National University of Singapore
School of Computing

CS4344 **Problem Set 1: State Consistency** Semester 2 14/15

1. Explain what are the effects (if any) of high network delay jitter on the game play when the following techniques are used. Explain if you answer would be different when clocks between the clients and the server are synchonized/not synchronized.
   (a) permissible client/server architecture: a client always waits for acknowledgment from the server before updating its local state.
   (b) artificial lag: server postpones processing of client messages to improve fairness among the clients.
   (c) lag compensation: server rewind to a previous state to determine if a player has hit another player.
   (d) local perception filter: rendering of passive objects are time-shifted to create consistent view among the players.
   (e) bucket synchronization with dead reckoning: events are collected into buckets and processed at fixed time interval.
   (f) stop-and-wait: events are collected into buckets and processed, but messages from all players must be received before the game simulation proceeds.

2. Instead of dead reckoning, someone proposed a history-based prediction method. When update from B is not available, player A extrapolates the next position of B, by looking at the two last positions of the object. Based on these two positions and the time it takes for the object to move from one position to the other, the velocity (speed and direction) of the object can be computed, and therefore the next position of the object can be predicted.

   What are the advantages and disadvantages of history-based prediction?

3. An issue that needs to be addressed when using dead reckoning is the value of the distance error threshold τ, i.e., the threshold of the distance between actual position and predicted position above which an update needs to be sent. When two players are close to each other in the game world, accuracy is important and therefore the threshold should be small. For two players that are further apart, the threshold can be larger.

   Now consider how such adaptive dead reckoning scheme can be used in the following scenario. We have a centralized client/server architecture, with a server s, and clients (i.e., players) p_0, p_1, ..., p_n. We may assume that the latency between the clients and the server is negligible. Let $d(i, j)$ be the distance between player p_i and p_j in the game world, and $\tau(i, j)$ be the error threshold between p_i and p_j. The game designers have determined that the following relationship between error threshold and players distance is suitable for their game.

   $$\tau(i, j) = 0.2 * d(i, j) \quad (1)$$

   A player always updates the server s whenever its velocity changes. When s receives an update from a player p_i, s uses adaptive dead reckoning to decide whether to forward this update to another player p_j ($i \neq j$) based on $\tau(p_i, p_j)$ and the current prediction error of p_i's position at p_j.
   (a) Explain how the adaptive dead reckoning approach above can lead to huge computational overhead at s.
   (b) Suggest how the computation of error threshold can be modified to reduce the computational overhead at s, while still being adaptive to distance between players.
   (c) In the scenario above, should a player use dead reckoning between itself and the server? Justify your answer.

4. Assuming that all hosts have synchronized clocks, the error introduced by dead reckoning consists of three components: (a) error due to dead reckoning threshold, (b) error during convergence, and (c) error caused by network delay.
   (a) Which component above causes unfairness among players?
   (b) Suggest how we can improve the fairness among the players by tweaking the dead reckoning algorithm.