Bounding Inconsistency Using a Novel Threshold Metric for Dead Reckoning Update Packet Generation

Dave Roberts[1]
Rob Aspin
Centre for Virtual Environments, Business House
University of Salford, Salford M5 4WT, UK
d.j.roberts@salford.ac.uk

Damien Marshall[1]
Seamus McLoone
Declan Delaney
Tomas Ward
National University of Ireland Maynooth
Maynooth, Co. Kildare, Ireland

Human-to-human interaction across distributed applications requires that sufficient consistency be maintained among participants in the face of network characteristics such as latency and limited bandwidth. The level of inconsistency arising from the network is proportional to the network delay, and thus a function of bandwidth consumption. Distributed simulation has often used a bandwidth reduction technique known as dead reckoning that combines approximation and estimation in the communication of entity movement to reduce network traffic, and thus improve consistency. However, unless carefully tuned to application and network characteristics, such an approach can introduce more inconsistency than it avoids. The key tuning metric is the distance threshold. This paper questions the suitability of the standard distance threshold as a metric for use in the dead reckoning scheme. Using a model relating entity path curvature and inconsistency, a major performance related limitation of the distance threshold technique is highlighted. We then propose an alternative time–space threshold criterion. The time–space threshold is demonstrated, through simulation, to perform better for low curvature movement. However, it too has a limitation. Based on this, we further propose a novel hybrid scheme. Through simulation and live trials, this scheme is shown to perform well across a range of curvature values, and places bounds on both the spatial and absolute inconsistency arising from dead reckoning.

Keywords: Key words text

1. Introduction

A Distributed Interactive Application (DIA) allows participants connected by a computer network to communicate in a virtual world. They leverage the latest developments in communication and graphical technology to enhance collaborative activities across distributed teams, and offer many advantages in today's globalized socio-economic information culture. Applications that qualify as DIAs include collaborative virtual environments, military simulations, video conferencing, collaborative whiteboards and networked games [1–3].

In a DIA such as a collaborative virtual environment, a participant controls a virtual entity, which is the virtual representation of the participant. This virtual entity has a

SIMULATION, Vol. 84, Issue 5, May 2008 239–256

DOI: 10.1177/0037549708092221

1. Joint first authors

number of state variables, which include position and rotation. Communication between participants is achieved by sharing these state variables using synchronization messages. These synchronization messages typically contain the most recent information about the entity's state. They are transmitted periodically across the connecting network, and update the remote state of the virtual entity, which is the state replicated on other participants' computers.

During transmission, each synchronization message is subjected to the limitations of the network connecting participants [4, 5]. The most significant limitations are network latency and network jitter. Network latency is the time taken from start of exchange of a synchronization message at the application layer of one participating node to the end of exchange of the same message at the application layer of a second participating node. Jitter is the variation of latency with time. Latency and jitter can be attributed to a number of aspects such as speed of light delays, transmission delays due to the speed of the communications link and queuing and processing at nodes within the network [6]. The two latter issues are related to the bandwidth of the network. Network bandwidth is a measure of maximum throughput of traffic on the network. If the traffic exceeds the bandwidth, then data will need to be buffered or dropped until the flow of data decreases. This buffering can occur on a dedicated network node, such as a router, or on the computer hosting the application itself. Latency and jitter, and loss of data, will increase every time the bandwidth is exceeded.

Due to the limitations of the network, each synchronization message is not delivered to its intended recipient for a time period at least equal to network latency. Applying these updates naively to the virtual world upon receipt could result in different views of the virtual world for each participant. In this case, each participant is said to have an inconsistent view of the shared world. A sufficient level of consistency is required to maintain the shared sense of space, time and presence necessary for fruitful collaboration between participants [7].

There is a clear link between consistency and DIA synchronization message transmission rates. If the rate exceeds network bandwidth, then consistency will be affected by increased network latency due to overloaded network hardware. To deal with this issue, numerous techniques and approaches have been researched and developed, which optimize the use of network bandwidth and assist the maintenance of consistency. These can be subdivided into three general categories.

1. *Information Management Techniques* These all optimize the bandwidth usage by reducing the amount of information transmitted across the network. Examples include predictive contract agreement mechanisms, relevance filtering, packet bundling and packet compression.
2. *Time Management Techniques* These techniques attempt to preserve some rules of time that can get broken when state is communicated through discrete updates over a network. They include two subcategories: those that manage consistency such as total ordering, causal ordering and wall clock synchronization; and those that hide some perceivable effect such as delayed consistency, time warp and local perception filtering.
3. *Software and Hardware Architecture Techniques* These all aim to improve the efficiency of processing or disseminating information. Examples include QoS, protocols, network architectures and software design.

These techniques and approaches have been summarized and described by previous authors [7–11].

The focus of this work is on one information management technique widely adopted in distributed simulation, known as dead reckoning [12]. Predictive schemes, such as dead reckoning, attempt to reduce inconsistency by reducing latency due to network saturation. To fully understand the operation of such a technique, it is first necessary to briefly review some of the various types of inconsistency considered in the literature [4, 5, 13]. One of the most popular is remote spatial inconsistency. This measures the spatial difference between a virtual entity's local position and that of its remote representation. However, it does not consider time. Temporal inconsistency measures time taken to receive an update following its transmission. It does not consider the effect of the update on the state of the virtual world. For example, an update may arrive soon after its transmission, so temporal inconsistency is low. However, the update may be applied out of order, so it still induces inconsistency. Remote time–space inconsistency combines both a temporal and spatial inconsistency measure into a single value. It considers the length of time the spatial inconsistency persists, as well as the spatial inconsistency itself [14, 15]. In this work, it is referred to as absolute inconsistency.

Dead reckoning operates by providing a controlled level of spatial inconsistency in order to gain a decrease in temporal inconsistency via a reduction in network traffic. It does this by maintaining a model of the actual entity position, the behavior of which is determined by a parametric model of the actual entity position. The difference between the actual position and the model position is continually calculated. This gives a measure of local spatial inconsistency. When this value exceeds a predefined error threshold, known as the spatial error threshold, the parameters describing the parametric model are updated to reflect the most current behavior of the actual entity. A synchronization message containing these parameters is then transmitted over the connecting network to interested remote participants. Upon receipt, each receiving participant then uses this data to update the remote position of

the modeled entity, and to predict the behavior of a remote entity until the next update. A key aspect in the generation of synchronization messages is the value of the spatial threshold. A small threshold value tends to increase the number of messages.

This work questions the use of the standard spatial threshold value as a suitable criterion for generating update packets in dead reckoning. It is demonstrated how use of the spatial threshold does not necessarily provide the best tradeoff between absolute and temporal inconsistency as it can allow a persisting spatial inconsistency over an extended period of time, which could potentially allow unbounded absolute inconsistency. The root cause of this is investigated and a method for capturing and quantifying it using a measure of absolute inconsistency is illustrated. Based on this analysis, we propose an alternative criterion, known as the time–space threshold, which is based on a measurement of both time and distance. Comparative analysis shows that the time–space measure imposes an upper bound on absolute inconsistency, while the spatial threshold method results in a wide range of unpredictable absolute inconsistency values. However, a limitation of the proposed technique is highlighted and a final hybrid solution is suggested. The performance of the hybrid scheme is then analyzed using simulation and live trials conducted over the Internet.

The rest of the paper is structured as follows. Section 2 questions the use of a distance-based threshold in a dead reckoning predictive contract algorithm. Based on this analysis, a novel time–space threshold metric is proposed and analyzed in Section 3. Section 4 highlights a drawback with the time–space threshold metric and considers the use of a hybrid metric. In Section 5, results from live Internet-based trials using the hybrid metric are presented. The paper ends with some concluding remarks in Section 6.

2. The Dead Reckoning Threshold Problem

The success of the dead reckoning scheme largely depends on the spatial threshold value employed. The threshold value determines the maximum spatial inconsistency that can occur between an entity's actual movement and its model of that movement. However, in a heavily loaded networked, the threshold value also impacts the temporal inconsistency between two nodes, as it directly influences the number of update packets generated. For example, a large threshold value results in less update packets leading to a corresponding decrease in temporal inconsistency in heavily loaded conditions, but also allows a larger spatial inconsistency to occur for an extended period of time, which causes an increase in absolute inconsistency.

The trade-off between temporal and absolute inconsistency is clearly evident in the inconsistency metric proposed by Zhou et al. [15]. Here, the absolute inconsistency Ω is defined as:

$$\Omega = \begin{cases} 0 & \text{if} \quad |\Delta(t)| < \varepsilon \\ \int_{t_0}^{t_0+\tau} |\Delta(t)| \mathrm{d}t & \text{if} \quad |\Delta(t)| \geq \varepsilon, \end{cases} \tag{1}$$

where Δ is the difference between the position of a local object and its remote replication, $\Delta(t)$ is the equivalent difference over a duration t, t_0 is the start time of the inconsistency, τ is the duration and ε is the minimum perceivable distance.

Essentially, Ω is the area under the graph of spatial consistency over the period between the transmission of two updates. When $\Omega = 0$ absolute consistency has been achieved. Zhou et al. [15] derive two equations to express inconsistency when dead reckoning is employed. The first of these, Ω_1, refers to the inconsistency accrued between the time an update packet is sent and the time it is received at the remote node. The second, Ω_2, refers to the inconsistency accrued between the time an update packet is received at the remote node and the time the next update is sent by the local node. The final value of inconsistency is simply obtained by adding Ω_1 and Ω_2:

$$\Omega_1 = \bar{v}\gamma\, T_\mathrm{d} + \delta T_\mathrm{d} + |a|_{\max}\, T_\mathrm{DR} \frac{T_\mathrm{d}^2}{2}, \tag{2}$$

$$\Omega_2 = \left(\bar{v}\gamma + \frac{\delta}{2}\right)(T_\mathrm{DR} - T_\mathrm{d}), \tag{3}$$

where v is velocity, γ is synchronization between remote and local clocks, T_DR is the time between updates, δ is the dead reckoning spatial threshold, T_d is the time between an update being sent and received (latency), and a is acceleration.

We therefore obtain

$$\Omega_1 = \delta T_\mathrm{d}^2 \tag{4}$$

and

$$\Omega_2 = \left(\frac{\delta}{2}\right)(T_\mathrm{DR} - T_\mathrm{d}). \tag{5}$$

These equations clearly illustrate the dual nature of temporal and absolute consistency in relation to the spatial threshold value δ. Consider the simplified scenario where each participant clock is synchronized so γ is zero. Entity velocity v is also assumed to be constant, so acceleration $a_{\max}$ is also zero. This scenario is represented in Equations (4) and (5). The network connecting participants is also overloaded. Increasing the threshold δ causes a decrease in synchronization messages, leading to a reduction in network latency T_d. This has the effect of reducing the value of inconsistency, as Ω_1 is reduced. On the other hand, increasing the threshold value also causes the time between updates T_DR to increase. This increases the value of Ω_2, meaning that overall inconsistency increases. The obvious question then arises: what is the impact of the spatial error threshold value on temporal inconsistency and

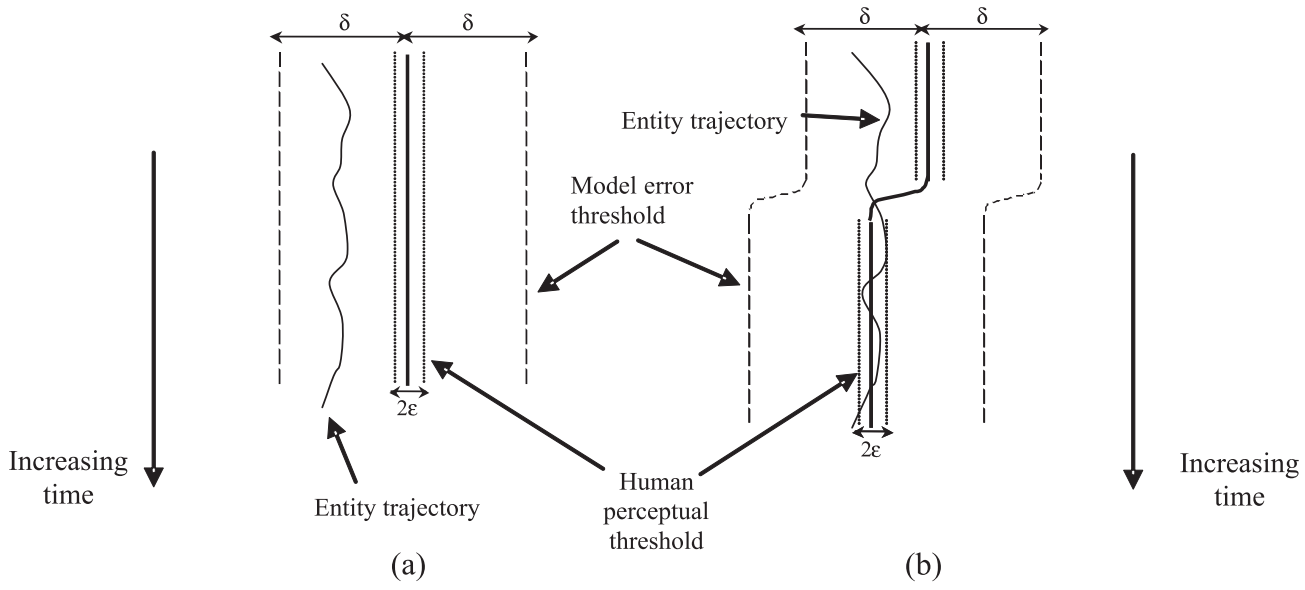

Figure 1. (a) An update is not generated as the entity remains within the spatial threshold; and (b) an update is generated to improve absolute consistency

absolute or time–space inconsistency? The answer is unclear, as increasing the spatial threshold can both increase and reduce inconsistency in the virtual world.

Using a spatial threshold in the dead reckoning prediction contract mechanism is evidently questionable, as varying its value within a heavily loaded network has an uncertain impact on the value of inconsistency. Furthermore, the spatial threshold also exhibits another performance related drawback. Consider the diagram in Figure 1a. In this case, the local user remains within the distance threshold δ, but outside the perceivable error ε. Such a scenario could arise in a network racing car game, for example, where an entity navigates a straight section of the racing track. As far as the spatial threshold is concerned, the position of the entity is considered to be spatially consistent as it remains within the distance error threshold, so no dead reckoning updates are generated. However, the position of the entity then remains spatially inconsistent over an extended period of time. This scenario could result in interaction difficulties. For example, in the case of the racing game, one driver might see a successful overtaking maneuver, while the other sees a collision. In this case, an objective outcome may need to be agreed between the distributed processes, which may leave at least one driver confused. The chances of this occurring can be reduced by sending an update when a spatial inconsistency below the threshold value persists over an extended period of time, as shown in Figure 1b.

It is clear from this example that it is not sufficient to only consider the spatial inconsistency when analyzing the performance of the dead reckoning scheme. The length of time the dead reckoning model has been inconsistent also needs to be considered. With this in mind, a model of the inconsistency arising during a dead reckoning update period, that takes both time and spatial inconsistency into account via Zhou's measure, will now be derived.

An overview of the operation of this inconsistency model is shown in Figure 2. The position of both the actual and modeled entity position is simulated. The position of the actual entity is governed by a path curvature value k. A high curvature value means that the actual position moves quickly away from the dead reckoning model, whereas a lower curvature will replicate the scenario shown in Figure 1a. At t_0 a dead reckoning update is transmitted. Both the model and actual entity then move with a constant speed of v. At time t, the model position and actual position are given by P_M and P_A, respectively:

$$P_{\mathrm{M}}\left(\frac{1}{k}, vt\right) \quad \text{and} \quad P_{\mathrm{A}}\left(\frac{\cos\theta}{k}, \frac{\sin\theta}{k}\right). \tag{6}$$

As speed is constant, the distance l traveled by both the modeled and the actual position in each time interval is equivalent. Using the standard arc length formula, the angle θ is found to be vkt. At each time-step the distance between the actual and the modeled position d can be calculated as:

$$\begin{aligned} d &= \sqrt{(P_{\mathrm{A}}(x) - P_{\mathrm{M}}(x))^2 + (P_{\mathrm{A}}(y) - P_{\mathrm{M}}(y))^2} \\ &= \sqrt{\left(\frac{\cos(vkt)}{k} - \frac{1}{k}\right)^2 + \left(\frac{\sin(vkt)}{k} - vt\right)^2}. \end{aligned} \tag{7}$$

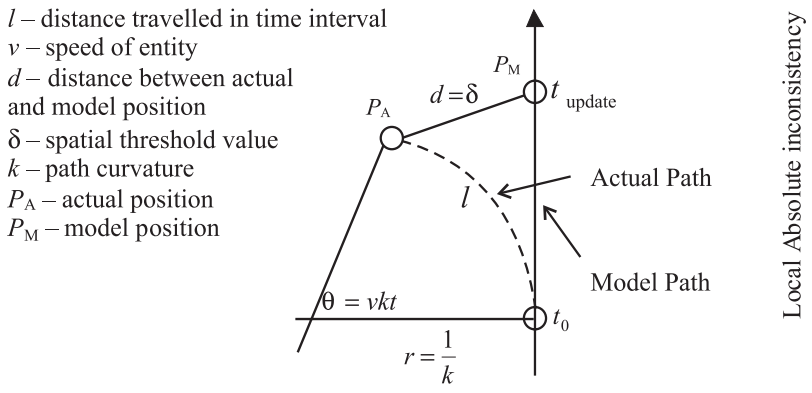

Figure 2. Model of actual and dead reckoning model positions

Approximating $\cos(vkt)$ and $\sin(vkt)$ using the Taylor series expansion i.e.

$$\cos(vkt) \approx 1 - \frac{(vkt)^2}{2} + \frac{(vkt)^4}{4} \quad \text{and}$$

$$\sin(vkt) \approx vkt - \frac{(vkt)^3}{3} \tag{8}$$

reduces Equation (7) to

$$d = 0.5v^2kt^2. \tag{9}$$

Evaluating Equation (9) gives the distance between the model and actual entity position at any time t during an update period. When d is equal to the spatial threshold δ, an update is transmitted. Hence, reworking Equation (9), the time taken to transmit an update t_{update} given a path curvature and spatial threshold value can be expressed:

$$t_{\text{update}} = \sqrt{\left(\frac{2\delta}{v^2k}\right)}. \tag{10}$$

According to Equation (1), absolute inconsistency during the update period t_{update} can now be calculated as:

$$\int_{t_0}^{t_0+t_{\text{update}}} d(dt) = \int_{t_0}^{t_0+t_{\text{update}}} 0.5v^2kt^2 = \frac{v^2kt^3}{6}\Bigg]_{t_0}^{t_{\text{update}}}. \tag{11}$$

Using Equation (10), the times taken to transmit an update for varying path curvature values and a spatial error threshold value of 1, 10 and 20 world units were calculated. Note, these threshold values were chosen to be representative of a low, medium and high threshold value for an entity of 5 world units in width. These values were then used in Equation (11) to calculate the local absolute inconsistency arising during a particular update period. The results of these calculations are presented in Figure 3, where local absolute inconsistency is plotted against varying curvature values. In all cases, v is arbitrarily set to a constant value of 25 world units per second (which is approximately 1.5 m s^{-1}). Note, this model assumes that all packets are delivered successfully, and that network latency is zero.

Analyzing Figure 3, it is clear that the spatial threshold performs poorly at the lower end of the spectrum of curvature values. When curvature is high, the time between updates is short, meaning absolute inconsistency is low. On the other hand, when curvature is low, the time between update packets is long. In this case, the modeled entity can remain inside the error threshold over an extended period of time, leading to an increase in absolute inconsistency. Clearly this is not an ideal scenario. By using the measure of absolute inconsistency, situations such as that presented in Figure 1a can be quantified and identified, providing a more complete reflection of the true performance of the spatial threshold. In the next section, it is demonstrated how this measure can be employed in a novel error threshold scheme, in order to prevent the scenario depicted in Figure 1a occurring.

3. The Time–Space Threshold Metric

In an attempt to alleviate some of the drawbacks associated with the spatial threshold, and inspired by Zhou et al's inconsistency metric, we propose a novel threshold metric, known as the time–space threshold metric, which takes both distance and time into account during its operation. Under this scheme, the local spatial modeling error is integrated over the time interval between two successive update packets. This gives a measure of local time–space inconsistency, represented by the shaded area in Figure 4. Update packets are then generated when this local inconsistency value exceeds a specified threshold value. In this way, absolute inconsistency has an upper bound and will never be able to increase indefinitely.

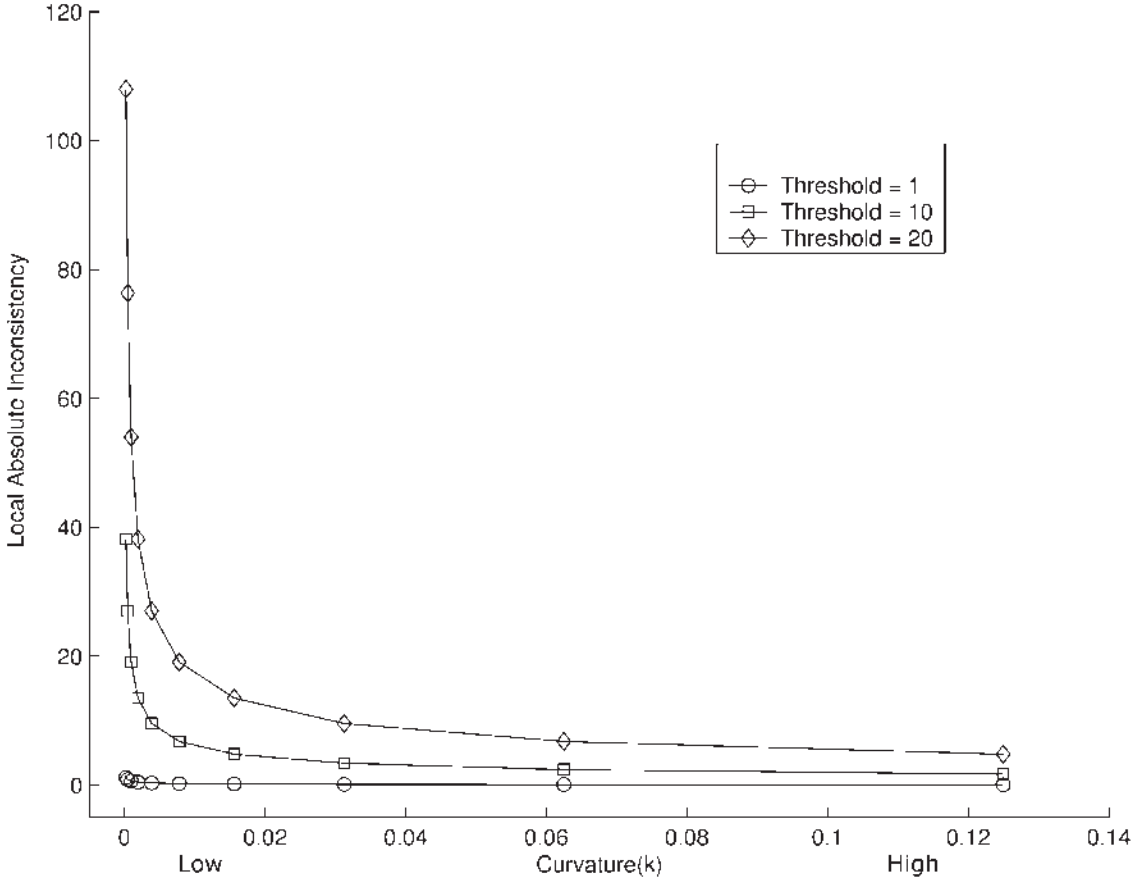

Figure 3. Absolute inconsistency for varying curvature: spatial threshold metric used

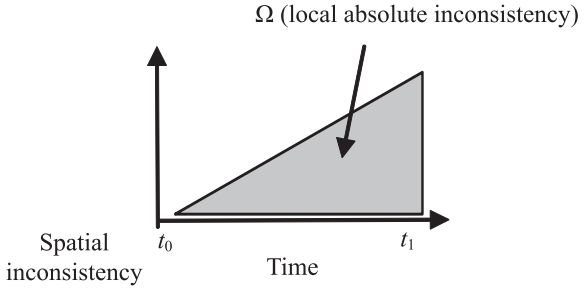

Figure 4. The actual and modeled path using dead reckoning and a spatial threshold metric

In order to fully evaluate the time–space threshold and compare its performance to that of the spatial threshold, a number of experiments were conducted. These will now be described, along with an analysis of the results collected.

3.1 Experimentation and Analysis

An experimental test platform was implemented using the Torque game engine [16]. The engine was extended to support logging of player position. Positional data from three different participants of varying sex, age and virtual environment experience were collected in a game environment that consisted of a single route from a start to target position. A plan view of the test environment can be seen in Figure 5a. Each participant was first given a practice run with the environment in order to familiarize themselves with the controls and the environment itself. The behavior of each participant was then recorded for three separate attempts at the course.

This data was then used as input for a Matlab simulation. This simulation applies a first order dead reckoning algorithm to the input data, which is recorded from the experimental platform. The threshold metric and threshold value was varied for each experiment. The simulation results are now presented and discussed. It should be noted that results are given for only one typical player, but similar results were obtained for all players. Figure 5

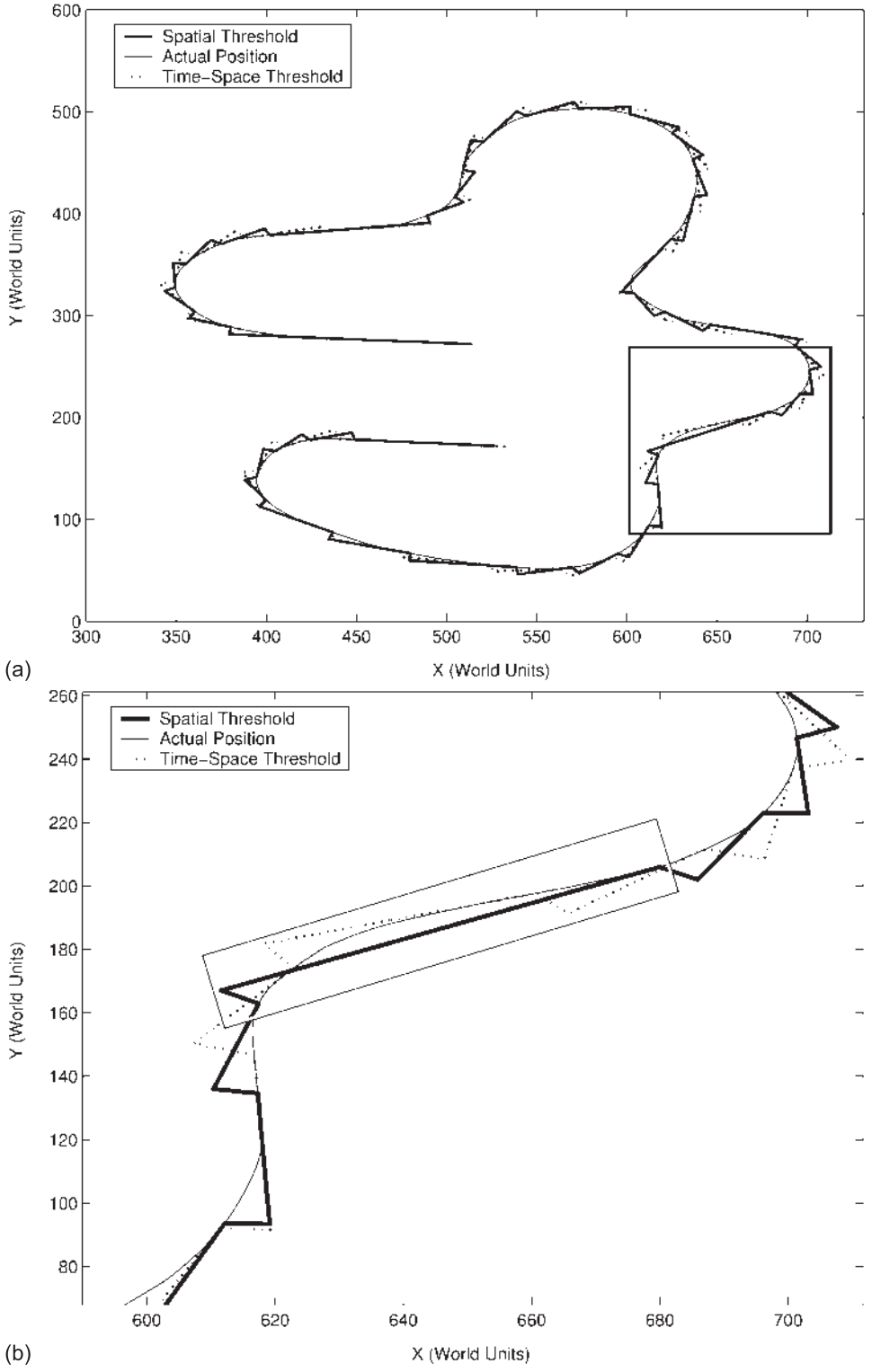

Figure 5. Actual and modeled positions for a spatial threshold of 7 world units and a time–space error threshold of 7 world units-seconds. (b) A zoomed-in section of the graph in (a).

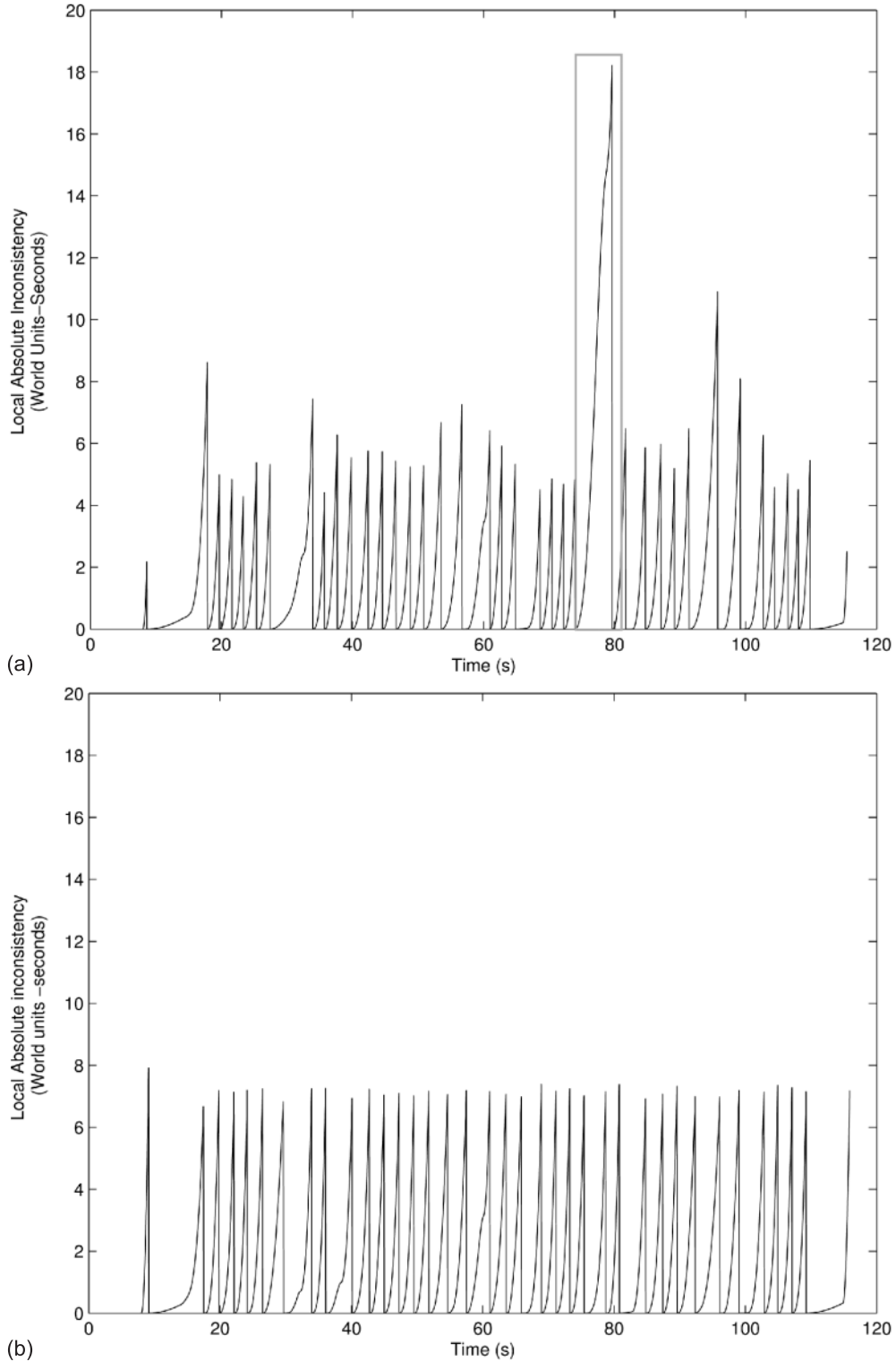

Figure 6. Local absolute inconsistency for (a) a spatial error threshold of 7 world units and (b) a time–space error threshold of 7 world units-seconds

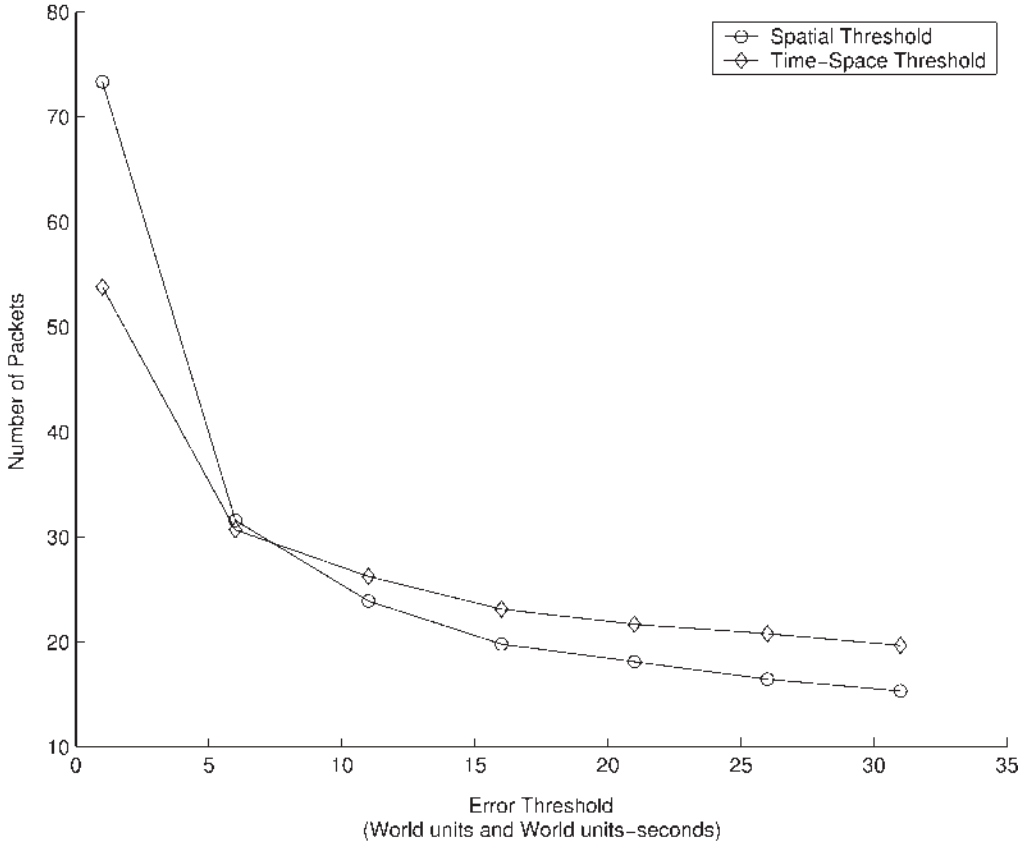

Figure 7. Average packets required for increasing spatial and time–space error threshold

shows the actual player path, along with the modelled position using both the distance and the time–space threshold metrics. An error threshold of 7 world units and world units seconds was used for both models. This value was chosen, as the virtual entity used in the experiments is approximately 3.5 world units in width. The expanded view clearly shows that update packets are sent at different times depending on the threshold metric used.

Figures 6a and b show the local absolute inconsistency arising during the simulation for a spatial threshold metric and a time–space threshold metric, respectively. The peaks in both cases represent a time when an update packet is generated. When this occurs, the measure of local absolute inconsistency reinitialized to 0. Figure 6a shows a wide range of varying local absolute inconsistencies when the spatial threshold metric is used. The highlighted section in Figure 6a corresponds to the section highlighted in the expanded view in Figure 5. In this case, it can be seen that the actual position deviates around the model position, but does not exceed the error threshold. This is a low path curvature scenario, which results in the high absolute inconsistency visible in the highlighted section in Figure 6a. In contrast, the time–space threshold metric results in a clearly upper-bounded local inconsistency measure (Figure 6b), thus preventing an indefinite increase in inconsistency, as would be the case for the situation demonstrated in Figure 1b. This result seemingly vindicates the use of the latter threshold metric.

3.2 Limitations of Proposed Technique

As with all bandwidth optimization techniques, it is of utmost importance that the packets generated by the scheme are examined. Figure 7 shows the average number of update packets required for varying error threshold values of both the spatial and time–space error threshold values. As expected, the number of packets decreases as the error threshold increases in both cases. Intuitively, it would be expected that the time–space threshold would always generate a higher number of packets than the spatial threshold value, as it provides a lower level of inconsistency. However, upon analysis of the collected data, it was found that at lower threshold values this was not the case; more update packets are required when the spatial threshold metric is employed. The reason behind this can be fully understood by examining the performance of the time–space

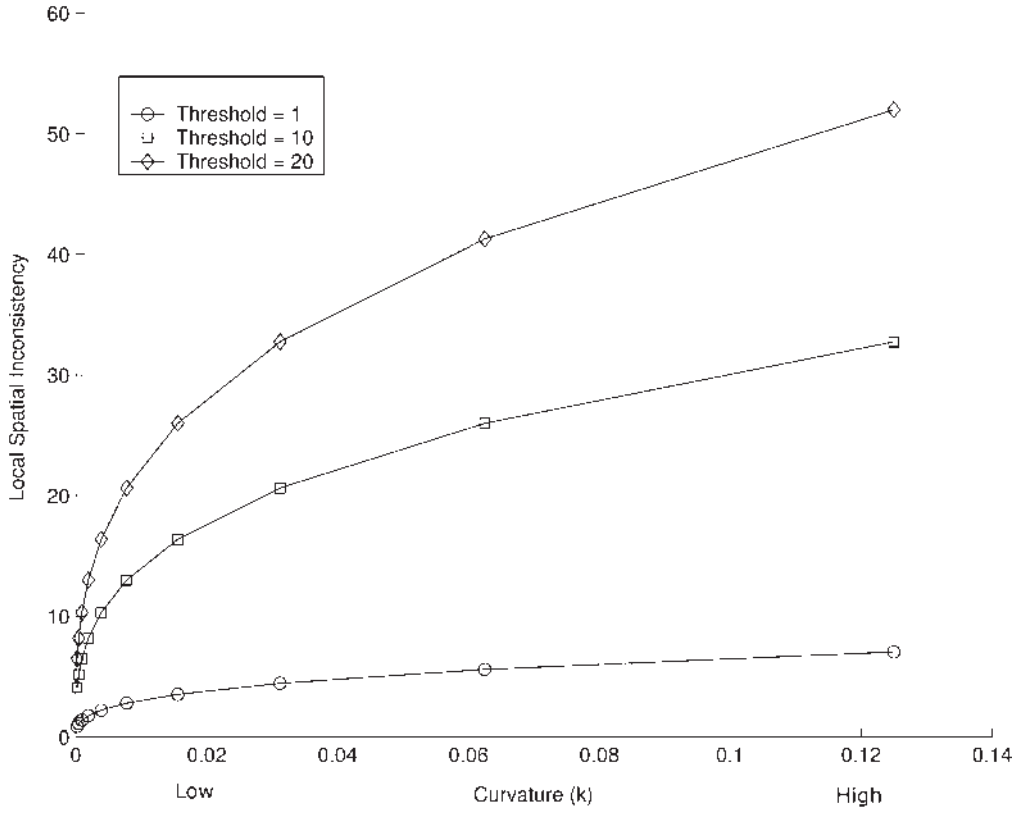

Figure 8. Spatial inconsistency for varying curvature: time–space threshold metric used

threshold scheme from the point of view of spatial inconsistency, similar to the analysis carried out earlier in the context of the spatial threshold and absolute inconsistency. This can be achieved using Equations (9) and (11).

Using Equation (11), the time between updates when a time–space threshold of 1, 10 and 20 world units-seconds was calculated for varying path curvature values. Again, v was set to 25 world units per second. The resulting time values were then employed in Equation (9) to determine the amount of local spatial inconsistency that would have arisen during that update period. The results from these calculations are shown in Figure 8, where local spatial inconsistency is plotted against varying curvature values.

Figure 8 demonstrates that, in terms of spatial inconsistency, the time–space threshold performs poorly at higher path curvature values. For example, at very high curvature, a time–space threshold of 10 world unit-seconds yields a spatial inconsistency of approximately 30 world units during an update period. This occurrence (a very large spatial error in a very short period of time) is due to the fact that integrating the model error up to a point in time t_1 can actually result in a smaller value than the distance error between at time t_1, as demonstrated in Figure 9. Here, a scenario of high path curvature is shown, where the dead

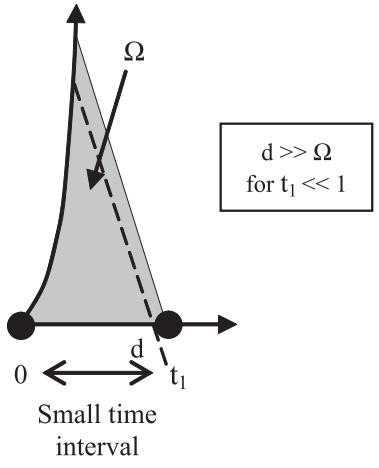

Figure 9. Limitation of using a time–space threshold metric

reckoning model becomes very inaccurate very quickly. If a distance-based threshold metric is employed, then an

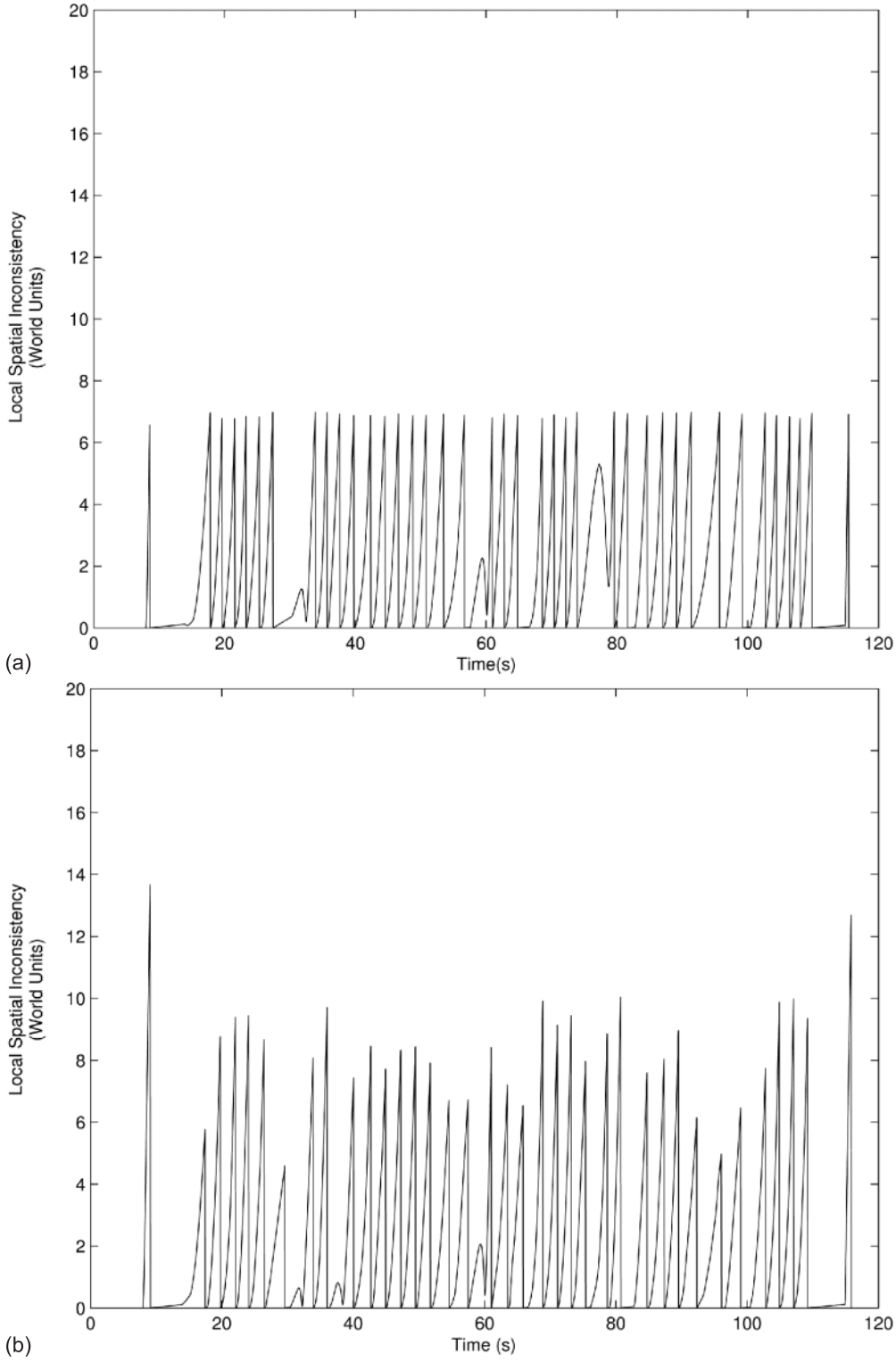

Figure 10. Local spatial inconsistency for (a) a spatial error threshold of 7 world units and (b) a time–space threshold of 7 world units-seconds

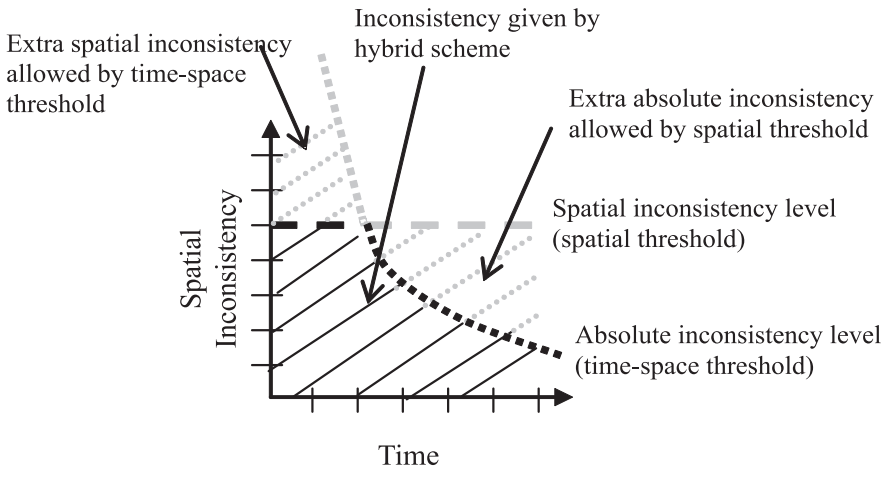

Figure 11. The different threshold schemes allow different levels and types of inconsistency

update packet will be forthcoming almost immediately, which is what we would hope for. However, if a time–space threshold metric is used, it could take significantly longer before an update packet is sent, as a small time interval counters the large modeling error. This means that an entity's remote movement could potentially be very inaccurate over a short period of time, resulting in extremely poor spatial consistency.

This analysis is evident in the results shown in Figure 10. Using the same input trajectory and MatLab simulation as the experiments outlined in Section 3.1, the spatial inconsistency was recorded for both a spatial threshold of 7 world units and time–space threshold of 7 world unit-seconds. The results are shown in Figures 10a and b, respectively. As would be expected, spatial inconsistency is bounded to the spatial error threshold value. In the case of the time–space threshold metric however, spatial inconsistency varies, and can be likened to the local absolute inconsistency measured when a spatial threshold is employed as shown in Figure 6a.

It is clear from the analysis presented here and in Section 2 that both threshold metrics are sensitive to opposite ends of the spectrum of curvature values, and tend to under-perform within these regions. Further examination of the data presented in these sections, particularly Figures 6 and 10, suggests that the obvious solution is to use both threshold metrics in one dead reckoning algorithm. In the next section, such a scheme is discussed, and its ability to bound both types of inconsistency is demonstrated.

4. The Hybrid Threshold Scheme

In order to deal with the two key performance-related issues of both schemes, a novel hybrid dead reckoning algorithm is now proposed. Under this scheme, both metrics are simultaneously evaluated. As soon as one has reached the error threshold, an update packet is transmitted. Both spatial and time–space inconsistency measures are then reinitialized to zero. The benefit of using the hybrid threshold scheme can be appreciated by examining Figure 11. Here, spatial inconsistency is plotted against time. Absolute inconsistency is the shaded area underneath the plotted graph lines. If the spatial threshold is employed, the amount of spatial inconsistency is limited to the threshold value. Similarly, when the time–space threshold metric is used, absolute inconsistency is also bounded to the time–space threshold value. The line representing this level is curved as, under the time–space threshold scheme, a small spatial inconsistency over a long period of time is equivalent in terms of inconsistency to a large spatial inconsistency in a short period of time.

The lightly colored areas in the figure expose the limitations of both techniques. Using either scheme in isolation allows extra inconsistency to occur. The spatial threshold allows extended absolute inconsistency over longer time periods, whereas the time–space scheme can give rise to increased spatial inconsistency over short time periods. By using both in tandem, inconsistency is bounded to the darkly shaded area.

To test the hybrid scheme, a number of MatLab simulations were again conducted. Figure 12 shows the local spatial and absolute inconsistency for a dead reckoning routine using the hybrid metric. An error threshold of 7 is used for both metrics. The input trajectory is again the same as employed in previous tests. The effect of simultaneously using the two threshold metrics is evident from Figure 12. There are now bounds placed on both levels of inconsistency. To further demonstrate these bounds, two sections of both figures have been emphasized. The areas marked as A1 in both figures highlight a low path curvature scenario. This situation was quickly identified using the time–space threshold, however, and an update

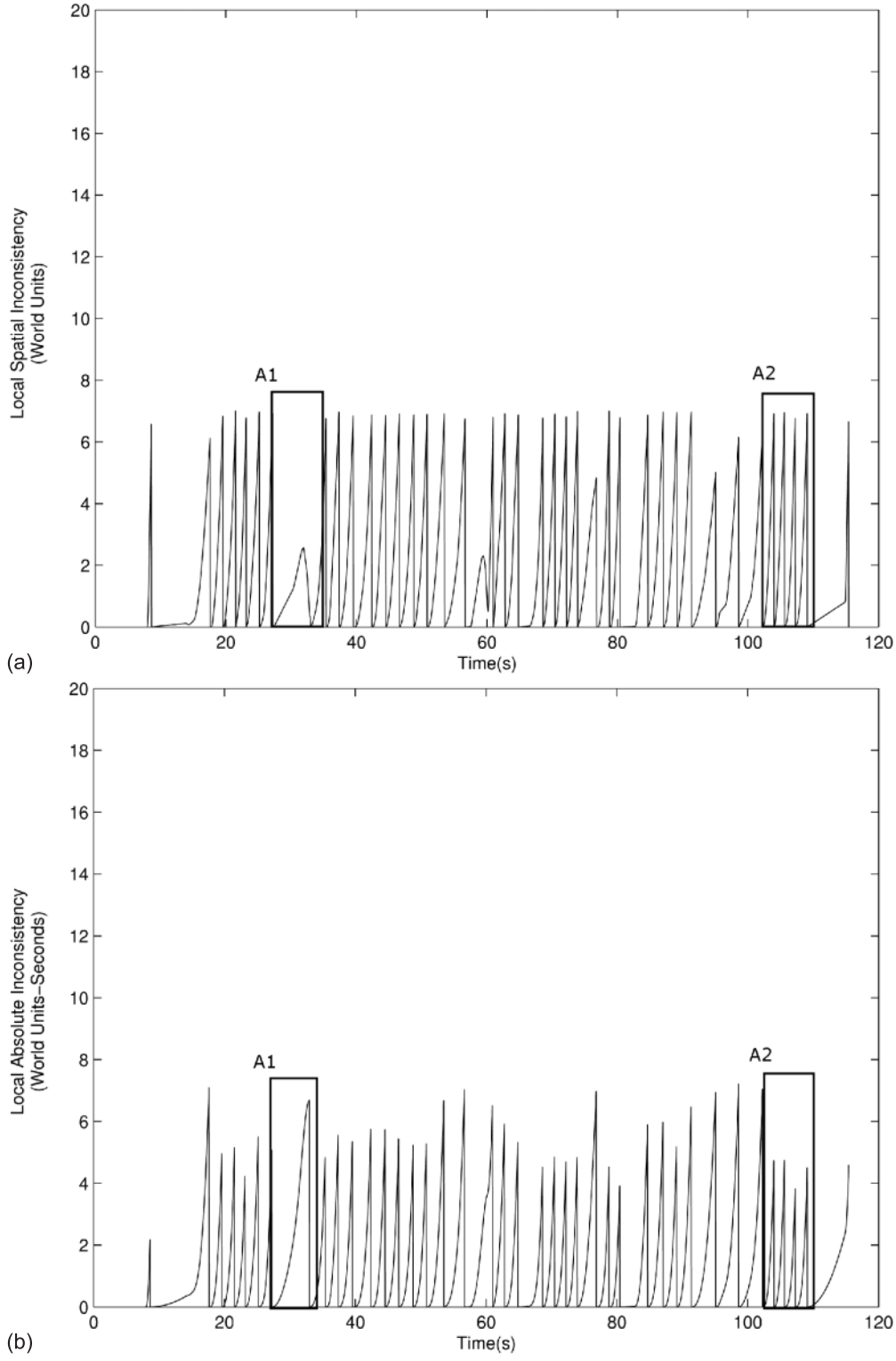

Figure 12. (a) Local spatial inconsistency and (b) local absolute inconsistency for a hybrid error threshold of 7 world units and 7 world units-seconds

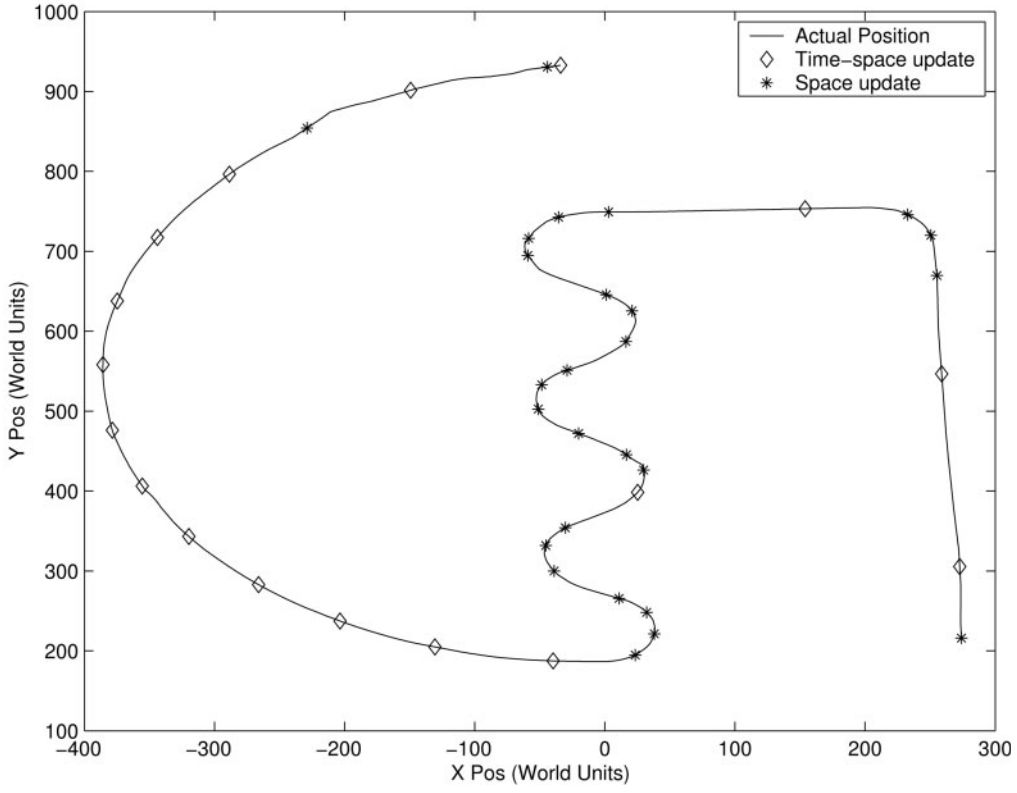

Figure 13. Varying updates are transmitted as the curvature of the course changes

was generated. The areas marked as A2 indicate a high path curvature scenario. Using the time–space threshold in isolation here could have allowed a large spatial inconsistency. As the hybrid threshold is being employed, this situation is identified by a breach of the spatial error threshold.

5. Internet-Based Trials

To strengthen the mathematical and analytical analysis presented in the previous sections, a number of live Internet-based trials were conducted using the experimental test platform implemented in the Torque games engine. The engine was further modified to support the three dead reckoning threshold schemes (spatial, time–space and the hybrid). As dead reckoning relies on replicated databases, the engine was also modified to support a peer-to-peer network architecture [17]. Six participants of varying age, sex and virtual environment experience participated in the trials which took place across the Internet, ensuring that data was subject to real network problems such as latency and jitter. Three participants were based in University of Salford, UK and the others were based in the National University of Ireland, Maynooth, Ireland. Each participant was first given a chance to familiarize themselves with the course and the controls of the virtual environment. A plan view of the course can be seen in Figure 1s. The objective of the task assigned to participants was to complete the given course before the other participant. For ease of analysis, each section of the course was given a constant curvature value.

Two participants, one from each of the two universities, were involved in each trial. Six experiments were conducted during each trial. These experiments tested the use of the three different threshold metrics for error threshold values of 5 and 10 (world units and world units-seconds, depending on the threshold employed). The clocks on participant machines were synchronized at the start using the Network Time Protocol [18]. On each game tick (approximately 32 ms), each participant logged the position of all virtual entities visible on their local machine, including their own position. Each participant also logged the number and type of updates transmitted during the simulation. Each log entry was time stamped with synchronized wall clock time.

5.1 Results and Analysis

We now examine the results collected in the live trials described above. Figure 14 shows remote spatial and ab-

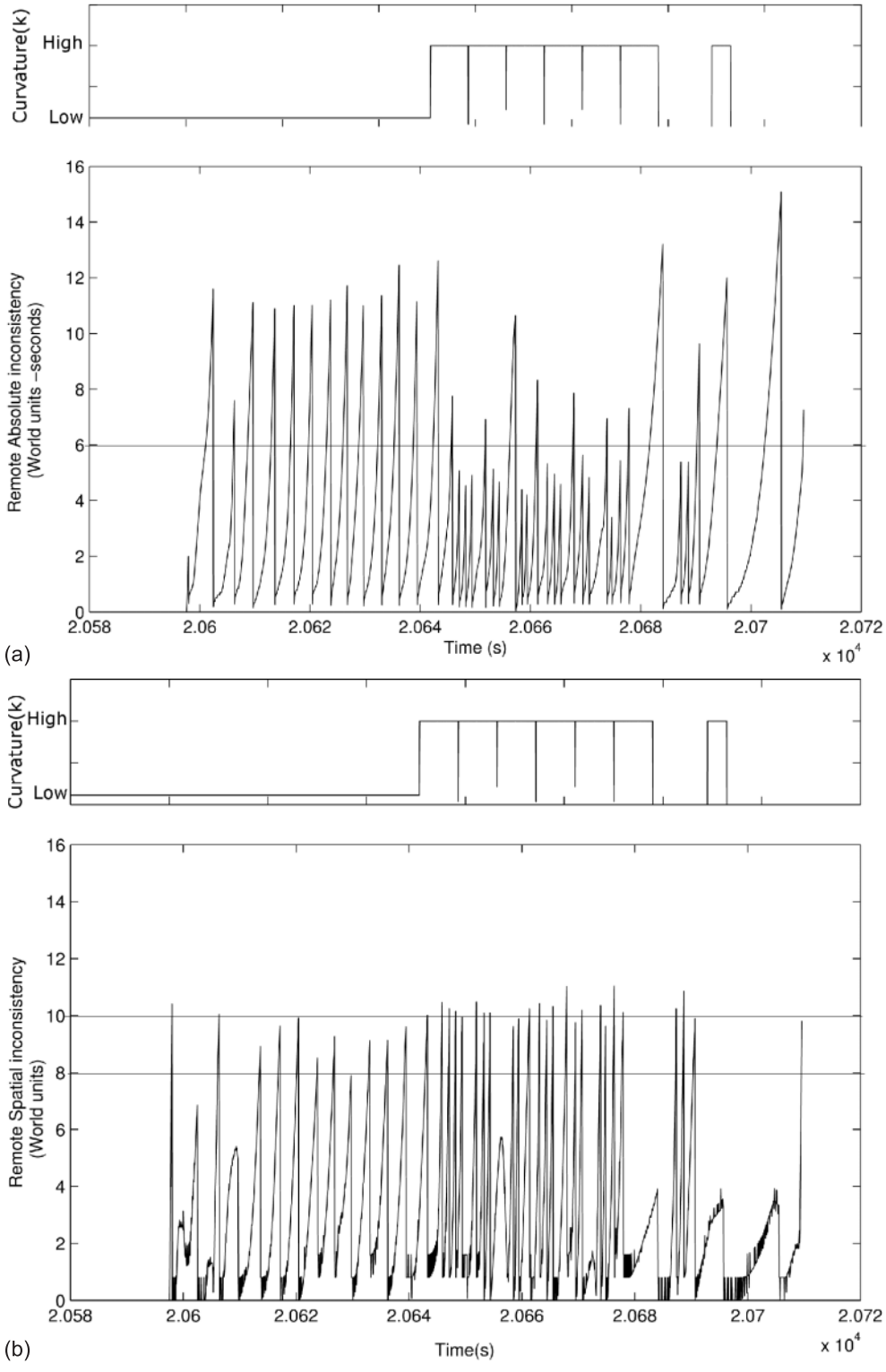

Figure 14. (a) Absolute and (b) spatial inconsistency using a hybrid threshold of 10 world units and 10 world units-seconds

solute inconsistency obtained when a hybrid threshold of 10 world units-seconds is used. A plot of the course curvature is also shown in both figures in order to aid analysis. It should be noted that, while the results presented in this section are for a single user, they are nevertheless representative of all the users. Also, the actual value of time in the graphs of both figures represents the point in time in which data was collected with respect to the start of the time server. Furthermore, Figure 13 plots the types of updates transmitted during the simulation from Figure 14. This is superimposed over the participant trajectory. Results were also collected for both the spatial and time–space threshold being employed in isolation. However, they are similar to that presented in Section 3 and have not been reproduced here.

The effect of latency and jitter on the simulation is evident from Figure 14. Each peak represents a point where an update is generated. It can be seen how some peaks rise above the threshold level of 10 in both cases. The effect is greater in Figure 14a, as the absolute inconsistency measure takes both time and space into account. The increased inconsistency in this case is due to the extra time caused by latency and jitter (T_d in Equations (1) and (2)). However, taking the latency factor into account, it can be seen how using the hybrid threshold value places bounds on both levels of inconsistency arising from the dead reckoning scheme.

By comparing both plots in Figure 14 with their respective curvature plots and considering the results presented in Figure 13 it can be seen that the impact of curvature demonstrated in our previous analysis, although still evident, has now been mitigated. It can be seen in Figure 14b that on the sections of the course with low curvature, spatial inconsistency is approximately 8 world units or lower, and would not have generated an update. However, the time–space threshold can capture and quantify this behavior, resulting in an update being generated during this period. This effect is evident in Figure 13, where it can be seen how the start and end sections of the track, those with the lowest curvature, have the highest frequency of breaches of the time–space threshold. On the other hand, in the middle section of the track and on the sharp corners where curvature is higher (Figure 13), the frequency of spatial threshold updates increases. This effect is also evident by analyzing the absolute inconsistency falling under the high curvature area in Figure 14a. Absolute inconsistency is approximately 6 unit-seconds or lower in this area, and would not have generated an update until a later time.

The tighter bounds placed on inconsistency by the hybrid scheme result in higher levels of both absolute and spatial consistency, supporting more accurate and objective shared experience of relative movement of the virtual entities. However, this tighter consistency has an impact on the number of packets generated in the environment, as can be seen in Figure 15. Here, the average number of packets generated per user in the live trials for spatial, time–space and hybrid threshold values of 5 and 10 are considered. The results show that the hybrid threshold results in a relatively small increase in packets across the network. This increase in packet number is expected, as the hybrid threshold reduces inconsistency when compared to both other approaches.

6. Conclusions

The maintenance of a low level of inconsistency in the face of network limitations is a key issue in distributed interactive applications. An important tool in the maintenance of a low level of inconsistency is dead reckoning. This approach uses a measure of local spatial inconsistency to determine when to transmit synchronization messages. In this work, the effect of using such a measure with dead reckoning was examined, and a key performance-related issue was highlighted. It was shown that the spatial threshold value alone cannot identify scenarios where local spatial inconsistency persists over an extended period of time. Its use can result in a large level of absolute inconsistency.

This was examined in further detail using a model of entity path curvature and local absolute inconsistency. This analysis concluded that the spatial threshold performs poorly when the modeled entity path curvature is low. Based on this result a new threshold, known as the time–space threshold, was proposed. This metric measures local absolute inconsistency and when this value reaches a predefined error threshold, a dead reckoning update is generated. Simulation results using this threshold show that while this metric prevents the case of indefinite local absolute inconsistency, it can result in a large local spatial inconsistency over short time periods.

A novel hybrid threshold metric, which evaluates both the spatial and time–space threshold simultaneously, was then proposed. Simulation results demonstrate that this approach places a bound on both local spatial and local absolute inconsistency, thus providing the increased performance benefits of both metrics.

A number of live trials were then conducted using the spatial, time–space and hybrid threshold schemes. Analysis of the remote inconsistency values collected during the live trials confirms our theoretical analysis in the face of actual network latency and jitter. The hybrid threshold scheme places bounds on spatial and absolute inconsistency. However, it results in a relatively small increase in network traffic. As with the spatial threshold, the actual impact of this extra network traffic on inconsistency is unclear in a heavily loaded network. The increase in traffic improves absolute inconsistency, but may reduce temporal inconsistency. Future work will investigate this trade-off in detail.

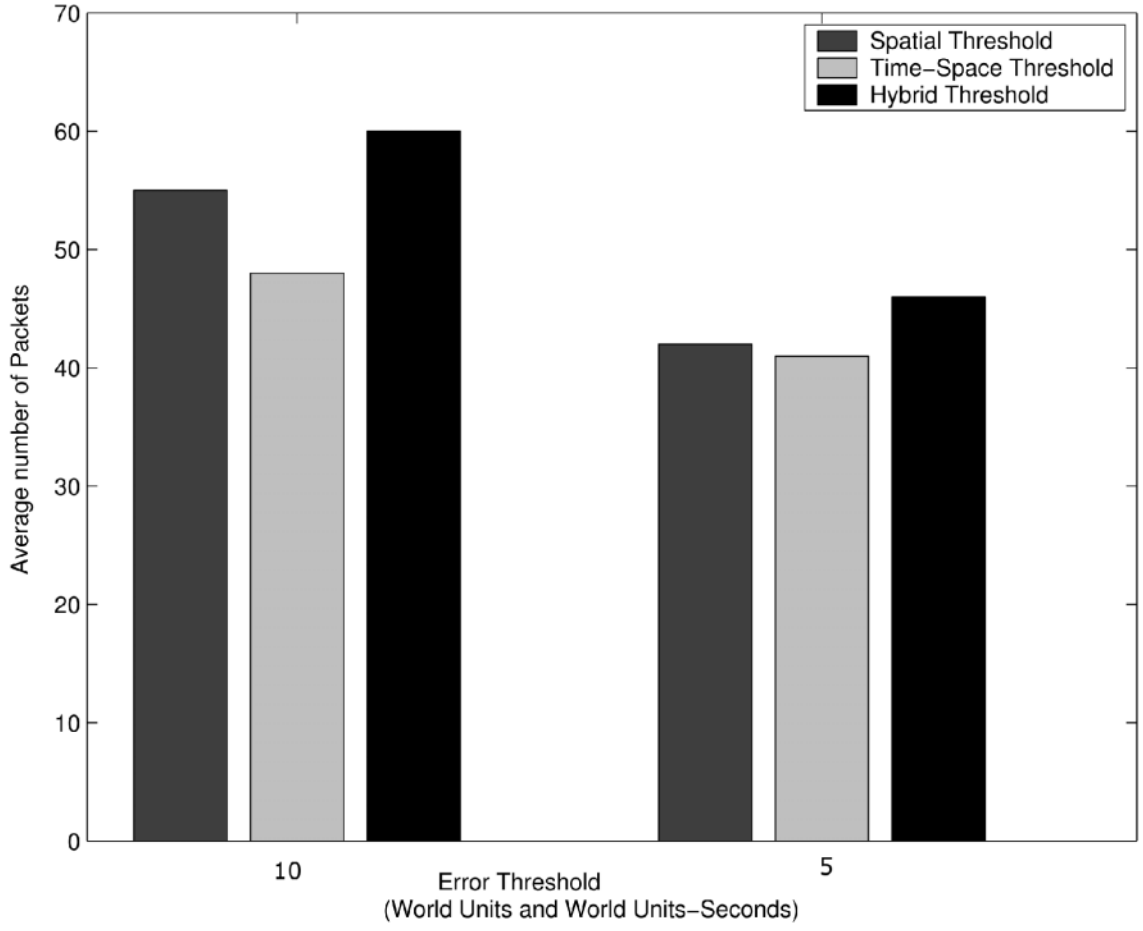

Figure 15. Average number of packets generated for the different error threshold metrics during the live Internet trials.

7. Acknowledgements

This work was supported by Science Foundation Ireland and Enterprise Ireland under grant IRCSET/SC/04/CS0289 and by Marie Curie HPMT-CT-2001-00390.

8. References

[1] Gautier, L. and C. Diot. 1998. Design and Evaluation of MiMaze, a Multi-player Game on the Internet. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, June 28–July 1, Austin, Texas, pp. 233–236.

[2] Kauff, P. and O. Schreer. 2002. An immersive 3D video-conferencing system using shared virtual team user environments. In *Proceedings of the 4th international conference on Collaborative virtual environments*, September 30–October 02, Bonn, Germany, pp. 105–112.

[3] Churchill, E. and D. Snowdon. 1998. Collaborative virtual environments: An introductory review of issues and systems. *Virtual Reality* 3(1), 3–15.

[4] Delaney, D., T. Ward, and S. McLoone. 2006. On consistency and network latency in distributed interactive applications: A survey – Part I. *Presence: Teleoperators and Virtual Environments* 15(2), 218–234.

[5] Delaney, D., T. Ward, and S. McLoone. 2006. On consistency and network latency in distributed interactive applications: A survey – Part II. *Presence: Teleoperators and Virtual Environments* 15(4), 465–482.

[6] Roehle, B. 1997. Channeling the data flood. *IEEE Spectrum* 34(3), 32–38.

[7] Singhal, S. and M. Zyda. 1999. *Networked virtual environments: design and implementation*. ACM Press/Addison-Wesley Publishing Co.: New York.

[8] Roberts, D. 2004. Communication infrastructures for inhabited information spaces. Inhabited Information Spaces, Living with your Data: p. 233–267.

[9] Joslin, C., T. Di Giacomo, and N. Magnenat-Thalmann. 2004. Collaborative virtual environments: from birth to standardization. *Communications Magazine, IEEE* 42(4), 28–33.

[10] Delaney, D., T. Ward, and S. McLoone. 2003. Reducing Update Packets in Distributed Interactive Applications using a Hybrid Mode. In *Proceedings of 16th International Conference on Parallel and Distributed Computing Systems*, August 13–15, Reno, USA, pp. 417–422.

[11] McCoy, A., T. Ward, S. McLoone, and D. Delaney. 2006. Multi-step-ahead Neural-Networks for Network Traffic Reduction in DIAs. To appear in. *ACM Transactions on Modelling and Computer Simulation* (TOMACS).

[12] IEEE. 1993. IEEE Standard for Distributed Interactive Simulation: Application Protocols, in IEEE Std 1278, IC Society, Editor. IEEE: New York.

[13] Bouillot, N. and E. Gressier-Soudan. 2004. Consistency models for distributed interactive multimedia applications. *ACM SIGOPS Operating Systems Review* 38(4), 20–32.

[14] Gautier, L., C. Diot, and J. Kurose. 1999. End-to-end Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet. In *Proceedings of INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, March 21–25, pp. 1470–1479.

[15] Zhou, S., W. Cai, B. Lee, and S. Turner. 2004. Time-space consistency in large-scale distributed virtual environments. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 14(1), 31–47.

[16] Llyod, J. 2004. The Torque Game Engine. *Game Developer*, 8–9.

[17] Marshall, D., S. McLoone, T. Ward, and D. Delaney. 2006. 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*, 22–24 November 2006, Dublin, Ireland, pp. 88–92.

[18] Mills, D. 1989. Network Time Protocol (version 1): Specification and implementation. DARPA Network Working Group Report RFC-1119, University of Delaware, September.