5 (Medium)	58.79 kbit/s	70.51 kbit/s	83.64 kbit/s
1 (Heavy)	47.62 kbit/s	62.49 kbit/s	72.62 kbit/s
5 (Heavy)	39.76 kbit/s	42.51 kbit/s	60.66 kbit/s

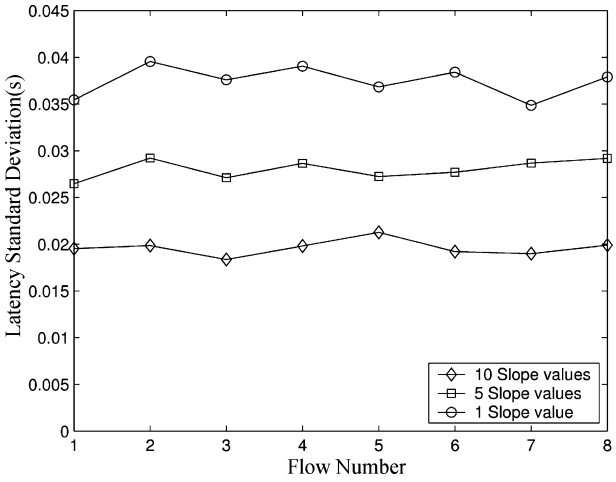

Fig. 11. Impact of CO Algorithm aggressiveness on standard deviation in network latency.

in the case of 5 HTTP Requests/s, which represents heavy cross traffic, the performance of the CO Algorithm begins to suffer. For example, for low and medium aggressiveness, the CO Algorithm forces the DIA to reduce its data transmission rates to the minimum packet transmission rate of 5 packets per second per logical connection, resulting in a bandwidth utilization of approximately 40 kbit/s in both scenarios.

By increasing the aggressiveness of the CO Algorithm, performance can be improved. For example, for the maximum aggressiveness tested here, the average bandwidth usage is approximately 85 kbit/s for light to medium cross traffic. This is the same average bandwidth usage reported within the live trial scenario results presented in Section 4, where no cross traffic was present on the last mile link (but was present on external links). As the level of cross traffic increases, performance does begin to suffer, but not to the extent of the less aggressive test scenarios.

While the results presented in Table II suggest that the low level of aggressiveness used within the live trial scenarios was unsuitable, consider the results presented in Figure 11. Here, the standard deviation in network latency along each of the eight logical connections from the Server within the NS2 simulation is shown for varying levels of aggressiveness of the CO Algorithm. To collect these results, the NS2 simulations described previously were repeated. On this occasion, however, no cross traffic was present.

It can be clearly seen from Figure 11 that as the aggressiveness of the CO Algorithm increases, it negatively impacts on the network latency experienced by the DIA. The standard deviation in network latency doubles when the aggressiveness of the CO Algorithm increases from low to high aggressiveness. This arises as, when the aggressiveness of the algorithm is increased, packet

transmission rates are increased rapidly following a reduction in response to congestion. This means that the DIA frequently exceeds available bandwidth on the last mile link, resulting in a subsequent increase in fluctuation of network latency.

The results presented in this section highlight the delicate balance between network latency and algorithm aggressiveness, as first discussed in Section 3.3. A high level of aggressiveness can allow the algorithm to maintain a high level of bandwidth usage in response to the presence of competing flows on the same link. However, it can result in an increase in fluctuation in network latency for the DIA. A question then arises regarding which of these elements is of most importance to maintaining consistency within the DIA employing the CO Algorithm. This issue, and a more in-depth study of the effect of cross traffic on the CO Algorithm, remain the focus of future work with the CO Algorithm.

5. RELATED WORK

While much of the work related to adaptive data transmission schemes for DIAs has focused on adaptation based on application-level parameters, such as player behavior [Cai et al. 1999; Zhang et al. 2004; Chen 2005; McCoy et al. 2007], there has been work in the area of network based adaptation. An earlier example of such a scheme is Kravets et al. [1998]. The approach detailed in this work operates by dynamically adjusting transport level reliability requirements of a DIA to minimise the volume of data transmission during periods of congestion. Another variable reliability approach is the Interactivity Loss Avoidance (ILA) mechanism [Palazzi et al. 2005]. This mechanism operates on the premise that fast-paced games can tolerate a certain degree of loss without affecting user experience. Based on this assumption, the ILA approach drops game-related events that are deemed unimportant to the game state in order to avoid impending congestion. Our approach differs to these as it attempts to deal with congestion via data transmission management at the application level rather than relying on transport or network level congestion management.

Other mechanisms that adopt a similar data transmission management to ours are the Switchboard architecture and the Network Aware Bandwidth Adaptation (NABA) mechanism [Trefftz et al. 2003; Yu et al. 2007]. The Switchboard architecture is a Client/Server approach that attempts to adjust system parameters, such as world detail and message update rate, to satisfy both user and system designer constraints. In particular, message update rates are managed via a subscription mechanism. Each world variable is associated with a multicast address that transmits updates at varying rates. To reduce the number of updates a client receives from a server, the client can subscribe to a multicast address with a slower update rate. While this approach can be used to manage data transmission rates, it requires user intervention at runtime in order to choose the optimal rate best suited to the underlying network, unlike our automated approach.

The NABA mechanism represents an approach that is closest to our own. In this work, the authors present an approach that attempts to find the trade-off between state inconsistency and bandwidth usage when an Information Management technique, known as dead reckoning, is employed. Dead reckoning reduces data transmission rates by providing a controlled level of spatial inconsistency [IEEE 1995; Delaney et al. 2006]. It does this by only transmitting a synchronization message when the difference between the actual and remote entity position exceeds a predefined error threshold, known as the spatial error threshold. In the absence of state information, remote participants predict the position using previously received entity dynamics information. The NABA technique operates by tracking the number of updates arising from the use of dead reckoning, and tailoring the spatial error threshold so that the number of updates lies within a specific bandwidth value. Where the CO algorithm differs from this approach is that our algorithm will attempt to estimate the available bandwidth during run time, and is suited to more general applications as it uses constant packet generation rather than dead reckoning. The NABA mechanism, on the other hand, does not provide any bandwidth estimation facility.

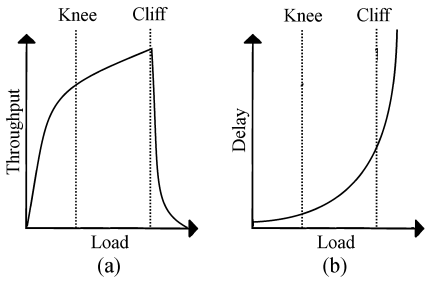

Fig. 12. Traditional adaptive control algorithms attempt to stay left of the cliff. However, to minimise delay, a protocol needs to stay left of the knee.

The recent surge in popularity of Massively Multiplayer games, such as World of Warcraft, has seen an increase in research activity towards adaptive mechanisms for managing load and scale in such large-scale simulations, which typically feature thousands of simultaneous players [Lee and Lee 2003; Lu et al. 2006]. Early examples of these architectures include RING [Funkhouser 1995] and NetEffect [Das et al. 1997] while more recent adaptive approaches include Matrix [Balan et al. 2005]. Although our work is related to these, the adaptation approach differs. While these approaches assume the presence of some extra resources (such as another available server node) that can be called upon to accommodate increased load, our approach attempts to adapt the application characteristics to accommodate the increased load within the same available resources.

One cannot discuss any adaptive data transmission algorithm without making reference to the large body of work has focused on adaptive transport level protocols for non traditional networked based applications, such as streaming video [Rejaie et al. 1999; Mukherjee and Brecht 2000; Krasic et al. 2001; Aboobaker et al. 2002]. Of these, the most pertinent in relation to our own mechanism are the class of protocols referred to as "Slowly Responsive" congestion control algorithms [Bansal et al. 2001; Widmer et al. 2001]. Such algorithms have been developed in response to the widespread deployment of applications that require congestion control, but cannot tolerate the large and sudden changes in data transmission rate that ensues when TCP reacts to congestion.

The current state of the art in such protocols is the TFRC (TCP-Friendly Rate Control) protocol [Floyd et al. 2000]. The goal of TFRC is to provide stable data generation rates, whilst still being fair to coexistent TCP streams. This is referred to as "TCP-Friendliness." The TFRC protocol achieves this by using an equation to model how a TCP flow would react to the network congestion indicators the TFRC flow is experiencing, and adjusting its data transmission rates accordingly.

There are two issues that affect the use of such protocols with a DIA. First, as they operate at the transport level, they cannot exploit application semantics in order to manage data transmission rates. Second is their TCP heritage and thus their effect on bottleneck link queue dynamics. As they respond primarily to packet loss, they are designed to maximize throughput, and ensure that the load on the bottleneck link is as close to the maximum as possible, the "cliff" in Figure 12(a), without exceeding it. Due to this, such protocols tend to maximize queue usage, and as can be seen from the latency profile in Figure 12(b), cause an increase in network latency [Goel et al. 2008].

This issue is exacerbated in Slowly Responsive algorithms, such as TFRC. As TFRC is designed to respond gradually to congestion indicators, it tends to converge to a state of persistent queue usage [Floyd et al. 2000]. Such behaviour could make TFRC unsuitable for the strict timeliness requirements of a DIA. The main difference between our approach, and that taken in TFRC, is that our mechanism attempts to track the position of the "knee," as shown in Figure 12(b), where network latency and queue usage are minimised.

6. CONCLUDING DISCUSSION

In this work, the issue of consistency and Information Management (IM) techniques in Distributed Interactive Applications (DIAs) was considered. As the majority of existing IM techniques are controlled by application or user-centric parameters, they risk creating more inconsistency issues than they resolve. In order to minimize inconsistency, the performance of the network needs to be constantly considered. Based on this, a novel information management technique for adaptive data transmission in DIAs, which we refer to as the Consistency Optimization (CO) algorithm, was proposed. This approach operates with minimal extra data requirements, and is designed to optimize the consistency of the state data transmitted by a DIA. It does this by continually measuring the impact of the DIA on the underlying network, and estimating the available bandwidth when congestion arises. The data transmission characteristics of the DIA can then be modified to match the estimated value.

Results from live Internet trials demonstrate that the CO algorithm can accurately track bandwidth values in a typical current day broadband environment. In doing so, the data generation rates of a DIA can be modified to suit the available bandwidth. As the number of participants involved in the DIA increases and decreases, the algorithm can quickly modify data generation rates to accommodate the extra load, with little impact on network latency or packet loss. In this way, the consistency arising during the execution of the DIA can be optimized.

Although the CO algorithm operates using network-centric control parameters, it can easily be combined with existing IM mechanisms that employ system-or user-centric control parameters as it operates at the application level. By doing this, consistency can be further optimized. For example, consider the popular IM technique known as dead reckoning. Consider a situation where the dead reckoning algorithm may be employing a predetermined threshold that under utilizes the available bandwidth. However, the threshold value is perceptually acceptable to the end-user. In this case, there is little point in transmitting extra position and velocity information in every packet, as it will make no difference to the end user's experience. In such a situation, the CO algorithm could use dead reckoning to remove redundant spatial data from packets that will be transmitted over the network. In its place, other types of data, such as voice data, can be transmitted. In this way, the available bandwidth usage could still be maximized by the CO algorithm, and the consistency of other types of data could be improved. Future work will examine the usefulness of such a scheme in the context of our algorithm.

Other previous work has examined how existing bandwidth estimation schemes can be expressed and compared in terms of *service curves* [Liebeherr et al. 2007]. A service curve describes the available service at a network link. Using Network Calculus, multiple service curves can be combined to describe the end-to-end bandwidth characteristics from a path with multiple links. Future work will examine how the bandwidth estimation approach used by the CO Algorithm can be interpreted using service curves. By doing this, the relationship between our bandwidth estimation approach and existing bandwidth estimation approaches could be thoroughly examined, and the performance of our technique in the presence of other flows on the same bottleneck link could also be analyzed.

ACKNOWLEDGMENTS

Thanks to the staff at IT Carlow and Karl Jeacle. Thanks also to the reviewers for their helpful and constructive comments.

REFERENCES

ABOOBAKER, N., CHANADY, D., GERLA, M., AND SANADIDI, M. Y. 2002. Streaming media congestion control using bandwidth estimation. In *Proceedings of the 5th IFIPI/IEEE Internatioml Conference on Management of Multimedia Networks and Services: Management of Multimedia on the Internet*. 89–100.

BALAN, R., EBLING, M., CASTRO, P., AND MISRA, A. 2005. Matrix: Adaptive middleware for distributed multiplayer games. In *Proceedings of the 6th ACM/IFIP/USENIX International Middleware Conference (Middleware)*. 390–401.

BANSAL, D., BALAKRISHNAN, H., FLOYD, S., AND SHENKER, S. 2001. Dynamic behavior of slowly-responsive congestion contro! algorithms. In *Proceedings of the ACM SIGCOOM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. 253–274.

BEIGBEDER, T., COUGHLAN, R., LUSHER, C., PLUNKETT, J., AGU, E., AND CLAYPOOL, M. 2004. The effects of loss and latency on user performance in Unreal Tournament 2003. In *Proceedings of the ACM SIGCOMM Workshops on Network and System Support for Games (NetGames'04)*. 144–151.

BERNIER, Y. 2001. Latency compensating methods in client/server in-game protocol design and optimization. In *Proceedings of the Game Developers Conference*.

CAI, W., LEE,F., AND CHEN, L. 1999. An auto-adaptive dead reckoning algorithm for distributed interactive simulation. In *Proceedings of the 13th Workshop on Parallel and Distributed Simulation*. 82–89.

CHEN, L. 2005. An adaptive consistency maintenance appproach for replicated continuous applications. In *Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05)*. 795–801.

CHENG, L. AND MARSIC, I. 2001. Bandwidth Measurement in XDSL networks. In *Proceedings of the Joint 4th IEEE International Conference on ATM (ICATM '01) and High Speed Intelligent Internet Symposium*. 222–226.

CLAYPOOL, M. AND CLAYPOOL, K. 2006. Latency and player actions in online games. *Comm. ACM, 49*, 11, 40–45.

CLAYPOOL, M., LAPOINT, D., AND WINSLOW, J. 2003. Network analysis of Counter-Strike and Starcraft. In *Proceedings of the IEEE International Performance, Computing and Communications Conference*. 261–268.

DAS, T., SINGH, G., MITCHELL, A., KUMAR, P., AND MCGEE, K. 1997. Neteffect: A network architecture for large-scale multi-user virtual worlds. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*.·157–163.

DELANEY, D., WARD, T., AND S. MCLOONE, S. 2006a. On consistency and network latency in distributed interactive applications: A survey - Part I. *Presence: Teleop. Virt. Environ. 15*, 2, 218–234.

DELANEY, D., WARD, T., AND S. MCLOONE, S. 2006b. On consistency and network latency in distributed interactive applications: A survey - Part II. *Presence: Teleop. Virt. Environ. 15*, 4, 465–482.

DOURISH, P. 1995. The parting of the ways: Divergence, data management and collaborative work. In *Proceedings of the 4th European Conference on Computer-Supported Cooperative Work*. 213–229.

DUBE, P., LIU, Z., SAHU, S., AND SILBER, J. 2005. Last mile problem in overlay design. In *Proceedings of the IEEE Global Telecommunications Conference*. 920–925.

FENG, W., CHANG, F., AND WALPOLE, J. 2002. Provisioning on-line games: A traffic analysis of a busy Counter-Strike server. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*. 151–156.

FLOYD, S., HANDLEY, M., PADHYE, J., AND WIDMER, J. 2000. Equation-based congestion control for unicast applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. 43–56.

FRECON, E. AND STENIUS, M. 1998. Dive: A scaleable network architecture for distributed virtual environments. *Dist. Syst. Engin. J. 5*, 3, 91–100.

FUNKHOUSER, T. 1995. Ring: A client-server system for multi-user virtual environments. In *Proceedings of the Symposium on Interactive 3D Graphics*. 85–92.

GOEL, A., KRASIC, C., AND WALPOLE, J. 2008. Low latency adaptive streaming over TCP. *Trans. Multimedia Comput. Commun. Appl., 4*, 3.

IEEE. 1995. IEEE standard for distributed interactive simulation - application protocols. IEEE Standard 1278.1-1995:

JAIN, M. AND DOVROLIS, C. 2002. Pathload: A measurement tool for end-to-end available bandwidth. In *Proceedings of the Passive and Active Measurements (PAM) Workshop*. 14–25.

JEHAES, T., DE VLEESCHAUWER, D., COPPENS, T., VAN DOORSELAER, B., DECKERS, E., NAUDTS, W., SPRUYT, K., AND SMETS, R. 2003. Access network delay in networked games. In *Proceedings of the 2nd Workshop on Network and System Support for Games*. 63–71.

KRASIC, C., LI, K., AND WALPOLE, J. 2001. The case for streaming multimedia with Tcp. In *Proceedings of 8th International Workshop on Interactive Distributed Multimedia Systems*. 213–218.

KRAVETS, R., CALVERT, K., AND SCHWAN, K. 1998. Payoff-based communication adaptation based on network service availability. In *Proceedings of the IEEE Conference on Multimedia Computing and Systems*. 33–42.

LANG, T., BRANCH, P., AND ARMITAGE, G. 2004. A synthetic traffic model for Quake3. In *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*. 233–238.

LEE, K. AND LEE, D. 2003. A scalable dynamic load distribution scheme for multi-server distributed virtual environment systems with highly-skewed user distribution. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. 160–168.

LEITH, D. J., HEFFNER, J., SHORTEN, R. N., AND MCCULLAGH, G. D. 2007. Delay based AIMD congestion control. In *Proceedings of the Workshop on Protocols for Fast Long Distance Networks (PFLDNet '07)*. 73–78.

LIEBEHERR, J., FIDLER, M., AND VALAEE, S. 2007. A min-plus system interpretation of bandwidth estimation. In *Proceedings of 26th IEEE International Conference on Computer Communications (INFOCOM '07)*. 1127–1135.

LU, F., PARKIN, S., AND MORGAN, G. 2006. Load Balancing for massively multiplayer online games. In *Proceedings of the 5th ACM SIGCOMM Workshop on Network and System Support for Games*. 1–11.

MACEDONIA, M. R. AND ZYDA, M. J. 1997. A taxonomy for networked virtual environments. *IEEE Multimedia 4*, 1,48–56.

MARSHALL, D., MCLOONE, S., ROBERTS, D., DELANEY, D., AND WARD, T. 2006a. Exploring the effect of curvature on the consistency of dead reckoned paths for different error threshold metrics. In *Proceedings of the 10th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DSRT'06)*. 77–84.

MARSHALL, D., MCLOONE, S., WARD, T., AND DELANEY, D. 2006b. Does reducing packet transmission rates help to improve consistency in distributed interactive applications? In *Proceedings of 9th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games*. 88–92.

MARSHALL, D., MOONEY, B., MCLOONE, S., AND WARD, T. 2007. An unobtrusive method for tracking network latency in online games. In *Proceedings of the China Ireland International Conference on Information and Communications Technologies*. 54–61.

MCCOY, A., WARD, T., MCLOONE, S., AND DELANEY, D. 2007. Multistep-ahead neural-network predictors for network traffic reduction in distributed interactive applications. *ACM Trans. Model. Comput. Simul. 17*, 4.

MCGOVERN, P., MURPHY, S., AND MURPHY, L. 2006. Protection against link adaptation for VoWLAN. In *Proceedings of the 15th IST Mobile an Wireless Communications Summit*.

MUKHERJEE, B. AND BRECHT, T. 2000. Time-lined TCP for the TCP-friendly delivery of streaming media. In *Proceedings of the International Conference on Network Protocols*. 165–176.

NICHOLS, J. AND CLAYPOOL, M. 2004. The effects of latency on Online Madden NFL Football. In *Proceedings of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '04)*. 146–151.

PALAZZI, C., FERRETTI, S., CACCIAGUERRA, S., AND ROCCETTI, M. 2005. A RIO-like technique for interactivity loss-avoidance in fast-paced multiplayer online games. *Comput. in Entertain. 3*, 2, 3–3.

PRASAD, R., DOVROLIS, C., MURRAY, M., AND CLAFFY, K. 2003. Bandwidth estimation: Metrics, measurement techniques, and tools. *IEEE Netw. 17*, 6, 27–35.

REJAIE, R., HANDLEY, M., AND ESTRIN, D. 1999. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. I*n Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies*. 1337–1345.

RIBEIRO, V., RIEDI, R., BARANIUK, R., NAVRATIL, J., AND COTTRELL, L. 2003. Pathchirp: Efficient available bandwidth estimation for network paths. In *Proceedings of the Passive and Active Measurements (PAM) Workshop*.

ROEHLE, B. 1997. Channeling the data flood. *IEEE Spectrum 34*, 3, 32–38.

SINGHAL, S. AND ZYDA, M. 1999. *Networked Virtual Environments: Design and Implementation*. ACM Press/Addison-Wesley Publishing Co., New York.

TREFFTZ, H., MARSIC, I. AND ZYDA, M. 2003. Handling heterogeneity in networked virtual environments. *Presence: Teleop. Virt. Environ. 12*, 1, 37–51.

VAGHI, I., GREENHALGH, C., AND BENFORD, S. 1999. Coping with Inconsistency due to network delays in collaborative virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. 42–49.

WIDMER, J., DENDA, R., AND MAUVE, M. 2001. A survey on TCP-friendly congestion control. *IEEE Netw. 15*, 3, 28–37.

XUE, L., ORGUN,M., AND ZHANG, K. 2002. A user-centred consistency model in real-time collaborative editing systems. In *Proceedings of the 4th International Workshop onDistributed Communities on the Web*. 135–154.

YASUI, T., ISHIBASHI, Y., AND IKEDO, T. 2005. Influences of network latency and packet loss on consistency in networked racing games. In *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games*. 1–8.

YU, Y., LI, Z., SHI, L., CHEN, Y., AND XU, H. 2007. Network-aware state update for large scale mobile games. In *Proceedings of the 16th International Conference on Computer Communications and Networks*. 563–56.

ZHANG, X., GRACANIN, D. AND DUNCAN, T. 2004. Evaluation of a pre-reckoning algorithm for distributer virtual environments. In *Proceedings of the 10th International Conference on Parallel and Distributed System*. 445–452.

Received July 2008; revised February 2009; April 2009; accepted June 2009