Dead Reckoning-Based Update Scheduling Against Message Loss for Improving Consistency in DVEs

Zengxiang Li, Wentong Cai, Xueyan Tang
Parallel and Distributed Computing Center
School of Computing Engineering
Nanyang Technological University, Singapore 639798
Email: {zxli, aswtcai, asxytang}@ntu.edu.sg

Suiping Zhou
School of Computing
Teesside University, UK
Email: S.Zhou@tees.ac.uk

***Abstract*—Maintaining a consistent presentation of the virtual world among participants is a fundamental problem in the Distributed Virtual Environment (DVE). The problem is exacerbated due to the limited network bandwidth and error-prone transmission. This paper investigates Dead Reckoning (DR) update scheduling to improve consistency in the DVE against message loss. Using the metric of Time-Space Inconsistency (TSI), which combines the spatial magnitude and temporal duration of inconsistency, we analytically derive the impact of message loss on TSI when using a DR-based update mechanism. To improve consistency against message loss, a naive update scheduling algorithm is first proposed, in which the expected spatial difference is calculated by taking message loss into account. In order to reduce the TSI to the case without transmission failures, a compensation update scheduling algorithm is further proposed by reducing the DR threshold according to the message loss rate. Using these algorithms, a budget-based mechanism is developed to meet the network bandwidth constraint. We show through experiments using a racing car game that the budget-based mechanism using the compensation update scheduling algorithm makes the best use of available network bandwidth to reduce the inconsistency and its impact on the participants. In addition, it ensures fairness among participants in spite of widely varying message loss rates.**

I. Introduction

A distributed virtual environment (DVE) encourages a large amount of participants from different corners of the world to communicate and interact with each other in a virtual world [1]. DVEs are widely used in various areas such as military training [2], e-learning [3], and massively multiplayer online games [4]. Maintaining a consistent presentation of the virtual world among participants is a fundamental problem in the DVE. Once the state of an entity in the virtual world changes, all participants should be able to observe the change in real time. Otherwise, inconsistent views can seriously affect meaningful interactions among the participants. For instance, some players in a car racing game may behave unexpectedly after observing the wrong positions of their cars. Recently, wireless distributed virtual environments have attracted more and more interests, because wireless networks, especially IEEE 802.11 WLAN [5], have emerged as a common last-hop in the Internet. However, wireless networks commonly have limited bandwidth due to the power restriction. They are also error-prone due to various factors such as obstacles and signal interference. For these reasons, the problem of consistency maintenance in the DVE is exacerbated.

Dead Reckoning (DR) [6], [7] is commonly used to extrapolate the position of an entity, for reducing the amount of data exchanged over the network. In this mechanism, DR updates of the entity are sent only when the difference between real position and extrapolated position is greater than a predefined threshold. In this paper, we investigate DR-based update scheduling to utilize the limited network bandwidth efficiently and to improve consistency in DVEs in the presence of transmission failures. Zhou et al [8] have proposed a metric called *Time-Space Inconsistency* (TSI) to measure the inconsistency by combining its spatial magnitude and temporal duration. It has been shown that TSI is inherent in DVEs and may significantly affect the perceptions of participants.

Due to network failures, some DR updates might be lost in the transmission. As a result, the receiver would fail to use the most recently sent DR update to extrapolate the position of the entity, which would increase the inconsistency of the entity. Using the metric of TSI, we analytically derive the impact of message loss on inconsistency when using a DR-based update mechanism. To improve consistency against message loss, a naive update scheduling algorithm is first proposed. In this algorithm, message loss in the transmission is taken into account to calculate the expected spatial difference between the real position of an entity and its extrapolated position at the receiver. A DR update is sent when the expected spatial difference exceeds the predefined threshold. In order to reduce the TSI to the case without transmission failures, a compensation update scheduling algorithm is further proposed by reducing the DR threshold according to the message loss rate. With limited bandwidth provided by the network, some DR updates may not be sent in time. To send DR updates selectively within the bandwidth constraint, a budget-based mechanism is then developed using the above update scheduling algorithms. Experimental results show that the budget-based mechanism using the compensation algorithm has advantages over that using the naive algorithm. It can provide fairness among participants in spite of widely varying transmission failures. With enough network bandwidth available, it can even reduce the inconsistency and its impact on the perceptions of participants to the level of the case without transmission failures.

The rest of the paper is organized as follows: Section II discusses the related work. Section III introduces the system

model and our objectives. Section IV presents the naive and compensation update scheduling algorithms, as well as the budget-based mechanism. Section V describes the experiments using a racing car game and discusses the experimental results. Finally, Section VI concludes the paper and outlines the future work.

II. Related work

In Dead Reckoning (DR) [6], [7], each participant maintains a DR model to extrapolate the position of a moving entity between updates. Hence, participants can observe the approximate view using less number of position updates. In a DR-based update mechanism, there is inconsistency between the real position and the extrapolated position of an entity. It is straight-forward to measure the inconsistency using the spatial difference. Zhou et al [8] have verified that the temporal duration of the inconsistency also affects human perception significantly. Therefore, the metric TSI is proposed to measure inconsistency by combining spatial difference and temporal duration. Similarly, the cumulated error within the update interval is used to measure the inconsistency in [9], [10]. Roberts et al [11] have also proposed to use an alternative DR threshold based on spatial difference and temporal duration.

As shown in [8], [12], the sender and receiver of DR updates may observe inconsistent views of an entity, if their clocks are not synchronized. A globally synchronized clock can be used to reduce the inconsistency of the entity. The impact of message delay on the inconsistency of entities and on the player's experience in the virtual world is studied in [8], [12], [13], [14], [15]. These studies have reached an agreement that the inconsistency increases with increasing network delays. Local-lag technique is used in [16] to account for network delays. Different from the above work, we focus on improving consistency in DVEs in the presence of transmission failures in the DR-based update mechanism. As illustrated in [15], players in a car racing game may observe different positional relations of cars due to message loss. Cheung and Sakamoto [17] have proposed to use application-aware redundant Extrapolated Parity Packets to recover the lost DR updates. Instead of recovering the lost DR updates which might be out of date anyway, we propose to schedule DR updates more frequently to reduce the impact of message loss on the TSI of entities.

With the constraint of network bandwidth, the sender may fail to send DR updates to all receivers in time. To utilize the limited bandwidth efficiently, a number of update scheduling algorithms (e.g., [18], [19], [20], [21]) have been proposed. Different from these algorithms, the update scheduling algorithms proposed in this paper take message loss into account. A budget-based mechanism is further developed using the proposed update scheduling algorithms to send DR updates selectively within the bandwidth constraint.

III. System Model

A DVE normally consists of a group of interconnected and geographically distributed computers (nodes) that simulate the entities in the virtual world. The nodes are generally organized in either a client-server or a peer-to-peer architecture. Without loss of generality, this paper considers update dissemination from a node hosting an entity to another node interested in the state of the entity. The former is called the sender, which can be either a server in the client-server architecture or a peer in the peer-to-peer architecture. The latter is called the receiver, which is either a client in the client-server architecture or a peer in the peer-to-peer architecture. In this paper, we focus on the updates of entity positions in the virtual world.

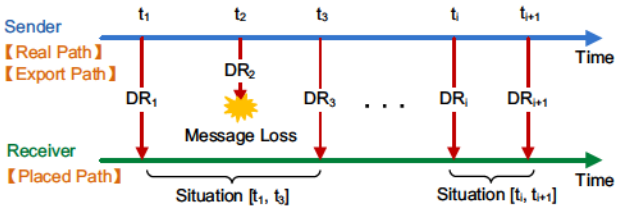


Fig. 1. System model of a DR-based update mechanism

The system model of a DR-based update mechanism is shown in Figure 1, where a sender sends DR updates to a receiver. The sender refreshes the entity position in real time according to user actions, e.g., accelerating, braking and changing orientation in a racing game. Following the terminologies defined by Aggarwal et al [12], the trajectory of the entity at the sender side is referred to as the *real path*. The position of the entity in its real path at time t is denoted as $\mathcal{R}(t)$. Besides that real path, the sender also keeps the *exported path*, i.e., the path extrapolated using previously sent DR updates. The position of the entity in its exported path at time t is denoted as $\mathcal{E}(t)$. The difference between real path and the exported path is referred to as the *DR error*. The DR error at time t denoted as $DE(t)$ is equal to $\mathcal{R}(t) - \mathcal{E}(t)$. Using the received DR updates, the receiver renders the trajectory of the entity, which is referred to as the *placed path*. The position of the entity in its placed path at time t is denoted as $\mathcal{P}(t)$. A deviation between the exported path and the placed path of the entity is referred to as the *export error*. The export error at time t denoted as $EE(t)$ is equal to $\mathcal{E}(t) - \mathcal{P}(t)$. The export error exists due to the clock asynchrony [8], [12], message delay [8], [14], [12], or message loss in the transmission [15]. The *real error* is defined as the inconsistency between the real path of the entity at the sender and the placed path at the receiver. The real error at time t denoted as $RE(t)$ is equal to $\mathcal{R}(t) - \mathcal{P}(t)$. It is composed of the DR error and the export error.

In this paper, we propose new DR-based update scheduling algorithms to reduce the real error of the entity in the presence of message loss. As mentioned in Section II, the impact of clock asynchrony and message delay can be handled by a global synchronized clock (e.g., [8], [12]) and local lag technique (e.g., [16]) respectively. For simplicity, we assume the clocks of the sender and receiver are synchronized and the message delay in the transmission can be ignored.

The TSI metric [8] is used in this paper to measure the inconsistency of the entity. On receiving the DR updates, the position of the entity is updated at the receiver. Thus, the real error is brought down to 0 at times of update receipts. Between updates receipts, the real error can be greater than 0. The time interval between two successive DR update receipts is referred to as a situation. Let t_b and t_e be the times of two successive update receipts. Then, the TSI of the situation [t_b, t_e] is calculated as:

$$\int_{t_b}^{t_e} RE(t)dt$$

Due to the constraint of network bandwidth, a sender can send DR updates up to only a certain number to its receivers at any given time. We assume that the underlying network supports unicast transmission only. Without enough bandwidth, the sender may fail to send the DR updates in time. In addition, the message loss rates in the transmission between the sender and receivers (for short, the message loss rates of receivers) may vary widely. Due to different message loss rates, the receivers may observe different levels of inconsistency for the same entity even if the same set of DR updates are sent by the sender to various receivers. Therefore, the sender has to decide which DR updates to send to which receivers to reduce the differences of inconsistency amongst receivers within the bandwidth constraint. In this paper, we investigate how to schedule DR updates for the following two objectives:

- $\mathcal{O}1$: obtain equal TSIs of an entity between the sender and various receivers irrespective of their different message loss rates.
- $\mathcal{O}2$: reduce the TSIs of an entity to the level of the case without message loss.

As shown in [8], the impact of inconsistency on the perceptions of participants is highly dependent on its TSI. Once the above objectives have been achieved, participants in the virtual world can compete fairly and enjoy the same experience as that without transmission failures.

IV. DR-based Update Scheduling

A. Impact of Message Loss

Consider the scenario that there is no bandwidth constraint for a sender to send DR updates of an entity to a receiver. First, we analytically derive impact of message loss on the TSI of the entity when using the original update scheduling algorithm. For simplicity, we shall assume one dimensional linear movements in our analysis. However, the analysis and algorithms presented in this paper can also be extended to multiple dimensional movements.

The real path of the entity can be modeled using Taylor series expansion approximately, because $\mathcal{R}(t)$ is assumed to be a continuous and differentiable function of the variable t [10]. Suppose that, a DR update DR_i of an entity is sent at time t_i. The real path of the entity right after t_i can be estimated using following equation, where v_i and a_i denote the velocity and acceleration of the entity at time t_i:

$$\mathcal{R}(t_i + x) \approx \mathcal{R}(t_i) + \frac{v_i}{1!}x + \frac{a_i}{2!} \cdot x^2$$

We assume that first order extrapolation formula is used in the DR-based update mechanism. Then, the exported path of the entity is given by:

$$\mathcal{E}(t_i + x) = \mathcal{R}(t_i) + v_i \cdot x$$

Therefore, the DR error is:

$$DE(t_i + x) = \mathcal{R}(t_i + x) - \mathcal{E}(t_i + x) \approx \frac{1}{2} \cdot a_i \cdot x^2$$

As mentioned in Section III, the sender maintains the real path and exported path of the entity. Hence, it can calculate the DR error at any time. In the original update scheduling algorithm, a DR update is sent once the DR error exceeds a predefined threshold δ. That is, the next DR update DR_{i+1} should be sent at time t_{i+1}, when:

$$DE(t_{i+1}) \approx \frac{1}{2} \cdot a_i \cdot (t_{i+1} - t_i)^2 = \delta$$

Thus, the duration between the two successively sent DR updates DR_i and DR_{i+1}, which is denoted as λ_i, can be calculated as:

$$\lambda_i = t_{i+1} - t_i = \sqrt{\frac{2\delta}{a_i}} \tag{1}$$

In the case that no DR update is lost, both DR_i and DR_{i+1} are received by the receiver. So, for any $0 \leq x \leq \lambda_i$, $RE(t_i + x) = DE(t_i + x)$. Thus, the TSI of the situation [t_i, t_{i+1}] is given by:

$$\Omega_i = \int_0^{\lambda_i} \frac{1}{2} \cdot a_i \cdot x^2 \, dx = \frac{1}{6} \cdot a_i \cdot \lambda_i^3 = \sqrt{\frac{2\delta^3}{9 \cdot a_i}} \tag{2}$$

As shown in Equations (1) and (2), the smaller the acceleration, the greater the length and the TSI of the situation. However, the time-averaged TSI of the situation, i.e., the ratio between TSI and length of the situation,

$$\frac{\Omega_i}{\lambda_i} = \frac{\delta}{3}$$

is only concerned with the threshold. It is not concerned with entity's kinetics, such as velocity and acceleration. Suppose that the threshold is a constant value over a given time period T, all situations during the period should have the same time-averaged TSIs. Therefore, the total TSI during the period can be calculated as follows:

$$\sum \Omega = \frac{\delta}{3} \cdot T \tag{3}$$

In the case that some DR updates are lost due to transmission failures, the real error of an entity generally increases as the receiver may not be able to use the most recently sent DR update to extrapolate the entity's position. In the case that updates DR_{i+1}, DR_{i+2}, $\cdots$, and DR_{i+k-1} are all lost, the two successively received DR updates are DR_i and DR_{i+k}.

Suppose that the threshold and acceleration of the entity is a constant value during the situation $[t_i,\ t_{i+k}]$, it follows from Equation 1 that the durations between two successively sent DR updates are all the same during the situation, i.e., $\lambda_i = \lambda_{i+1} = \cdots = \lambda_{i+k-1}$. Hence, the length of the situation $[t_i, t_{i+k}]$ is $k \cdot \lambda_i$. We can also get the placed path of the entity during the situation as follows:

$$\mathcal{P}(t_i + x) = \mathcal{R}(t_i) + v_i \cdot x$$

for any $0 \leq x \leq k \cdot \lambda_i$. Therefore,

$$RE(t_i + x) = \mathcal{R}(t_i + x) - \mathcal{P}(t_i + x) = \frac{1}{2} \cdot a_i \cdot x^2$$

and the TSI of the situation $[t_i, t_{i+k}]$ is:

$$\int_0^{k \cdot \lambda_i} \frac{1}{2} \cdot a_i \cdot x^2 \, dx = \frac{1}{6} \cdot a_i \cdot (k \cdot \lambda_i)^3 = k^3 \cdot \Omega_i$$

As described above, the situation $[t_i, t_{i+k}]$ occurs if DR_{i+1}, DR_{i+2}, $\cdots$, and DR_{i+k-1} are all lost and DR_i and DR_{i+k} are received. Assuming that the message loss rate is a constant value p, thus, the probability of the occurrence of situation $[t_i, t_{i+k}]$ is thus $p^{k-1} \cdot (1-p)$. Hence, the expected length and the expected TSI of the situation starting from t_i can be calculated as follows (refer to Appendix for the derivation of Equation 5):

$$\sum_{k=1}^{\infty} p^{k-1} \cdot (1-p) \cdot k \cdot \lambda_i = \frac{\lambda_i}{1-p} \tag{4}$$

$$\sum_{k=1}^{\infty} p^{k-1} \cdot (1-p) \cdot k^3 \cdot \Omega_i = \frac{1 + 4p + p^2}{(1-p)^3} \cdot \Omega_i \tag{5}$$

From the above two equations, we can then obtain the time-averaged TSI of the situation as follows:

$$\frac{\frac{1+4p+p^2}{(1-p)^3} \cdot \Omega_i}{\frac{\lambda_i}{1-p}} = \frac{1 + 4p + p^2}{(1-p)^2} \cdot \frac{\Omega_i}{\lambda_i} = \frac{1 + 4p + p^2}{(1-p)^2} \cdot \frac{\delta}{3}$$

Similar to the case without message loss, we can infer that the time-averaged TSI is only concerned with the threshold and message loss rate, which are both assumed to be constant values. Therefore, the total TSI over a given time period T can be calculated as:

$$\sum \Omega = \frac{1 + 4p + p^2}{(1-p)^2} \cdot \frac{\delta}{3} \cdot T \tag{6}$$

To illustrate the impact of message loss on inconsistency, the total TSI in the case of message loss is normalized by that in the case without message loss (Equation (3)) and then plotted as the "Original" curve in Figure 2. As can be seen, using the original update scheduling algorithm, the normalized total TSI increases sharply with increasing message loss rate. For this reason, new update scheduling algorithms are required to reduce the TSI in the presence of message loss.

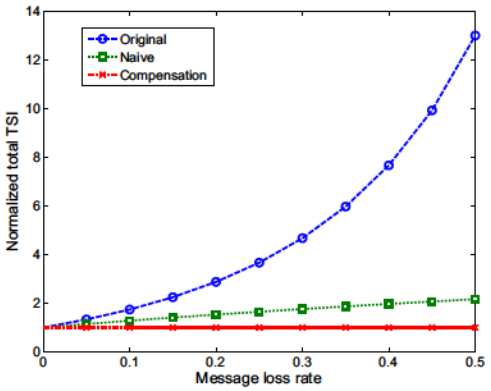


Fig. 2. Normalized total TSI versus message loss rate

B. Naive Algorithm

In the original update scheduling algorithm, a DR update is sent once the DR error exceeds the threshold. However, due to message loss, the real error, which is composed of the DR error and the export error is generally greater than the threshold at the time of DR update. Thus, an intuitive strategy to reduce inconsistency is to send the DR update once the real error exceeds the threshold. Unfortunately, the sender, which does not know the placed path of the entity, cannot calculate the exact value of the real error. Instead, the expected real error can be calculated based on previously sent DR updates and the message loss rate. Suppose that DR_i is the last sent DR update by the sender and DR_{i-j+1}, where $j \geq 1$, is the last received DR update at the receiver. Then, the receiver has to extrapolate the entity's placed path at time $t_i + x$ (the time right after t_i) using DR_{i-j+1}. Assume $RE(t, DR_k)$ denotes the real error at time t given that the last received DR update at the receiver is DR_k $(t > t_k)$. In the above case, $j-1$ successively sent DR updates, DR_{i-j+2}, $\cdots$, DR_{i-1} and DR_i, are lost. The probability for this case to occur is $p^{j-1} \cdot (1-p)$. Consequently, the expected $RE(t_i + x)$ can be calculated as follows:

$$\sum_{j=1}^{\infty} p^{j-1} \cdot (1-p) \cdot RE(t_i + x, DR_{i-j+1}) \tag{7}$$

Therefore, the next DR update DR_{i+1} should be sent at time t_{i+1} when:

$$\sum_{j=1}^{\infty} p^{j-1} \cdot (1-p) \cdot RE(t_{i+1}, DR_{i-j+1}) = \delta$$

We shall refer this algorithm as the naive update scheduling algorithm.

The duration between the two successively sent DR updates DR_i and DR_{i+1} in this algorithm is denoted as λ'_i, which is equals to $t_{i+1} - t_i$. In the case that the last received DR update at the receiver is DR_i, the real error at t_{i+1} equals the DR error at that time and is calculated as:

$$RE(t_{i+1}, DR_i) = DE(t_{i+1}) = DE(t_i + \lambda'_i) = \frac{1}{2} \cdot a_i \cdot \lambda'^2_i \tag{8}$$

In the case that the last received DR update is DR_{i-j+1}, the real error at time t_{i+1} is generally greater than the DR error at that time. Suppose that the message loss rate and acceleration are constant values during the period between t_{i-j+1} and t_{i+1}. The durations between any two successively sent DR updates are the same, i.e., $\lambda'_{i-j+1} = \cdots = \lambda'_{i-1} = \lambda'_i$. So, $t_{i+1} - t_{i-j+1} = j \cdot \lambda'_i$. Consequently, the real error at time t_{i+1} can be calculated as:

$$RE(t_{i+1}, DR_{i-j+1}) = \frac{1}{2} \cdot a_{i-j+1} \cdot (j \cdot \lambda'_i)^2 = \frac{1}{2} \cdot a_i \cdot (j \cdot \lambda'_i)^2$$

The probability for this case to occur, as mentioned above, is $p^{j-1} \cdot (1-p)$. Hence, the expected $RE(t_{i+1})$ can be calculated as (refer to Appendix for the derivation of Equation 9):

$$\sum_{j=1}^{\infty} p^{j-1} \cdot (1-p) \cdot \frac{1}{2} \cdot a_i \cdot (j \cdot \lambda'_i)^2 = \frac{1+p}{(1-p)^2} \cdot \frac{1}{2} \cdot a_i \cdot \lambda'^2_i \quad (9)$$

In the naive update scheduling algorithm, the next DR update DR_{i+1} is sent when the expected $RE(t_{i+1}) = \delta$. Hence, λ'_i can be calculated as:

$$\lambda'_i = \frac{1-p}{\sqrt{1+p}} \cdot \sqrt{\frac{2\delta}{a_i}} = \frac{1-p}{\sqrt{1+p}} \cdot \lambda_i \quad (10)$$

Because $0 \leq p \leq 1$, $(1-p)/\sqrt{1+p} \leq 1$, and thus $\lambda'_i \leq \lambda_i$. As a result, more DR updates are sent by the sender using the naive update scheduling algorithm than that using the original update scheduling algorithm.

If both DR_i and DR_{i+1} are received by the receiver, we can calculate the TSI of the situation $[t_i, t_{i+1}]$ denoted as Ω'_i, in a similar way as Ω_i calculated in Equation (2). That is,

$$\Omega'_i = \int_0^{\lambda'_i} \frac{1}{2} \cdot a_i \cdot x^2 \, dx = \frac{1}{6} \cdot a_i \cdot \lambda'^3_i. \quad (11)$$

Assuming that the message loss rate is a constant value p, the expected length and the expected TSI of the situation starting from t_i can also be calculated in a similar way as those calculated in Equations (4) and (5). Hence, the time-averaged TSI of the situation can be given by:

$$\frac{\frac{1+4p+p^2}{(1-p)^3} \cdot \Omega'_i}{\frac{\lambda'_i}{1-p}} = \frac{1+4p+p^2}{(1-p)^2} \cdot \frac{\Omega'_i}{\lambda'_i}$$

Using Equations (10) and (11), the above can be simplified as follows:

$$\frac{1+4p+p^2}{1+p} \cdot \frac{\delta}{3}$$

Therefore, the total TSI of the entity over a given time period T is:

$$\sum \Omega = \frac{1+4p+p^2}{1+p} \cdot \frac{\delta}{3} \cdot T \quad (12)$$

To compare with other scheduling algorithms, the above total TSI is normalized by that in the case without message loss (Equation (3)) and then plotted as the "Naive" curve in Figure 2. It can be seen that the total TSI of the naive update scheduling algorithm is much lower than that of the original update scheduling algorithm when there is message loss. However, the total TSI is still greater than that in the case of no message loss. Moreover, it still increases with increasing message loss rate.

In the implementation of the naive update scheduling algorithm, calculating the expected real error using Equation (7) directly could be time-consuming. In fact, it may not be necessary to calculate the expected real error based on all previously sent DR updates. The above theoretical analysis is also helpful to determine the number of previously sent DR updates which should be involved in the calculation of the expected real error. The value of $p^{j-1} \cdot (1-p) \cdot j^2$, a coefficients of the item in Equation (9), is negligible when $j > 16$ and $0\% \leq p \leq 50\%$. Therefore, when the message loss rate does not exceed 50%, the expected $RE(t_i + x)$ can be approximated by:

$$\sum_{j=1}^{16} p^{j-1} \cdot (1-p) \cdot RE(t_i + x, DR_{i-j+1})$$

In other words, the sender can estimate the expected real error using the past 16 DR updates sent.

C. Compensation Algorithm

As shown in Equation (6), the total TSI of the entity is determined by the message loss rate and the threshold. Hence, it is possible to reduce the TSI to the level of the case without message loss (Equation (3)) by reducing the threshold according to the message loss rate. To do so, the reduced threshold δ'' should satisfy:

$$\frac{1+4p+p^2}{(1-p)^2} \cdot \frac{\delta''}{3} \cdot T = \frac{\delta}{3} \cdot T$$

Hence, δ'' can be calculated as:

$$\delta'' = \frac{(1-p)^2}{1+4p+p^2} \cdot \delta \quad (13)$$

As a result, the duration between two successively sent DR updates DR_i and DR_{i+1} decreases to:

$$\begin{aligned} \sqrt{\frac{2\delta''}{a_i}} &= \frac{1-p}{\sqrt{1+4p+p^2}} \cdot \sqrt{\frac{2\delta}{a_i}} \\ &= \frac{1-p}{\sqrt{1+4p+p^2}} \cdot \lambda_i \end{aligned} \quad (14)$$

We shall refer to this algorithm as the compensation update scheduling algorithm. Using this algorithm, the total TSI is equal to that of the case without message loss. That is, the total TSI normalized by that of the case without message loss (Equation (3)) equals 1, as shown by the "Compensation" curve in Figure 2. Therefore, the receivers should enjoy the same experience in the virtual world regardless of whether and how frequent the messages are lost.

Compared with the naive update scheduling algorithm, the compensation algorithm is easier to implement. It is the same as the original algorithm except for reducing the threshold according to the message loss rate. Moreover, the compensation algorithm is able to reduce the TSI of the entity to that of the case without message loss and ensure the fairness among receivers with different message loss rates. However, DR updates are sent more frequently in the compensation algorithm (Equation (14)) than those in the naive algorithm (Equation (10)). In the next subsection, a budget-based mechanism is developed to deal with the network bandwidth constraint using these update scheduling algorithms.

D. Budget-based Mechanism

In a DVE, a sender is generally required to send DR updates of multiple entities to their corresponding interested receivers. Due to the constraint of network bandwidth, the sender can send DR updates up to only a certain number to the receivers at any given time. Now, we develop a budget-based mechanism to send DR updates selectively within the bandwidth constraint.

Specifically, a priority is calculated for each combination of an entity and a receiver. Suppose that DR_i is the last sent DR update of the entity to the receiver and DR_i is sent at time t_i. Using the original update scheduling algorithm, the next DR update DR_{i+1} is sent when $DE(t_i + x)$ exceeds δ. Accordingly, the priority at time $t_i + x$ is defined as:

$$\frac{DE(t_i + x)}{\delta}$$

Using the naive algorithm, the next DR update DR_{i+1} is sent when the expected $RE(t_i + x)$ exceeds δ. Accordingly, the priority at time $t_i + x$ is defined as:

$$\frac{Expected\ RE(t_i + x)}{\delta}$$

Using the compensation algorithm, the next DR update DR_{i+1} is sent when $DE(t_i+x)$ exceeds δ''. Accordingly, the priority at time $t_i + x$ is defined as:

$$\frac{DE(t_i + x)}{\delta''} = \frac{(1 + 4p + p^2)}{(1 - p)^2} \cdot \frac{DE(t_i + x)}{\delta}$$

Only when the priority is greater than 1, it is necessary to send the DR update of the entity to the corresponding receiver. Suppose that the sender can send up to W DR updates to the receivers at a time. In the case that more than W combinations of entities and receivers have priorities greater than 1, the W combinations with the highest priorities are chosen and the DR updates corresponding to these combinations are sent. After sending the updates, the priorities of these W combinations would drop to 0. Thus, the remaining combinations would be more likely to be chosen in the future. If fewer than W combinations have priorities greater than 1, only part of the network bandwidth is used to send DR updates for all these combinations with priorities greater than 1.

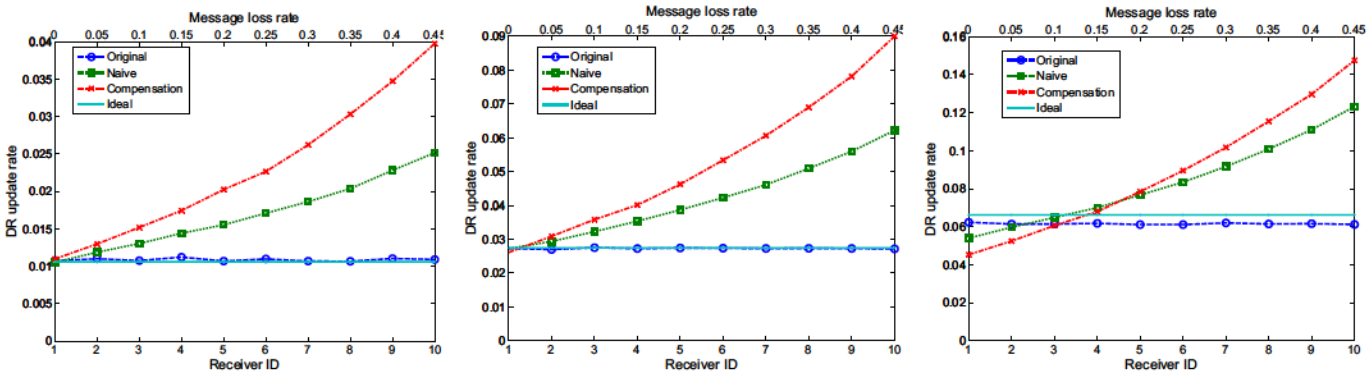
With enough bandwidth available, DR updates can be sent for all combinations with priorities greater than 1. Otherwise, DR updates are sent only for combinations with priorities greater than a value h (which is greater than 1). This has the same effect on the inconsistency of the entities as increasing the DR threshold from δ to $h \cdot \delta$. For the budget-based mechanism using the compensation update scheduling algorithm, the time-averaged TSIs of an entity at receivers with different message loss rates are all close to $\frac{\delta}{3}$ if enough bandwidth is available. Therefore, we can infer that objectives $\mathcal{O}1$ and $\mathcal{O}2$ mentioned in section III are achieved. If there is not enough bandwidth, the time-averaged TSIs of an entity at the receivers are increased to $h \cdot \frac{\delta}{3}$. Therefore, objective $\mathcal{O}1$ is still achieved. However, the objective $\mathcal{O}2$ is not achieved due to the insufficient number of DR updates sent. For the budget-based mechanisms using the original and naive update scheduling algorithms, the TSIs of an entity at the receiver with message loss rate p are close to Equations (6) and (12) respectively if enough bandwidth is available. Otherwise, the TSIs increase to h times of Equations (6) and (12) respectively. Therefore, the TSIs never reduce to the level in the case of no message loss and the receivers with higher message loss rates have greater TSIs. Hence, neither of objectives $\mathcal{O}1$ nor $\mathcal{O}2$ is achieved.

V. Experiment and Results

A. Experiment Design

In order to evaluate and compare the update scheduling algorithms proposed in this paper, simulation experiments are carried out using the traces of car positions in a car racing game called TORCS [22]. In TORCS, the position of a car is described by its X,Y and Z coordinates in the three dimensional virtual world. Since the cars can only run along a particular track, for simplicity, we model the track as one dimensional space and map the car positions into this space by calculating the distance (in meters) travelled by the car along the track. Using the calculated distance, we can also obtain the positional relations among different cars (i.e., which car is in front of another) that affect players most significantly in a racing game. There are $32,000$ recordings of car positions in each trace. The time interval between two successive position recordings in the traces is 0.02 second, which is assumed to be the length of a frame.

A simulator is developed to simulate 11 players competing in an online car racing game. The players communicate with each other in a peer to peer manner. Each player controls one car and sends DR updates of the car to all other players. The players are assumed to have synchronized clocks and the transmission delays of DR updates are assumed to be negligible. In the simulation, we focus on the DR updates from the first player to the other 10 players. The first player is referred to as the sender, whereas the other 10 players are referred to as the receivers with IDs from 1 to 10. The message loss rates of receivers 1, 2, $\cdots$, 10 are assumed to be 0%, 5%, $\cdots$, 45% respectively. The traces of two different cars collected from a TORCS game are used. One trace is used to simulate the car controlled by the sender. The other trace is used to simulate the cars controlled by the receivers. To compare the impact of message loss on consistency, the same



(a) Threshold = 5 (b) Threshold = 1 (c) Threshold = 0.2

Fig. 3. DR Update rate of receivers in different scenarios

trace is used to simulate the cars controlled by all receivers that have different message loss rates.

Four scenarios are investigated in our experiments. In the Ideal scenario, the original update scheduling algorithm is used without bandwidth constraint and message loss. In the Original, Naive and Compensation scenarios, the budget-based mechanism is applied, using the original, naive, and compensation update scheduling algorithms respectively. We assume that due to the bandwidth constraint, the sender can only send a DR update of its car to one receiver in each frame.

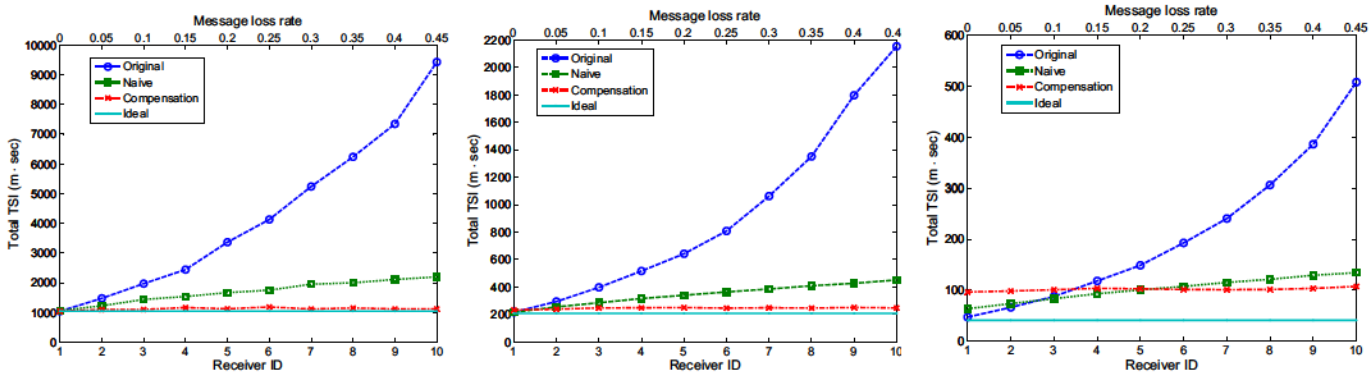
B. Experiment Results

The DR update rate of a receiver is defined as the ratio between the number of DR updates sent by the sender to the receiver and the total number of frames in the simulation. Figure 3 shows the update rates of receivers in different scenarios for different thresholds. With decreasing threshold, the priorities in all scenarios increase, and thus the numbers of DR updates sent by the sender to all receivers increase. In the Original scenario, message loss is not taken into account in update scheduling. Thus, similar numbers of DR updates are sent to all receivers just like the Ideal scenario. Different from the Original scenario, in the Compensation and Naive scenarios, the priorities of the combinations of sender-controlled car and receivers increase with increasing message loss rate. Hence, larger numbers of DR updates are sent to receivers with higher message loss rates. We can further observe from Figure 3 that this trend is more significant in the Compensation scenario. The number of DR updates sent to receivers is greater in the Compensation scenario when enough bandwidth is available, e.g., at thresholds 5 and 1 (see Figures 3(a) and 3(b)).

Figure 4 shows the total TSIs of receivers in different scenarios for different thresholds. With decreasing threshold, the number of DR updates sent to receivers increases, and thus the TSIs of the sender-controlled car at the receivers decrease. In the Ideal scenario, no DR update is lost, and thus the TSIs of different receivers are the same. The TSIs in the Original scenario increase dramatically with respect to the message loss rates of the receivers. The TSIs are reduced significantly in the Naive and Compensation scenarios due to higher DR update rates.

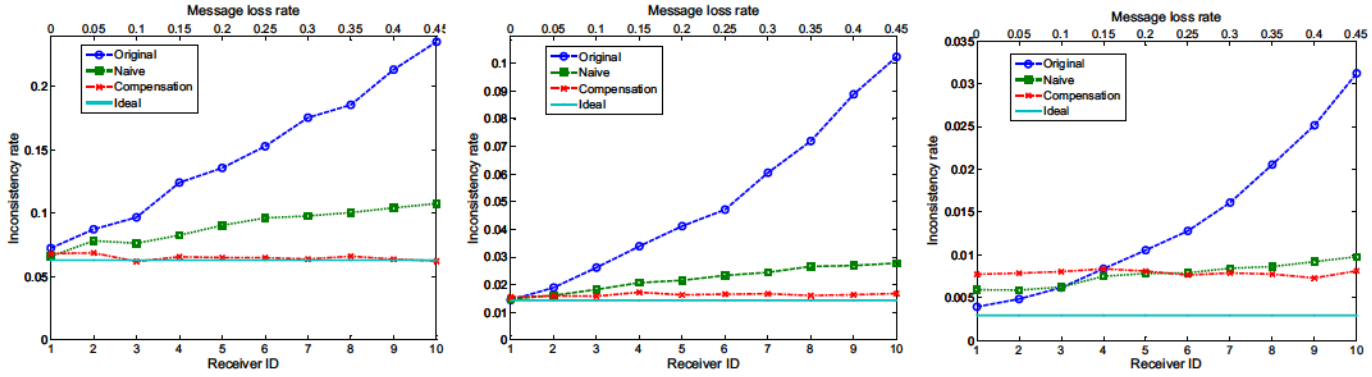
For the Compensation scenario, we can observe that the TSIs of all receivers are comparable. The TSIs are close to those in Ideal scenario when the threshold is equal to 5 or 1, but are greater than those in the Ideal scenario when the threshold is equal to *0.2*. These observations are consistent with the analysis at the end of Section IV. That is, the budget-based mechanism using the compensation update scheduling algorithm can ensure the fairness among receivers in terms of the TSIs of the sender-controlled car. With enough bandwidth available, it can reduce the TSIs to the level of the case without message loss. For the Naive scenario, we can observe from Figure 4 that the TSIs of receivers increase with increasing message loss rates and never reduce to the levels in the Ideal scenario. These observations also agree with the analysis at the end of Section IV. That is, the budget-based mechanism using the naive update scheduling algorithm can neither ensure fairness among receivers nor reduce the TSIs to the level of the case without message loss.

When the threshold is equal to *0.2*, the bandwidth constraint is critical. Almost all available bandwidth is used in the Naive and Compensation scenarios. As mentioned above, the number of DR updates increases more significantly in the Compensation scenario with increasing message loss rate. Therefore, we can observe from Figure 3(c) that, compared with the Naive scenario, more DR updates are sent in the Compensation scenario to those receivers with message loss rates $\geq 25\%$; whereas fewer DR updates are sent to receivers with message loss rates $< 25\%$. As a result, we can observe from Figure 4(c) that, compared with the Naive scenario, the TSIs of receivers with high message loss rates are smaller in the Compensation scenario; whereas the TSIs of receivers with low message loss rates are greater. In summary, although similar bandwidth is used in the Naive and Compensation scenarios, the sender in the Compensation scenario sends more DR updates to those receivers with high message loss rates for the purpose of reducing their TSIs to the levels of those receivers with low message loss rates.



(a) Threshold = 5 (b) Threshold = 1 (c) Threshold = 0.2

Fig. 4. Total TSI of receivers in different scenarios



(a) Threshold = 5 (b) Threshold = 1 (c) Threshold = 0.2

Fig. 5. Inconsistency rate of receivers in different scenarios

A receiver can observe the real path of its car and the placed path of the car controlled by the sender. The observed positional relation of these two cars might be different from the real positional relation, which is determined by the real paths of the cars. If the observed and real positional relations are different in a frame, the frame is referred to as an inconsistent frame. The inconsistency rate is defined as the ratio between the number of inconsistent frames and the total number of frames in the simulation. In general, the receivers with higher inconsistency rates observe incorrect positional relations more frequently, and thus is less favorable in terms of the fairness.

Figure 5 shows the inconsistency rates of receivers in different scenarios for different thresholds. As can be seen, the smaller the threshold, the lower the inconsistency rates. Figures 4 and 5, show that the trends of inconsistency rates are highly consistent with those of the corresponding TSIs. Compared to the Original scenario, the inconsistency rates of receivers are reduced significantly in the Naive and Compensation scenarios. In the Compensation scenario, all receivers have almost the same inconsistency rates, which are close to those in the Ideal scenario when the threshold is equal to 5 or 1, but are greater than those in the Ideal scenario when the threshold is equal to *0.2*. However, in the Naive scenario, the inconsistency rates of receivers increase with increasing message loss rates and are always higher than those in the Ideal scenario.

In summary, the budget-based mechanism using the compensation update scheduling algorithm ensures fairness among receivers. With enough bandwidth available, it can promise players the same gaming experience as in the case of no message loss.

VI. Conclusions and Future Work

In this paper, we analytically derive the impact of message loss on TSI of an entity in DVE when using a DR-based update mechanism. To reduce the TSI, the naive and compensation update scheduling algorithms are proposed, taking message loss in the transmission into account. Using these update scheduling algorithms, a budget-based mechanism is developed to send DR updates selectively within network bandwidth constraint. According to our theoretical analysis and experiment results, the budget-based mechanism using the compensation algorithm has advantages over that using the naive algorithm. It can provide fairness among participants in spite of widely varying message loss rates. With enough network bandwidth available, it can promise participants the same experience in the virtual world as in the case without message loss.

In the future, we will extend our work to different movement styles of entities in DVEs and more complex DR extrapolation formulas. Moreover, the message transmission delay and clock synchronization among participants will also be considered.

ACKNOWLEDGMENT

This work was supported in part by Nanyang Technological University under Academic Research Fund Tier 1 Grant RG14/08.

REFERENCES

[1] S. Singhal and M. Zyda, *Networked Virtual Environments: Design and Implementation*. Addison-Wesley, 1999.

[2] D. Miller and J. Thorpe, "Simnet: the advent of simulator networking," *Proc. IEEE*, vol. 83, no. 8, pp. 1114–1123, 1995.

[3] T. Nitta, K. Fujita, and S. Kohno, "An application of distributed virtual environment to foreign language education," in *Procs of the 30th Annual Frontiers in Education - Volume 01*, 2000, pp. 9–15.

[4] T. Hampel, T. Bopp, and R. Hinn, "A peer-to-peer architecture for massive multiplayer online games," in *Procs of 5th ACM SIGCOMM workshop on Network and system support for games (NetGames'06)*, 2006.

[5] IEEE, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (2007 revision)*, 2007.

[6] K.-C. Lin, "Dead reckoning and distributed interactive simulation," in *Procs of SPIE Conference (AeroSense'95)*, 1995.

[7] W. Cai, F. B. S. Lee, and L. Chen, "An auto-adaptive dead reckoning algorithm for distributed interactive simulation," in *Procs of the 13th workshop on Parallel and distributed simulation (PADS'99)*, 1999, pp. 82–89.

[8] S. Zhou, W. Cai, B.-S. Lee, and S. J. Turner, "Time-space consistency in large-scale distributed virtual environments," *ACM Trans. Model. Comput. Simul.*, vol. 14, pp. 31–47, January 2004.

[9] S. Aggarwal, H. Banavar, S. Mukherjee, and S. Rangarajan, "Fairness in dead-reckoning based distributed multi-player games," in *Procs of 4th ACM SIGCOMM workshop on Network and system support for games (NetGames'05)*, 2005, pp. 1–10.

[10] D. Hanawa and T. Yonekura, "A proposal of dead reckoning protocol in distributed virtual environment based on the taylor expansion," in *Procs of the 2006 International Conference on Cyberworlds*, 2006, pp. 107–114.

[11] D. Roberts, D. Marshall, R. Aspin, S. McLoone, D. Delaney, and T. Ward, "Exploring the use of local consistency measures as thresholds for dead reckoning update packet generation," in *Procs of the 9th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT '05)*, 2005, pp. 195–202.

[12] S. Aggarwal, H. Banavar, A. Khandelwal, S. Mukherjee, and S. Rangarajan, "Accuracy in dead-reckoning based distributed multi-player games," in *Procs of the 3rd ACM SIGCOMM workshop on Network and system support for games (NetGames'04)*, 2004, pp. 161–165.

[13] L. Pantel and L. C. Wolf, "On the impact of delay on real-time multiplayer games," in *Procs of the 12th international workshop on Network and operating systems support for digital audio and video (NOSSDAV'02)*, 2002, pp. 23–29.

[14] L. Pantel and L. C. Wolf, "On the suitability of dead reckoning schemes for games," in *Procs of the 1st workshop on Network and system support for games (NetGames'02)*, 2002, pp. 79–84.

[15] T. Yasui, Y. Ishibashi, and T. Ikedo, "Influences of network latency and packet loss on consistency in networked racing games," in *Procs of 4th ACM SIGCOMM workshop on Network and system support for games (NetGames'05)*, 2005, pp. 1–8.

[16] A. Malik Khan, S. Chabridon, and A. Beugnard, "A dynamic approach to consistency management for mobile multiplayer games," in *Procs of the 8th international conference on New technologies in distributed systems (NOTERE'08)*, 2008, pp. 42:1–42:6.

[17] G. Cheung and T. Sakamoto, "Construction and scheduling of extrapolated parity packets for dead reckoning in network gaming," in *Procs of the 6th ACM SIGCOMM workshop on Network and system support for games (NetGames'07)*, 2007, pp. 61–66.

[18] C. Faisstnauer, D. Schmalstieg, and W. Purgathofer, "Priority scheduling for networked virtual environments," *IEEE Comput. Graph. Appl.*, vol. 20, pp. 66–75, November 2000.

[19] Y. Yu, Z. Li, L. Shi, Y.-C. Chen, and H. Xu, "Network-aware state update for large scale mobile games," in *Procs of the 16th International Conference on Computer Communications and Networks*, 2007, pp. 563–568.

[20] X. Tang and S. Zhou, "Update scheduling for improving consistency in distributed virtual environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, pp. 765–777, June 2010.

[21] Y. Li and W. Cai, "Consistency aware dead reckoning threshold tuning with server assistance in client-server-based dves," in *Procs of 10th International Conference on Computer and Information Technology*, 2010, pp. 2925–2932.

[22] The Open Racing Car Simulator (TORCS). [Online]. Available: http://torcs.sourceforge.net/index.php

APPENDIX

According to the infinite geometric series formula, we can get the following equation, when $0 \leq p < 1$,

$$\sum_{k=1}^{\infty} p^k = \frac{p}{1-p}$$

Consequently, we can get the following equations:

$$\sum_{k=1}^{\infty} k \cdot p^k = p \cdot \sum_{k=1}^{\infty} k \cdot p^{k-1} = p \cdot \frac{\partial(\sum_{k=1}^{\infty} p^k)}{\partial p} = p \cdot \frac{\partial(\frac{p}{1-p})}{\partial p} = \frac{p}{(1-p)^2}$$

$$\sum_{k=1}^{\infty} k^2 \cdot p^k = p \cdot \sum_{k=1}^{\infty} k^2 \cdot p^{k-1} = p \cdot \frac{\partial(\sum_{k=1}^{\infty} k \cdot p^k)}{\partial p} = p \cdot \frac{\partial(\frac{p}{(1-p)^2})}{\partial p} = \frac{(1+p) \cdot p}{(1-p)^3}$$

$$\sum_{k=1}^{\infty} k^3 \cdot p^k = p \cdot \sum_{k=1}^{\infty} k^3 \cdot p^{k-1} = p \cdot \frac{\partial(\sum_{k=1}^{\infty} k^2 \cdot p^k)}{\partial p} = p \cdot \frac{\partial(\frac{(1+p)\cdot p}{(1-p)^3})}{\partial p} = \frac{p \cdot (1+4p+p^2)}{(1-p)^4}$$

Based on the above equations, we can then derive Equations 15 and 16 respectively. These two equations are used in the derivation of Equations 5 and 9 respectively.

$$\sum_{k=1}^{\infty} p^{k-1} \cdot (1-p) \cdot k^3 = \frac{1-p}{p} \cdot \sum_{k=1}^{\infty} k^3 \cdot p^k = \frac{1+4p+p^2}{(1-p)^3} \quad (15)$$

$$\sum_{k=1}^{\infty} p^{k-1} \cdot (1-p) \cdot k^2 = \frac{1-p}{p} \cdot \sum_{k=1}^{\infty} k^2 \cdot p^k = \frac{1+p}{(1-p)^2} \quad (16)$$