# DYNAMIC HYBRID STRATEGY MODELS FOR NETWORKED MULTIPLAYER GAMES

Aaron McCoy, Seamus McLoone, Tomás Ward
Department of Electronic Engineering
National University of Ireland, Maynooth
Maynooth, Co. Kildare, Ireland
E-mail: amccoy@eeng.nuim.ie

Declan Delaney
Department of Computer Science
National University of Ireland, Maynooth
Maynooth, Co. Kildare, Ireland

**KEYWORDS**
Distributed Interactive Applications, Hybrid Strategy Model, Networked Multiplayer Computer Games, Dead Reckoning, Packet-Reduction, User-Modelling.

**ABSTRACT**

Two of the primary factors in the development of networked multiplayer computer games are network latency and network bandwidth. Reducing the effects of network latency helps maintain game-state fidelity, while reducing network bandwidth usage increases the scalability of the game to support more players. The current technique to address these issues is to have each player locally simulate remote objects (e.g. other players). This is known as dead reckoning. Provided the local simulations are accurate to within a given tolerance, dead reckoning reduces the amount of information required to be transmitted between players. This paper presents an extension to the recently proposed Hybrid Strategy Model (HSM) technique, known as the Dynamic Hybrid Strategy Model (DHSM). By dynamically switching between models of user behaviour, the DHSM attempts to improve the prediction capability of the local simulations, allowing them to stay within a given tolerance for a longer amount of time. This can lead to further reductions in the amount of information required to be transmitted. Presented results for the case of a simple first-person shooter (FPS) game demonstrate the validity of the DHSM approach over dead reckoning, leading to a reduction in the number of state update packets sent and indicating significant potential for network traffic reduction in various multiplayer games/simulations.

## INTRODUCTION

Networked multiplayer computer games are one of the most important areas of an already burgeoning computer games industry. In a formal domain such applications belong to a class more commonly known as Distributed Interactive Applications (DIAs). These systems typically involve many users simultaneously interacting in a simulated virtual environment. Game designers are constantly seeking to scale such applications to more and more simultaneous users whilst still maintaining a high quality of interactivity and responsiveness. However, a number of technical problems combine to make delivery of such an experience difficult (Singhal and Zyda 1999; McCoy et al. 2003). One such problem is latency, which is the time it takes for information to propagate across the network to all participants. Another closely related issue is the problem of network bandwidth. Within a DIA we refer to these problems as the information updating issue. Several methods have been devised to reduce the quantity of data that needs to be transmitted between participants. The standard for Distributed Interactive Simulation (DIS) defines one such method known as dead reckoning (IEEE 1995), a form of client predictive contract mechanism.

Recently, an alternative technique known as the Hybrid Strategy Model (HSM) has been introduced, which offers an improvement over the performance of dead reckoning (Delaney et al. 2003; Delaney 2004). It is a hybrid predictive contract technique, which dynamically switches between a short-term dead reckoning model and a longer-term user behavioural model, allowing quasi-deterministic modelling of an entity's dynamics. This can reduce the number of communicated update packets required for remote modelling of entities when compared to the use of a pure dead reckoning model alone.

This paper demonstrates how to apply the HSM technique when behavioural models must be recalculated during run-time as users pursue a dynamic goal. The original exposition of the HSM method was only ever demonstrated with fixed navigational goals. In section two of this paper we describe predictive contract mechanisms as used for information updating in DIAs. Existing solutions to this issue, including dead reckoning, are outlined. In particular the HSM technique is summarized. Section three describes our extension of this proposed hybrid switching technique, referred to as a Dynamic Hybrid Strategy Model (DHSM), and its implementation is discussed in section four. Several test environments were developed to compare this new technique with existing dead reckoning techniques. These environments are described in section five. Example simulation results are presented in section six for both the DHSM and dead reckoning techniques. Finally, the paper ends with the conclusions and suggestions for future research.

## PREDICTIVE CONTRACT MECHANISMS

The most common solution to the information updating issue within DIAs involves a client-side predictive contract mechanism called dead reckoning (IEEE 1995). All participating clients agree to maintain the same low-order local models of the dynamics of all other participating entities. This is the contract. Each participant also maintains a model of its own entity dynamics, which it continuously compares to its actual dynamics. When these differ by a pre-defined error-threshold amount, update information is broadcast to all other participants, who then proceed to update their models for that entity based on this newly received state information. Convergence algorithms are typically incorporated to allow a natural transition to occur between the modelled and actual motion when update data arrives.

Alternative methods have also been explored, and these include relevance/area-of-interest filtering and multicast groups (Rak and van Hook 1996), packet bundling (Liang et al. 1999), data compression (Van Hook et al. 1994), time management, priority scheduling and visibility culling (Faisstnauer et al. 2000). A comprehensive overview of such consistency maintenance measures can be found in (Delaney 2004). In addition, various enhancements have also been proposed for use with the standard dead reckoning algorithm, including adaptive error-thresholds (Lee et al. 1999; Shim and Kim 2001), multi-step dynamic curve fitting (Singhal and Cheriton 1994) and pre-determination of likely error-threshold deviations (Zhang et al. 2004).

### The Hybrid Strategy Model

The Hybrid Strategy Model (HSM) as proposed in (Delaney et al. 2003) is a predictive model of the following form:

$$M = px + (1 - p)\Gamma$$

where $x$ is any conventional dead reckoning model, $\Gamma$ is a long-term model of entity behaviour and $p$ is a binary weighting factor governed by:

$$p = 1, \; for \; \|E - \Gamma\| \geq \theta$$
$$= 0, \quad otherwise$$

where $\theta$ represents a distance measure threshold between the actual behaviour $E$ and the long-term model.

The model given by $M$ is used by participating clients in a DIA. The parameters and initial entity state used by the model are updated every time the state deviates from the true state by a predefined threshold amount $T_m$.

It has been shown that this approach leads to a reduction in the number of packets that need to be transmitted in a DIA for a given consistency. However up to now this approach has only ever been applied to DIAs where the long-term models are fixed navigational behaviours in which an entity is traversing an environment. To apply the HSM when a user's behaviour is reacting to a changing goal (originally defined as a dynamic goal in Delaney's original paper) is non-trivial. In such cases, suitable models of user behaviour are required which are clearly complex functions of, among other things, the environment, user experience and intention, and user ability. Nonetheless, for a restricted application such as an FPS many of these dependencies can be modelled adequately for HSM purposes in certain situations. In the following two sections, our approach to applying the HSM technique to dynamic goals in a realistic FPS is demonstrated.

## THE DYNAMIC HYBRID STRATEGY MODEL

A Dynamic Hybrid Strategy Model (DHSM) consists of a set of $i$ candidate prediction models given by:

$$M_{candidate} = \{m_1, \ldots\ldots, m_i\}$$

In addition, a default model is chosen to provide initial prediction for the DHSM (typically a short-term dead reckoning model would be chosen as default, though this is not a necessity):

$$m_{initial} = m_{default}, \quad \{ m_{default} \in M_{candidate} \}$$

During each simulation step, the DHSM has a currently selected *active* model (beginning with the initial model) that is used to perform actual prediction and entity-state-update (ESU) packet generation for a user. However, the DHSM also runs prediction for each candidate model in parallel with the active model and records each associated prediction error for every simulation step:

For $m_{current} \in M_{candidate}$
- Predict $P(m_{current})$
- Compute Error $E(m_{current})$

Endfor

When the current prediction error for the *active* model exceeds the allowable error-threshold, an ESU packet is required to be generated and transmitted. At this point, the average prediction error (APE) since the *last* transmitted ESU packet is computed for each candidate prediction model. This assigns a score to each model based on overall performance between consecutive ESU packet transmissions. The model that exhibits the best score (i.e. lowest APE) is then chosen to be the current *best* model.

If $E(m_{active}) \geq ErrorThreshold$

For $m_{current} \in M_{candidate}$

- Compute avg. error $APE\ (m_{current})$

Endfor

$m_{best}$ = Argmin $(APE(m_{current}), \forall m_{current})$

If $m_{active} \neq m_{best}$

- $m_{active} = m_{best}$

Endif

Endif

The currently selected active model is switched over to the determined best model if necessary (i.e. if we are not already using the best model), and an ESU packet is generated and transmitted. Appended onto the end of the packet is an identifier number indicating which candidate model is now in use. When remote clients receive this ESU packet, they can use this ID to switch their own local models of prediction in accordance with the new active model. This process is repeated every time the allowable error-threshold is exceeded, resulting in dynamic switching between prediction models based on their current performance in relation to one another.

## DHSM IMPLEMENTATION

The DHSM implementation presented in this paper utilizes a candidate set of just two prediction models: *dead reckoning* and *shortest-path*.

### Dead-Reckoning (DR)

This is the standard first-order, one-step short-term dead-reckoning prediction algorithm as originally detailed in the Standard for Distributed Interactive Simulations (DIS) (IEEE 1995). It is given by the following equation of motion:

$$P = P_0 + V_0 \Delta t$$

where $P_0$ and $V_0$ refer to initial position and velocity respectively and $\Delta t$ is the time increment for the current prediction step.

### Shortest-Path (SP)

This is a simple dynamic first-order, one-step behavioural model that assumes a user will move along the shortest-path from the current position to the target position (i.e. their dynamic goal) with constant speed. It is given by the following equation of motion:

$$P = P_0 + (T - P_0)\left(\frac{\|V_0\| \Delta t}{\|T - P_0\|}\right)$$

where $P_0$ and $V_0$ refer to initial position and velocity respectively, $\Delta t$ is the time increment for the current prediction step and $T$ refers to the current target (dynamic goal) position.

The design of this shortest-path model was driven by previous work that we have done in analysing the behaviour of users within various DIAs (and in particular networked multiplayer computer games) (McCoy et al. 2004a; McCoy et al. 2004b). It is grounded in the notion that within these networked games, users often exhibit tendencies to move directly closer to their target of interest when attempting to engage, thereby maximising their probability of disabling an opponent. During these time periods, correlation can often be observed between both sets of data, indicating a tight coupling of behaviour for both players. It is this coupling of data that we are exploiting with the shortest-path model in order to define a user's behaviour in terms of their dynamic goal. This allows us to reduce the transmission of state data required to accurately predict the user using a remote model.

For the implementation of the DHSM, we have also included an ancillary *line-of-sight* score function used to weight the selection of the shortest-path model over the dead reckoning model. It states that unless there is a *direct line-of-sight between a user and their dynamic goal*, the DHSM will not switch from using the DR model to the SP model, even if the SP model would have provided a better average prediction error since the last received ESU packet. The motivation behind this is the assumption that if a user cannot see the dynamic goal, the probability of rapidly changing behaviour occurring is far greater, thus decreasing the chance that the SP model will provide any benefit for short-term prediction over that provided by the standard DR model.

## DATA COLLECTION

We utilize the Torque Game Engine for performing experiments and data collection (Marshall et al. 2004). During a user's interaction within a test environment, data is sampled at a rate of 20Hz and consists of position, velocity, and forward facing direction vectors for each player (human or AI controlled). Discrete events are recorded for the cases of a player being disabled, firing their weapon or (in the case of a human user) interacting directly with the control device (i.e. mouse and/or keyboard). For each game tick, a trace can be performed to determine if there is an unobstructed line-of-sight (LOS) between any two players. This trace information is combined with the forward facing direction vectors in a post-processing step to determine *onscreen* LOS status for each sampling time.

Figure 1 shows plan (top-down) views of the three environments used for the experiments. In each case, a single test subject was asked to play against a single *non-reactive* computer-controlled opponent (BOT). The goal here was to disable the BOT a desired number of times as fast as possible. Each environment consisted of a unique *path network* that was used by the BOT for constant circumnavigation. The motivation behind this particular set of test scenarios was to ensure that the user's dynamic goal (in this case the BOT) was known

in advance, thus negating any possible requirement for performing some kind of target identification procedure (as would be the case if there were multiple independent dynamic goals for a user to choose from).

Figure 1: Test environment plan views showing path networks

## RESULTS AND ANALYSIS

Presented below in Tables 1-3 are simulation results for packet number, average prediction error and standard deviation error respectively for a collection of three datasets using a variety of increasing error-thresholds (to put the error-thresholds into perspective, the height of a player within our test environments is approximately 2.3 units). Each dataset was recorded using a different test subject and test environment (the particular environment is noted directly above the results in the tables). The term 'DR' refers to the standard first-order, one-step dead reckoning prediction model and is used as the base comparison with which to compare results. The term 'DHSM' refers to the Dynamic Hybrid Strategy Model using both the standard first-order, one-step dead reckoning prediction model and the first-order, one-step shortest-path prediction model (as outlined previously). The term '% Red' refers to the percentage reduction (or increase) in the associated variable (number of packets sent, average prediction error or standard deviation error). A negative value indicates better performance with respect to the Dynamic Hybrid Strategy Model in direct comparison with the standard dead reckoning model. A heartbeat timeout of 5 seconds was set for both the DR and DHSM models (meaning if the error-threshold has not been exceeded for 5 seconds or more, an ESU packet is automatically generated). This reflects a typical timeout value that might be used within a DIA system (Singhal et al. 1999). Finally, ideal network conditions were assumed (i.e. no network latency or packet loss).

From inspection of the results, it is noted that in almost every case, the DHSM offers a reduction in the number of packets sent that ranges from small bandwidth savings (in the region of 1% or lower packet reduction) to very large bandwidth savings (in the region of 20% and sometimes higher packet reduction). It is important to note here that a quantitative comparison of inter-dataset results is not productive, as each test conducted is independent of any other. Despite this fact however, comparatively speaking it would appear that the percentage savings are partly dependent on the particular test subject and test environment, and

Table 1: Packet number results for several datasets

| Number of Update Packets Sent (PacketNum) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Threshold | Dataset 1 (Environment 1) | | | Dataset 2 (Environment 2) | | | Dataset 3 (Environment 3) | | |
| | DR | DHSM | % Red | DR | DHSM | % Red | DR | DHSM | % Red |
| 0.5 | 1071 | **1087** | 1.49 | 618 | **624** | 0.97 | 478 | **494** | 3.35 |
| 1 | 724 | **720** | -0.55 | 419 | **412** | -1.67 | 340 | **325** | -4.41 |
| 1.5 | 565 | **549** | -2.83 | 336 | **323** | -3.87 | 268 | **241** | -10.07 |
| 2 | 485 | **467** | -3.71 | 274 | **264** | -3.65 | 237 | **210** | -11.39 |
| 2.5 | 442 | **420** | -4.98 | 256 | **244** | -4.69 | 208 | **181** | -12.98 |
| 3 | 391 | **375** | -4.09 | 228 | **216** | -5.26 | 184 | **154** | -16.30 |
| 3.5 | 364 | **349** | -4.12 | 214 | **203** | -5.14 | 172 | **143** | -16.86 |
| 4 | 327 | **315** | -3.67 | 185 | **178** | -3.78 | 165 | **133** | -19.39 |
| 4.5 | 311 | **298** | -4.18 | 173 | **167** | -3.47 | 160 | **126** | -21.25 |
| 5 | 284 | **272** | -4.23 | 170 | **161** | -5.29 | 149 | **124** | -16.78 |
| 5.5 | 265 | **256** | -3.40 | 156 | **150** | -3.85 | 143 | **110** | -23.08 |
| 6 | 249 | **244** | -2.01 | 150 | **143** | -4.67 | 136 | **107** | -21.32 |
| 6.5 | 239 | **232** | -2.93 | 146 | **139** | -4.79 | 131 | **105** | -19.85 |
| 7 | 233 | **223** | -4.29 | 138 | **130** | -5.80 | 127 | **101** | -20.47 |
| 7.5 | 220 | **214** | -2.73 | 129 | **120** | -6.98 | 120 | **97** | -19.17 |
| 8 | 211 | **211** | 0 | 124 | **115** | -7.26 | 114 | **94** | -17.54 |
| 8.5 | 207 | **200** | -3.38 | 121 | **115** | -4.96 | 109 | **93** | -14.68 |
| 9 | 201 | **197** | -1.99 | 123 | **113** | -8.13 | 108 | **92** | -14.81 |
| 9.5 | 194 | **191** | -1.55 | 115 | **110** | -4.35 | 108 | **92** | -14.81 |
| 10 | 202 | **192** | -4.95 | 111 | **104** | -6.31 | 107 | **90** | -15.89 |

Table 2: Average prediction error results for several datasets

| Average Prediction Error per Simulation Step (E) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Threshold | Dataset 1 (Environment 1) | | | Dataset 2 (Environment 2) | | | Dataset 3 (Environment 3) | | |
| | DR | DHSM | % Red | DR | DHSM | % Red | DR | DHSM | % Red |
| 0.5 | 0.2535 | **0.2568** | 1.30 | 0.2751 | **0.2786** | 1.27 | 0.254 | **0.2657** | 4.61 |
| 1 | 0.4111 | **0.4256** | 3.53 | 0.4509 | **0.4595** | 1.91 | 0.3938 | **0.4151** | 5.41 |
| 1.5 | 0.5755 | **0.5947** | 3.34 | 0.6148 | **0.6288** | 2.28 | 0.54 | **0.5751** | 6.50 |
| 2 | 0.7449 | **0.7586** | 1.84 | 0.7809 | **0.7867** | 0.74 | 0.6856 | **0.6817** | -0.57 |
| 2.5 | 0.9104 | **0.9283** | 1.97 | 0.9842 | **0.9921** | 0.80 | 0.8251 | **0.8895** | 7.81 |
| 3 | 1.0529 | **1.0597** | 0.65 | 1.1595 | **1.1723** | 1.10 | 0.9804 | **1.0412** | 6.20 |
| 3.5 | 1.2112 | **1.2192** | 0.66 | 1.3419 | **1.3792** | 2.78 | 1.1159 | **1.1702** | 4.87 |
| 4 | 1.364 | **1.3671** | 0.23 | 1.4808 | **1.5128** | 2.16 | 1.3394 | **1.3686** | 2.18 |
| 4.5 | 1.5533 | **1.5738** | 1.32 | 1.6675 | **1.6836** | 0.97 | 1.5256 | **1.4827** | -2.81 |
| 5 | 1.7816 | **1.7489** | -1.84 | 1.8471 | **1.7998** | -2.56 | 1.7058 | **1.4778** | -13.37 |
| 5.5 | 1.9759 | **1.9094** | -3.37 | 2.045 | **2.0221** | -1.12 | 1.811 | **1.6177** | -10.67 |
| 6 | 2.119 | **2.1186** | -0.02 | 2.1262 | **2.1933** | 3.16 | 1.9017 | **1.7369** | -8.67 |
| 6.5 | 2.3202 | **2.2692** | -2.20 | 2.3813 | **2.3934** | 0.51 | 2.0685 | **1.9731** | -4.61 |
| 7 | 2.4628 | **2.4195** | -1.76 | 2.6539 | **2.6244** | -1.11 | 2.3996 | **2.0513** | -14.51 |
| 7.5 | 2.74 | **2.5452** | -7.11 | 2.7276 | **2.7212** | -0.23 | 2.468 | **2.2653** | -8.21 |
| 8 | 2.8036 | **2.676** | -4.55 | 2.9641 | **2.979** | 0.50 | 2.6943 | **2.2987** | -14.68 |
| 8.5 | 2.9346 | **2.8045** | -4.43 | 3.07 | **3.0749** | 0.16 | 2.7637 | **2.3358** | -15.48 |
| 9 | 3.1314 | **3.0776** | -1.72 | 3.3125 | **3.2983** | -0.43 | 2.8355 | **2.3022** | -18.81 |
| 9.5 | 3.2789 | **3.1337** | -4.43 | 3.4683 | **3.5024** | 0.98 | 2.9132 | **2.4025** | -17.53 |
| 10 | 3.497 | **3.3846** | -3.21 | 3.6011 | **3.6299** | 0.80 | 3.0654 | **2.4462** | -20.20 |

Table 3: Standard deviation error results for several datasets

| Standard Deviation Error per Simulation Step (D) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Threshold | Dataset 1 (Environment 1) | | | Dataset 2 (Environment 2) | | | Dataset 3 (Environment 3) | | |
| | DR | DHSM | % Red | DR | DHSM | % Red | DR | DHSM | % Red |
| 0.5 | 0.1892 | **0.1887** | -0.26 | 0.1861 | **0.1849** | -0.64 | 0.1829 | **0.1822** | -0.38 |
| 1 | 0.3402 | **0.3392** | -0.29 | 0.3431 | **0.3373** | -1.69 | 0.3289 | **0.3211** | -2.37 |
| 1.5 | 0.4929 | **0.4946** | 0.34 | 0.5004 | **0.4913** | -1.82 | 0.4809 | **0.459** | -4.55 |
| 2 | 0.6534 | **0.6462** | -1.10 | 0.6469 | **0.6352** | -1.81 | 0.6383 | **0.6017** | -5.73 |
| 2.5 | 0.8106 | **0.7975** | -1.62 | 0.8129 | **0.7998** | -1.61 | 0.7939 | **0.7734** | -2.58 |
| 3 | 0.9625 | **0.9387** | -2.47 | 0.9556 | **0.9369** | -1.96 | 0.9371 | **0.8948** | -4.51 |
| 3.5 | 1.1214 | **1.1073** | -1.26 | 1.1245 | **1.1062** | -1.63 | 1.0779 | **1.0151** | -5.83 |
| 4 | 1.2799 | **1.2559** | -1.88 | 1.2485 | **1.2537** | 0.42 | 1.2713 | **1.1864** | -6.68 |
| 4.5 | 1.4389 | **1.4297** | -0.64 | 1.4292 | **1.4322** | 0.21 | 1.4345 | **1.3156** | -8.29 |
| 5 | 1.5846 | **1.5434** | -2.60 | 1.5834 | **1.5586** | -1.57 | 1.5892 | **1.4029** | -11.72 |
| 5.5 | 1.7745 | **1.7205** | -3.04 | 1.7257 | **1.6928** | -1.91 | 1.7644 | **1.5378** | -12.84 |
| 6 | 1.894 | **1.88** | -0.74 | 1.8698 | **1.8588** | -0.59 | 1.9044 | **1.6508** | -13.32 |
| 6.5 | 2.0589 | **1.985** | -3.59 | 2.0575 | **2.0248** | -1.59 | 2.0685 | **1.8111** | -12.44 |
| 7 | 2.1962 | **2.1392** | -2.60 | 2.2371 | **2.2154** | -0.97 | 2.1523 | **1.9146** | -11.04 |
| 7.5 | 2.3595 | **2.2902** | -2.94 | 2.3027 | **2.282** | -0.90 | 2.3568 | **2.0622** | -12.50 |
| 8 | 2.5071 | **2.4264** | -3.22 | 2.4868 | **2.4522** | -1.39 | 2.4912 | **2.1919** | -12.01 |
| 8.5 | 2.6202 | **2.5558** | -2.46 | 2.6211 | **2.5912** | -1.14 | 2.611 | **2.3198** | -11.15 |
| 9 | 2.7764 | **2.7084** | -2.45 | 2.7888 | **2.7221** | -2.39 | 2.7297 | **2.3558** | -13.70 |
| 9.5 | 2.8996 | **2.8537** | -1.58 | 2.9252 | **2.8624** | -2.15 | 2.8617 | **2.4948** | -12.82 |
| 10 | 3.0724 | **3.0582** | -0.46 | 3.0397 | **3.016** | -0.78 | 3.0601 | **2.6697** | -12.76 |

definitely dependent in this case on the particular dynamic prediction model being used (i.e. shortest-path). A more advanced and accurate prediction model would almost certainly yield even better results, particularly if it were tailored for a specific user's behavioural traits and habits. It is also worth noting that the reduction in number of packets sent does not appear to be linearly dependent on the error-threshold (i.e.

increasing the error-threshold does not guarantee a higher reduction in number of packets sent over the corresponding standard dead reckoning prediction model).

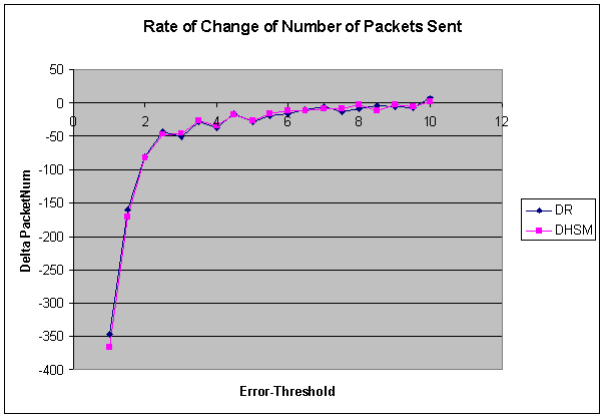


Figure 2: ΔPacketNum vs error-threshold (Dataset 1)

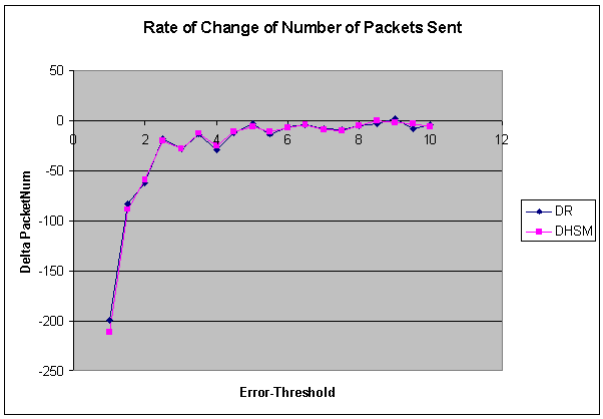


Figure 3: ΔPacketNum vs error-threshold (Dataset 2)

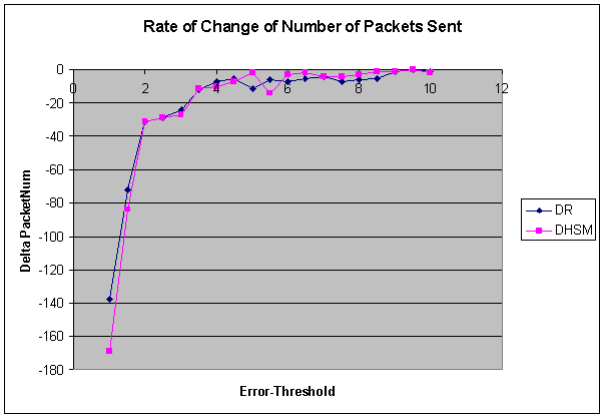


Figure 4: ΔPacketNum vs error-threshold (Dataset 3)

Further inspection of the results highlights the fact that the reduction in packet numbers (and hence bandwidth savings) for the DHSM often comes at the cost of slightly reduced prediction accuracy for tight error-thresholds ($\leq 4$). Conversely, at higher error-thresholds ($\geq 5$) the DHSM often provides improved prediction accuracy. In addition, the DHSM also offers a smaller standard deviation for the prediction error over its DR counterpart in practically every case, implying a slightly more stable predictive capability. This is further evidenced in Figures 2-4 that present the rate of change of generated packets as a function of the increasing error-threshold for each dataset. Inspection of these plots reveals no erroneous packet generation behaviour for the DHSM when compared with the DR mechanism. On the contrary, they provide a close match for all three datasets.

## CONCLUSIONS

In this paper, we have described a novel extension to the concept of the Hybrid Strategy Model (HSM) called the Dynamic Hybrid Strategy Model (DHSM). Like the HSM, the DHSM attempts to reduce the amount of entity-state-update (ESU) packets required to maintain consistency within a Distributed Interactive Application (DIA) such as a networked multiplayer computer game. Unlike the HSM, which uses fixed long-term goals, the DHSM takes account of a user's behaviour towards a dynamic goal and attempts to exploit the shared information contained within this relationship. By switching between various candidate prediction models at appropriate times, the DHSM provides improved remote prediction for clients participating within the DIA.

We have provided simulation results for several different test subjects and test environments that verify the validity of our approach, showing a reduction in the number of ESU packets sent (bandwidth usage) in favour of the DHSM technique over pure dead reckoning in the majority of our specific test situations. In addition, improved average prediction error for higher error-thresholds and improved standard deviation error make this a promising technique for possible use within a prediction scheme incorporating some kind of adaptive error-threshold selection (Lee et al. 1999; Shim and Kim 2001). Despite the promising results however, a better understanding of the relationship between packet reductions, user experience and test environment topology is required, and to this end additional tests will need to be conducted utilizing more complex, real-world scenarios. These include such things as realistic network conditions, reactive computer-controlled opponents and human vs. human experiments.

Future work will involve the investigation of advanced user-modelling techniques to provide a better pool of candidate prediction models that work under more general and complex situations, where the likes of simple shortest-path type prediction will not provide enough prediction accuracy. Possible avenues of approach here include the use of neural networks (Thurau et al. 2003) and probabilistic independence networks (Smyth et al. 1996) to model the relationships between users and dynamic goals. In addition, extensions to the switching criteria and score functions used by the DHSM will also be investigated with the aim of ensuring optimal model switching.

## ACKNOWLEDGEMENT

This material is based upon works supported by Enterprise Ireland under grant no. SC/2002/129/.

## REFERENCES

Delaney, D.; Ward, T.; and S. McLoone. 2003. "Reducing Update Packets in Distributed Interactive Applications using a Hybrid Approach." In *Proceedings of the 16th International Conference on Parallel and Distributed Computing Systems (PDCS 2003)* (Reno, USA, Aug. 13-15). 417-422.

Delaney, D. 2004. *Latency Reduction in Distributed Interactive Applications Using Hybrid Strategy-Based Models*. Ph.D. dissertation Department of Electronic Engineering. National University of Ireland, Maynooth. November 2004.

Faisstnauer, C.; Schmalstieg, D.; and W. Purgathofer. 2000. "Scheduling for Very Large Virtual Environments and Networked Games Using Visibility and Priorities." In *Proceedings of the 4th IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT'00)* (San Francisco, California, Aug. 25-27). 31-38.

IEEE. 1995. *IEEE Standard for Distributed Interactive Simulation – Applications Protocols*. Volume IEEE Standard 1278.1-1995. The Institute for Electrical and Electronics Engineers, Inc., New York, March 26 1996.

Lee, B.S.; Cai, W.; Turner, S.J.; and L. Chen. 1999. "Adaptive Dead Reckoning Algorithms for Distributed Interactive Simulation." *International Journal of Simulation* 1. No. 1-2.

Liang, L. A. H.; Cai, W.; Lee, B. S.; and S.J. Turner. 1999. "Performance Analysis of Packet Bundling Techniques in DIS." In *Proceedings of the 3rd IEEE International Workshop on Distributed Interactive Simulation and Real-Time Applications* (College Park, Maryland, Oct. 23-24). 75-82.

Marshall, D.; McCoy, A.; Delaney, D.; McLoone, S.; and T. Ward. 2004. "A Realistic Distributed Interactive Application Testbed for Static and Dynamic Entity State Data Acquisition." In *Proceedings of the Irish Signals and Systems Conference 2004* (Belfast, N.I., Jun. 30 – Jul. 02). 83-88.

McCoy, A.; Delaney, D.; and T. Ward. 2003. "Game-State Fidelity across Distributed Interactive Games." *ACM Crossroads*, 9.4, 4-9.

McCoy, A.; Delaney, D.; McLoone, S.; and T. Ward. 2004a. "Investigating Behavioural State Data-Partitioning for User-Modelling in Distributed Interactive Applications." In *Proceedings of the 8th IEEE International Symposium on Distributed Simulation and Real Time Applications* (Budapest, Hungary, Oct. 21-23). 74-82.

McCoy, A.; Delaney, D.; McLoone, S.; and T. Ward. 2004b. "Towards Statistical Client Prediction – Analysis of User Behaviour in Distributed Interactive Media." In *Proceedings of the 5th Game-On International Conference (CGAIDE 2004)* (MS Campus, Reading, UK, Nov, 8-10). 144-149.

Rak, S. J. and D. J. van Hook. 1996. "Evaluation of Grid-based Relevance Filtering for Multicast Group Assignment." In *Proceedings of the 14th DIS Workshop on Standards for the Interoperability of Distributed Simulation* (March 1996). 739 – 747.

Shim, K. H. and J. S. Kim. 2001. "A Dead Reckoning Algorithm with Variable Threshold Scheme in Networked Virtual Environment." In *Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics* Vol. 2 (Oct. 7-10). 1113-1118.

Singhal, S. and D. Cheriton. 1994. "Using a Position History-Based Protocol for Distributed Object Visualisation." Technical Report STAN-CS-TR-94-1505. Department of Computer Science, Stanford University.

Singhal, S. and M. Zyda. 1999. *Networked Virtual Environments: Design and Implementation*. ACM Press SIGGRAPH Series. Addison-Wesley, Reading, Massachusetts.

Smyth, P.; Heckerman, D.; and M.I. Jordan. 1996. "Probabilistic Independence Networks for Hidden Markov Probability Models." *Neural Computation* Vol.9, No.2. 227-269.

Thurau, C.; Bauckhage, C.; and G. Sagerer. 2003. "Combining Self Organizing Maps and Multilayer Perceptrons to Learn Bot-Behaviour for a Commercial Game." In *Proceedings of the 4th International Conference on Intelligent Games and Simulation (Game-On 2003)* (London, UK, Nov. 19-21). 119-124.

Van Hook, D. J.; Calvin, J. O.; and D. C. Miller. 1994. "A protocol independent compression algorithm (PICA)." *Advanced Distributed Simulation Memorandum* 20PM-ADS-005. MIT Lincoln Laboratories, Lexington, MA.

Zhang, X.; Gracanin, D.; and T. P. Duncan. 2004. "Evaluation of a Pre-Reckoning Algorithm for Distributed Virtual Environments." In *Proceedings of the 10th International Conference on Parallel and Distributed Systems (ICPADS'04)* (Newport Beach, California, Jul. 7-9).

## AUTHOR BIOGRAPHIES

**AARON MCCOY** received his B.Sc. Degree from the National University of Ireland, Maynooth in 2002, specializing in computer science and theoretical physics. His main areas of interest are distributed systems, networked virtual environments and the use of artificial intelligence in interactive games. He is currently studying for his Ph.D. at the Department of Electronic Engineering, NUI Maynooth, in the area of user modelling in Distributed Interactive Applications.

**DECLAN DELANEY** received his electronic engineering degree from University College Dublin in 1991 and his Masters degree in 1999. At present, he lectures in the Department of Computer Science at NUI Maynooth and is director of the Masters of Science in Software Engineering program. His current research interests focus on techniques for reducing the effects of latency in distributed networked applications. He is a chartered member of the IEI.

**SEAMUS MCLOONE** received his B.Eng. honours degree in Electrical and Electronic Engineering from Queen's University Belfast in 1996 and his Ph.D. in September 2000. At present, he is employed as a lecturer in the Department of Electronic Engineering at NUI Maynooth. His main research interests include multiple model approaches to nonlinear system identification and reducing/masking the effects of latency in distributed interactive applications.

**TOMÁS WARD** received the B.E., the M.Eng.Sc. and Ph.D. degrees from University College Dublin, Ireland in 1994, 1996 and 1999 respectively. In 2000, he became a lecturer in the Department of Electronic Engineering at NUI Maynooth. His main research interests lie in the areas of near-infrared optical and terahertz medical imaging, and distributed environments for multi-user interaction.