A Hybrid Interest Management Mechanism for Peer-to-Peer Networked Virtual Environments

Ke Pan, Wentong Cai, Xueyan Tang, Suiping Zhou and Stephen John Turner
Parallel and Distributed Computing Center
School of Computer Engineering
Nanyang Technological University
Singapore 639798
{pank0001, aswtcai, asxytang, asspzhou, assjturner}@ntu.edu.sg

Abstract

An Interest Management (IM) mechanism eliminates irrelevant status updates transmitted in Networked Virtual Environments (NVE). However, IM itself involves both computation and communication overhead, of which the latter is the focus of this paper. Traditionally, there are area-based and cell-based IM mechanisms. This paper proposes a hybrid IM mechanism for peer-to-peer NVEs, that utilizes the cell-based mechanism to reduce Area-Of-Interest (AOI) updates in the area-based mechanism so as to reduce its communication overhead. To compare the new mechanism with the two traditional approaches, a multiplayer game scenario is simulated. The performance results show that, compared to the traditional mechanisms, the hybrid mechanism reduces the upload bandwidth consumption by more than 25.28 percent, reduces the overhead ratio from more than 67.54 percent to only 25.17 percent, and allows more than 5000 players in the Internet to join the same game with today's network upload bandwidth.

1. Introduction

A Networked Virtual Environment (NVE) comprises a set of interconnected nodes that cooperatively simulate the behavior of various entities in a shared virtual environment [39], [46]. It originated with military simulators in 80's [30] and had since evolved to various networked game genres, such as First Person Shooter (FPS) games like Counter Strike, Real-Time-Strategy (RTS) games like Warcraft, and today's popular Massively Multiplayer Online Games (MMOG) like World of Warcraft. The special entertaining value of interacting with thousands of people in the same NVE makes the revenue of the computer game business twice that of movies [48], [7].

Traditional NVEs follow a client-server architecture [45], [11], [49], [1] in which one or multiple servers are responsible for both maintaining and disseminating entity status updates to other entities as well as account management and player authentication. Though the client-server architecture is good at cheating prevention [8], [5], it creates both computation and communication bottlenecks at the server site. This problem can be alleviated by using server clusters or Grid resources [38], [24], [37]. However, another drawback of the client-server architecture is the multi-hop latency and it is inevitable. Each status update of an entity needs to be calculated by the server and then relayed to all entities (including the sending entity for cheating prevention). As latency causes inconsistency between entities' views of other entities, NVE interaction and user satisfaction will be significantly degraded if latency is long [6], [26]. For FPS games, latency greater than 200ms is generally unacceptable [6] and Donnybrook uses 150ms as its latency deadline [7]. From the economic aspect, the client-server architecture needs high server maintenance cost and the cost is proportional to the NVE complexity. This may prevent small game publishers from entering the market, which is not healthy for the development of the game industry.

Recently, many Peer-to-Peer (P2P) NVEs emerge [25], [10], [21], [29], [7]. In a P2P architecture, each peer is responsible for maintaining a set of entities and disseminating their status updates to other entities. The computation and communication overhead is shared among peers, so the architecture is more scalable. Communications are not through a server, so latency is improved. From the economic aspect, since there is no server maintenance cost, play cost is reduced and this allows small game publishers to enter the market without too much investment risk in games of unknown popularity [7]. One drawback of the P2P architecture is the challenge to prevent cheating [5]. As entities are maintained by peers, it is difficult to detect any game-play logic violation of a peer [10]. This problem can be mitigated by several recently proposed techniques as discussed in [8].

We believe the P2P architecture is the trend for NVEs in the future and this paper focuses on optimizing communication efficiency (bandwidth consumption) in P2P NVEs. To keep a consistent view between entities, P2P NVEs require frequent entity status updates between peers. Due to the lack of IP multicast support between most peers in the Internet, status updates are generally sent to other peers using unicast. This leads to a tremendous upload bandwidth requirement by each peer and limits the total number of entities in an NVE. Two common approaches to reduce status updates between peers are dead reckoning [40], [35], [34] and Interest Management (IM) [4], [32].

IM is the focus of this paper and it is realized by defining an Area-Of-Interest (AOI) for each entity that describes a region in the Virtual Environment (VE) which an entity is interested in. An entity is normally represented as a single point in the VE and it only needs to receive status updates of other entities which are inside its AOI. In this way, the number of status updates can be significantly reduced. There are two traditional IM mechanisms, namely an area-based mechanism [16], [47], [10] and a cell-based mechanism [28], [36], [22], [25]. In an area-based mechanism, each entity keeps updating its AOI with all other entities. Upon receiving an AOI update of another entity, the receiving entity checks whether it falls in the AOI and this process is referred as matching. With this exact matching results, an entity is able to send its status updates to the other entities which are interested in it. However, sending AOI updates is a costly overhead as an entity's AOI changes dynamically with its position and an entity sends its AOI updates to all other entities irrespective of their relative positions to itself. This paper proposes a hybrid mechanism that utilizes the other traditional mechanism, i.e. a cell-based mechanism, to reduce the AOI updates in the area-based mechanism. To compare the hybrid mechanism with the two traditional mechanisms, a multiplayer game scenario is simulated. The performance results show that, compared to the traditional mechanisms, the hybrid mechanism reduces the upload bandwidth consumption by more than 25.28 percent, reduces the overhead ratio from more than 67.54 percent to only 25.17 percent, and allows more than 5000 players in the Internet to join the same game with today's network upload bandwidth.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries on the traditional IM mechanisms. Section 3 presents the proposed hybrid IM mechanism. Section 4 discusses the multiplayer game scenario and its simulation results. Section 5 summarizes the related work. Finally, Section 6 concludes the paper and discusses future work.

2. Preliminaries

We consider a set of inter-connected peers that host entities moving around in a shared VE. Each peer periodically refreshes the status of its hosted entities based on their kinetics and user actions. The period between two successive refreshes is called a frame. Each entity has an associated AOI that moves along with the entity and the AOI is a region of any shape in the VE. This paper focuses on the communication efficiency (bandwidth consumption) of P2P NVEs. As the total upload communication cost in an NVE equals the total download communication cost, the average upload communication cost per peer is equal to the average download communication cost per peer. Since today's broadband connections are generally asymmetric with the key limitation being upload capacity [7], this paper uses upload communication cost for discussions. The idea of this paper applies to VEs with any number of dimensions. However, for the simplicity of description, the sequel of this paper uses two-dimension VEs.

2.1. Area-Based IM

The area-based IM mechanism is also known as the region-based IM in the distributed simulation area [23], [16], [47], [15]. In this mechanism, each entity keeps updating its AOI with other entities for matching. We assume the AOI of an entity can always be described and encoded in an AOI update. For example, a rectangular AOI can be encoded with the coordinates of its top-left and right-bottom corners and a circular AOI can be encoded with its center coordinates and radius.

It is not affordable for an entity to send its AOI updates in each frame as this causes too much communication overhead as well as computation cost for matching [33]. A simple strategy is to send AOI updates periodically [33], [4], [27], [10]. Another strategy is to perform AOI updates in a space-driven manner [4], where an entity sends a new AOI update only when its position has changed over a distance threshold since its last AOI update. However, the matching results maintained by these strategies may be inaccurate since the entities are not aware of the changes in AOI positions between successive updates. For physically correct filtering [1], both strategies require an AOI to be artificially expanded to always include its true AOIs before the next AOI update [33], [10]. This expansion causes irrelevant status updates between entities in that an entity may receive status updates of other entities that are outside its current AOI. The tradeoff between the AOI update overhead and irrelevant status updates requires the updating period or distance threshold to be carefully chosen in order to optimize the communication efficiency of the area-based mechanism.

2.2. Cell-Based IM

The cell-based IM mechanism is also known as the grid-based IM in the distributed simulation area [36], [23], [42], [22]. This mechanism divides the VE into disjoint cells generally in squares [36], [23], [2] or hexagons [28], [18]. With underlying multicast network communication support, each cell is allocated a multicast address [36], [2]. An entity maps its AOI to the cells and joins the multicast groups of the cells overlapping with its AOI. Each entity sends its status updates to the multicast group of its residing cell. In this way, the cell-based IM mechanism is automatically realized.

However, IP multicast support is not available between most peers in the Internet. For a P2P NVE deployed in the internet, its IM mechanism needs to manage the cells by itself. An entity maps its AOI to the cells and sends out updates about the set of cells overlapping with its AOI. An entity also keeps a list for each cell, that remembers all other entities whose AOIs overlap with the cell. If an entity currently resides in a cell denoted as C, it needs to send its status update to all other entities in the list of Cell C. With the cell-based mechanism, an entity sends updates about its AOI only when its AOI moves across cell borders. However, the cell-based mechanism causes irrelevant status updates as its matching is not exact. As long as Entity B's AOI overlaps with Entity A's residing cell, A sends its status update to B even if A's position is not inside B's AOI. Both the IM overhead and the irrelevant status updates are dependant on the cell size. In order to optimize the communication efficiency of the cell-based mechanism, the cell size needs to be carefully chosen to balance the tradeoff between them. Ayani et. al. developed an analytical model and derived a formula for choosing the optimal cell size for cell-based mechanisms [3]. Their optimization considered both computation cost for matching and communication cost.

1. Physically correct filtering should ensure that a status update is always sent to an entity if the entity is interested in it.

When an entity's AOI moves across cell borders, the entity sends updates about its AOI to all other entities. This update can be encoded in two ways by the updating entity. The first way is to map the current AOI position to the cells and include in a new update the set of stale overlapping cells and the set of new overlapping cells of the AOI since the last update. The second way is to directly include the current AOI position in the update, e.g. the coordinates of the top-left and right-bottom corners for a rectangular AOI, and it requires all other entities to separately derive the sets of stale overlapping cells and new overlapping cells after receiving the AOI update. The first way saves the computation cost for matching, but it generally incurs more communication overhead. Since this paper focuses on optimizing communication cost, the second way is used in the later discussions and experimental evaluations of the cell-based mechanism.

3. Hybrid IM

As discussed in Subsection 2.1, both periodical and space-driven area-based mechanisms send AOI updates among all entities irrespective of their relative positions. Suppose an entity is located far from another entity's AOI, it is in fact unnecessary for the second entity to send its AOI update to the first entity. This paper proposes a hybrid mechanism that utilizes the cell-based mechanism to reduce the AOI updates in the area-based mechanism. Specifically, each entity updates its residing cell with all other entities when it moves across cell borders. As a result, each entity is able to maintain the residing cells of all other entities. With this information, an entity sends its AOI updates only to the entities near its AOI. In this way, the number of AOI updates is expected to be greatly reduced. Similar to original area-based mechanisms discussed in Subsection 2.1, AOI updates can be sent in a periodical manner or a space-driven manner. For either manner, the AOI is artificially expanded for physically correct filtering and we denote the expanded AOI as ExAOI.

3.1. Identification of AOI Update Receivers

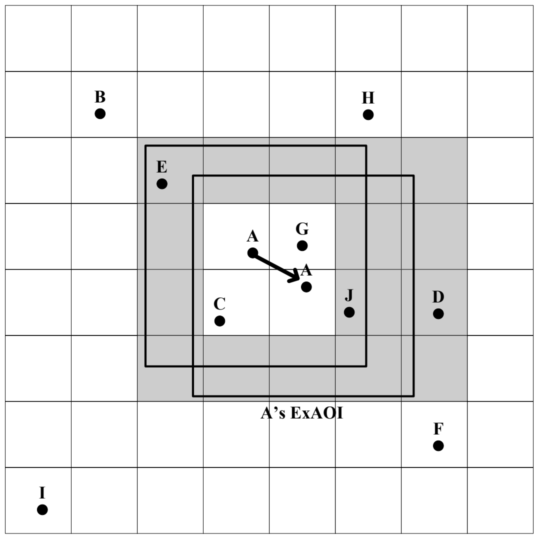

Figure 1. AOI Update Receivers

As shown in Figure 1, when the time since Entity A's last AOI update has exceeded its AOI updating period in a periodical mechanism or when the shifting distance from A's position at the last AOI update has exceeded the distance threshold in a space-driven mechanism, its ExAOI is recalculated and mapped to the cells. We denote the ExAOI in the last AOI update as LastExAOI and its overlapping cell set as LastExAOICellSet. Similarly, we denote the new ExAOI calculated as NewExAOI and its overlapping cell set as NewExAOICellSet. Entities B, F, H and I in Figure 1 lie in cells outside both LastExAOICellSet and NewExAOICellSet, so their matching status with A's ExAOI does not change and remains lying outside ExAOI. We name this condition as *Outside Irrelevance*. We call the cells that an ExAOI's periphery overlaps as *peripheral cells*. Entities C and G in Figure 1 lie in cells inside both LastExAOICellSet and NewExAOICellSet and their residing cells are not peripheral cells in either LastExAOICellSet or NewExAOICellSet, so their matching status with A's ExAOI does not change either and remains lying inside ExAOI. We name this condition as *Inside Irrelevance*. With the knowledge of the other entities' residing cells, Entity A does not need to send its AOI update to entities B, F, H, I and C, G.

In general, an entity sends its AOI update to another entity if and only if either of the following two conditions is true.

- **Condition 1**: The receiving entity's residing cell is inside either LastExAOICellSet or NewExAOICellSet of the sending entity but not both.
- **Condition 2**: The receiving entity's residing cell is a peripheral cell in either LastExAOICellSet or NewExAOICellSet of the sending entity.

This principle applies to any AOI shape. In the scenario of Figure 1, Entity A only sends its AOI updates to the entities residing in the shaded cells, i.e. entities D, E and J.

3.2. Per Entity Algorithm

The hybrid IM algorithm description in this subsection follows a per entity basis such that each entity executes the same algorithm in parallel. This is consistent with the feature of many P2P NVE applications as in most of today's multiplayer online games, each player controls only one entity, i.e. its avatar, in the VE. In other words, each peer is responsible for one entity. For P2P NVEs where a peer is responsible for a set of entities, some simple modifications are needed to make it follow a per peer basis such that each peer executes the same algorithm in parallel and manages the functionalities of all local entities. Due to the space limit, the modifications are not presented in this paper. The following notations are used for algorithm description.

- **NewResidingCell**: Local entity's current residing cell.
- **LastResidingCell**: Local entity's last residing cell which was sent to all remote entities.
- **AOIUpdatePeriod**: AOI update period in frames for a periodical mechanism.
- **DistanceThreshold**: Distance threshold for a space-driven mechanism.
- **NewExAOICellSet**: The set of cells overlapping with the new ExAOI of the local entity.
- **LastExAOICellSet**: The set of cells overlapping with the ExAOI of the local entity at the time of its last AOI update.
- **ResidingRemoteEntityList[NumberOfCells]**: An array of lists with each list remembering remote entities residing in a particular cell.
- **RemoteExAOI[NumberOfEntities]**: Remote entities' expanded AOI information based on their respective AOI updates.
- **PartiallyOverlappingRemoteExAOIList**: A list remembering the expanded AOIs of remote entities which partially overlap with local entity's current residing cell (i.e., local entity's current residing cell is a peripheral cell of the expanded AOIs in the list).

- **MatchingRemoteEntityList**: A list of remote entities whose ExAOIs enclose the local entity.

The algorithm for each entity to execute at each frame is shown in Algorithm 1. It shows only the space-driven mechanism. For the periodical mechanism, the procedure is similar with Step 10 changed to detect whether the time duration from the time the last AOI update of the local entity is sent out exceeds AOIUpdatePeriod. In addition to the main algorithm thread, each peer keeps another thread, namely Thread-Receiving. It is responsible for receiving messages (including both status update messages and IM control messages) from remote entities and its pseudo code is shown in Algorithm 2.

Algorithm 1 Per Entity Per Frame Algorithm

```
 1: make movement for the local entity;
 2: update the AOI of the local entity;
 3: calculate NewResidingCell;
 4: if $NewResidingCell \neq LastResidingCell$ then
 5:     send NewResidingCell to all remote entities;
 6:     LastResidingCell=NewResidingCell;
 7: else
 8:     check the current position of the local entity against the expanded AOIs of remote entities in PartiallyOverlappingRemoteExAOIList and update MatchingRemoteEntityList accordingly;
 9: end if
10: if position shifting from the last AOI update exceeds DistanceThreshold in either dimension then
11:     update the expanded AOI of the local entity;
12:     calculate NewExAOICellSet;
13:     for each remote entity E satisfying either Condition 1 or Condition 2 do
14:         send an AOI update to E;
15:     end for
16:     LastExAOICellSet=NewExAOICellSet;
17: end if
18: for each entity E in MatchingRemoteEntityList do
19:     send a status update to E;
20: end for
```

If a movement of the local entity in a frame changes its residing cell (Step 4 in Algo. 1), all remote entities need to be notified of this (Step 5 in Algo. 1). Assuming the LastResidingCell is remembered by all remote entities when they receive the previous notification from the local entity, the only information that needs to be contained in the notification is NewResidingCell. After receiving the notification (Step 1 in Algo. 2), the receiving entity updates the

Algorithm 2 Thread-Receiving

```
 1: if $type = NewResidingCellNotification$ then
 2:     append the sending entity's identifier to ResidingRemoteEntityList[the sending entity's NewResidingCell];
 3:     remove the sending entity's identifier from ResidingRemoteEntityList[the sending entity's LastResidingCell];
 4:     reply to the sending entity with local entity's last sent AOI update;
 5: end if
 6: if $type = AOIUpdate$ then
 7:     update RemoteExAOI[the sending entity] with the new expanded AOI information;
 8:     if local entity's residing cell is a peripheral cell of RemoteExAOI[the sending entity] and RemoteExAOI[the sending entity] is not in PartiallyOverlappingRemoteExAOIList then
 9:         add RemoteExAOI[the sending entity] to PartiallyOverlappingRemoteExAOIList;
10:     end if
11:     if local entity's residing cell is not a peripheral cell of RemoteExAOI[the sending entity] and RemoteExAOI[the sending entity] is in PartiallyOverlappingRemoteExAOIList then
12:         delete RemoteExAOI[the sending entity] from PartiallyOverlappingRemoteExAOIList;
13:     end if
14:     if the local entity resides in the received ExAOI and the sending entity is not in MatchingRemoteEntityList then
15:         append the sending entity to MatchingRemoteEntityList;
16:     end if
17:     if the local entity does not reside in the received ExAOI then
18:         delete the sending entity from MatchingRemoteEntityList if necessary;
19:     end if
20: end if
21: if $type = StatusUpdate$ then
22:     process the status update;
23: end if
```

corresponding ResidingRemoteEntityList (Steps 2 and 3 in Algo. 2) and replies to the sending entity with its last sent AOI update (Step 4 in Algo. 2). The sending entity uses the reply to update its RemoteExAOI, PartiallyOverlappingRemoteExAOIList and matching status stored in MatchingRemoteEntityList (Steps 6-20 in Algo. 2). So, the communication cost (upload cost) of an entity due to the changes of all entities' residing cells (denoted as $Cost_{ChangingResidingCell}$) consists of two parts, i.e., the cost for notifying remote entities of the changes of local entity's residing cell and the cost for replying to the changes of remote entities' residing cells.

When local entity's residing cell does not change (Step 7 in Algo. 1), its movement may cause the change of its matching status with a remote entity only if its residing cell is a peripheral cell of the remote entity's expanded AOI (Step 8 in Algo. 1). So, MatchingRemoteEntityList needs to be updated accordingly (Step 8 in Algo. 1). When an entity is going to send a new AOI update out (Step 10 in Algo. 1), the new AOI update is only sent to those entities which satisfy either Condition 1 or Condition 2 as discussed in Subsection 3.1 (Steps 13 and 14 in Algo. 1). We denote the communication cost (upload cost) of an entity for sending AOI updates as $Cost_{AOIUpdate}$. An entity updates its RemoteExAOI, PartiallyOverlappingRemoteExAOIList and matching status stored in MatchingRemoteEntityList upon receiving an AOI update from another entity (Steps 6-20 in Algo. 2).

Next, the local entity sends its status update to those entities in its MatchingRemoteEntityList (Steps 18 and 19 in Algo. 1). We denote the communication cost (upload cost) of an entity for sending status updates as $Cost_{StatusUpdate}$.

4. Experimental Evaluation

4.1. Experimental Setup

To evaluate the performance of the proposed hybrid IM mechanism, a P2P multiplayer game scenario is simulated with a two-dimensional VE of 1000 distance units by 1000 distance units. Each player controls only one entity, i.e. its avatar, and the avatar is associated with a square AOI of which the range per dimension is randomly generated with a uniform Probability Density Function (PDF) between 50 distance units and 150 distance units. The AOI, centered at the position of the player's avatar, moves along with the entity. The moving speed of each avatar was set at 0.2 distance unit per frame. The random direction mobility model [19] was used to simulate avatar movement. In this model, a player chooses a random direction to travel in and a random duration for the travel. A uniform PDF between 0 radian and $2 * \Pi$ radian is used to generate the random walking direction and a uniform PDF between 0 frames and 100 frames is used to generate the random walking duration. Upon arrival, the player chooses a new walking direction and duration, and repeats the process.

In addition to the proposed hybrid mechanism, we also implemented the area-based mechanism and the cell-based mechanism for performance comparison. For the area-based mechanism, we implemented both a periodical strategy and a space-driven strategy.

All players were initially placed at random in the VE. Each experiment run started with the algorithm variables preset according to avatars' positions and AOIs, and simulated the game execution for a period of 10000 frames. The communication cost (upload bandwidth consumption) of each mechanism is measured and compared in bytes. We assume the network packet header size is 28 bytes. Each data value transmitted, such as a coordinate or a cell number, accounts for 2 bytes. Note that we can encode a float data value with 2 bytes using the form $A * 10^B$ and use 12 bits for the unsigned value A and 4 bits for the signed value B. This generates a range sufficiently large for representing any position in the experimental VE. For VEs with larger space, more bytes can be used to transmit a float data value. The payload size of a status update is different for different games and it is expected to increase as games become more complex and include additional features, such as voice communication, in the future. For Counter Strike, the mean server packet size is 127 bytes [17]. For our experiments, we use 100 bytes for the payload size of a status update. In other words, a status update packet including its header has 128 bytes.

4.2. Results with 1000 Players

First, we analyze the performance of different mechanisms by fixing the total number of players at 1000.

The performance of the periodical area-based mechanism depends on the AOI update period chosen. We have varied the period (in frames) and the average upload communication cost per player per frame for different periods is shown in Figure 2. With an increasing period, the cost of AOI updates is reduced while the cost of irrelevant status updates increases due to the larger AOI expansion. The cost of relevant status updates stays the same. At a period of 60 frames, the communication cost is optimized. We

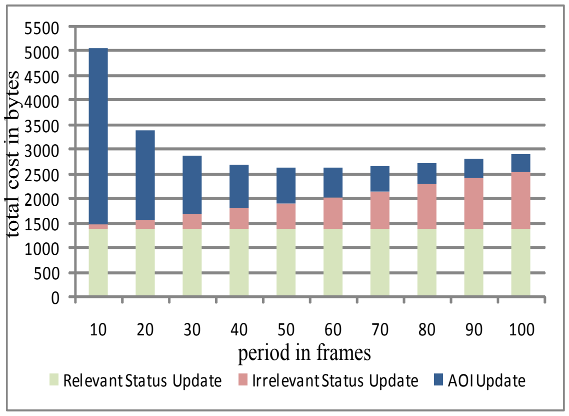

Figure 2. Per Player Per Frame Upload Communication Cost of the Periodical Area-based Mechanism

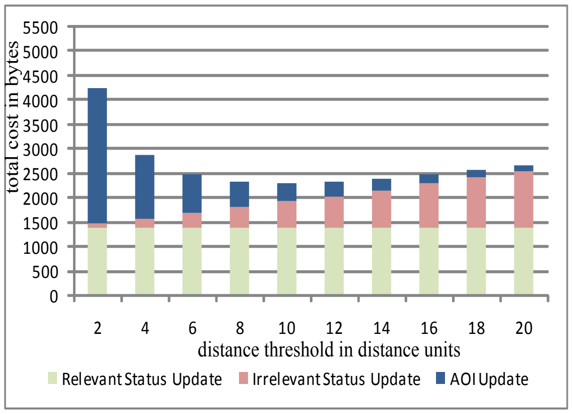

Figure 3. Per Player Per Frame Upload Communication Cost of the Space-Driven Area-based Mechanism

define overhead ratio (including both the IM overhead and irrelevant status updates) as $(TotalCost - RelevantStatusUpdate)/RelevantStatusUpdate$. The overhead ratio of the periodical area-based mechanism at its optimization point is 90.7 percent.

The performance of the space-driven area-based mechanism depends on the distance threshold chosen. We have varied the distance threshold and the performance results for different distance thresholds are shown in Figure 3. With an increasing distance threshold, the cost of AOI updates is reduced while the cost of irrelevant status updates increases due to the larger AOI expansion. The cost of relevant status updates stays the same. At a distance threshold of 10 distance units, the communication cost is optimized with overhead ratio equal to 67.54 percent.

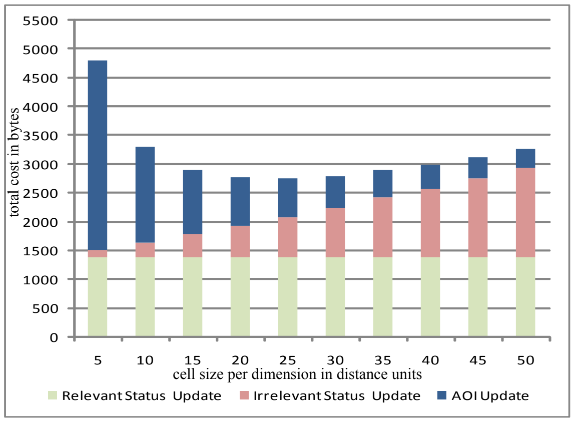

Figure 4. Per Player Per Frame Upload Communication Cost of the Cell-Based Mechanism

DistanceThreshold in distance units		cell size in distance units per dimension				
		90	100	110	120	130
1	**Cost$_{\text{ChangingResidingCell}}$**	161.29	148.06	124.62	114.25	95.5
	Cost$_{\text{AOIUpdate}}$	197.88	217.71	238.62	264.82	300.69
	Irrelevant Status Update	50.14	50.14	50.14	50.14	50.14
	Relevant Status Update	1386.9	1386.9	1386.9	1386.9	1386.9
	Total Cost	1796.2	1802.8	1800.3	1816.1	1833.2
1.5	**Cost$_{\text{ChangingResidingCell}}$**	161.29	148.06	124.62	114.25	95.5
	Cost$_{\text{AOIUpdate}}$	138.43	152.16	166.73	184.81	209.67
	Irrelevant Status Update	75.58	75.58	75.58	75.58	75.58
	Relevant Status Update	1386.9	1386.9	1386.9	1386.9	1386.9
	Total Cost	1762.2	1762.7	1753.8	1761.5	1767.6
2	**Cost$_{\text{ChangingResidingCell}}$**	161.29	148.06	**124.62**	114.25	95.5
	Cost$_{\text{AOIUpdate}}$	102.49	112.55	**123.27**	136.5	154.74
	Irrelevant Status Update	101.25	101.25	**101.25**	101.25	101.25
	Relevant Status Update	1386.9	1386.9	**1386.9**	1386.9	1386.9
	Total Cost	1751.9	1748.7	**1736**	1738.9	1738.4
2.5	**Cost$_{\text{ChangingResidingCell}}$**	161.29	148.06	124.62	114.25	95.5
	Cost$_{\text{AOIUpdate}}$	82.69	90.73	99.35	109.9	124.5
	Irrelevant Status Update	127.12	127.12	127.12	127.12	127.12
	Relevant Status Update	1386.9	1386.9	1386.9	1386.9	1386.9
	Total Cost	1758	1752.8	1738	1738.2	1734
3	**Cost$_{\text{ChangingResidingCell}}$**	161.29	148.06	124.62	114.25	95.5
	Cost$_{\text{AOIUpdate}}$	67.87	74.41	81.48	90.02	101.86
	Irrelevant Status Update	153.19	153.19	153.19	153.19	153.19
	Relevant Status Update	1386.9	1386.9	1386.9	1386.9	1386.9
	Total Cost	1769.2	1762.6	1746.2	1744.4	1737.4

Figure 5. Per Player Per Frame Upload Communication Cost in bytes of the Hybrid Mechanism

For the area-based mechanism, the optimized performance of the space-driven strategy is better than that of the periodical strategy in the experimental scenario. For later experiments, only the space-driven strategy is discussed for the area-based mechanism. The proposed hybrid mechanism can also use either the periodical or the space-driven strategy for its AOI updates. Similarly, only the space-driven strategy is used for analyzing the hybrid mechanism.

The performance of the cell-based mechanism depends on the cell size chosen. We have varied the size (in distance units per dimension) of square cells and the performance results for different cell sizes are shown in Figure 4. With an increasing cell size, the cost of AOI updates decreases as the probability for an AOI to move across cell borders becomes lower. However, an increasing cell size increases the cost of irrelevant status updates as the matching becomes less accurate. The cost of relevant status updates stays the same and equals that of the area-based mechanism. At a cell size of 25, the communication cost is optimized and it is worse than the optimized performance of the two area-based mechanisms. The overhead ratio at its optimization point reaches 98.93 percent.

The performance of the hybrid mechanism depends on both the cell size and distance threshold chosen. We have varied both parameters and the performance results are shown in Figure 5 [2]. The cost notations follow those introduced in Subsection 3.2. With an increasing cell size, $Cost_{ChangingResidingCell}$ is reduced while $Cost_{AOIUpdate}$ is increased due to the less accurate filtering of AOI updates. Both the costs of irrelevant and relevant status updates are not affected by the cell size. With an increasing distance threshold, $Cost_{ChangingResidingCell}$ stays the same while $Cost_{AOIUpdate}$ is reduced. With a larger distance threshold, the AOI is expanded more and this results in higher cost of irrelevant status updates. The cost of relevant status updates is not affected by the distance threshold and is the same as that of area-based and cell-based mechanisms. The communication cost of the hybrid mechanism is optimized with a cell size of 110 distance units and a distance threshold of 2 distance units. Compared with the optimization points of the two traditional mechanisms, the hybrid mechanism reduces upload communication cost by more than 25.28 percent. The overhead ratio of the hybrid mechanism is only 25.17 percent. This is much lower than the overhead ratios of the two traditional mechanisms.

4.3. Scalability Test Results with respect to Number of Players

The second experiment tests the scalability of all mechanisms with respect to the number of players. As the player number does not change the optimization point of each mechanism, we used the optimization point for each algorithm determined in Subsection 4.2. The performance of the area-based mechanism (space-driven), the cell-based mechanism and the hybrid mechanism (space-driven) is shown in Figures 6, 7 and 8 respectively. For each mechanism, all cost components increase linearly with an increasing number of players.

Typical frame rates are 1-5 frames per second for MMOGs and 10-20 frames/second for FPS games [21]. Consider the experimental scenario as a FPS game with its frame rate equal to 10 frames per second. With today's broadband connection, a player normally has an upload capacity not exceeding 1Mb/s [41], [7]. Consider the case that each player has an upload capacity of 800Kb/s, the upload bandwidth consumption of a player per frame should be smaller than or equal to 10000 bytes/frame. With this limit,

2. We have varied both parameters over wide ranges. Due to the space limit, Figure 5 only shows the portion of ranges near the optimization point.

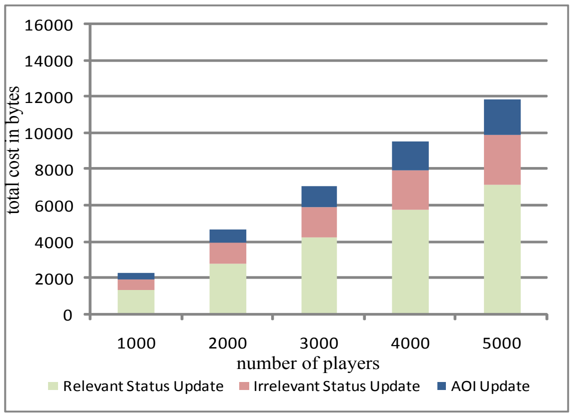

Figure 6. Scalability of the Space-Driven Area-based Mechanism

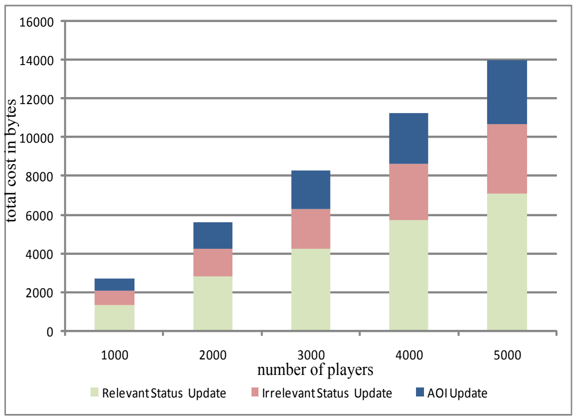

Figure 7. Scalability of the Cell-based Mechanism

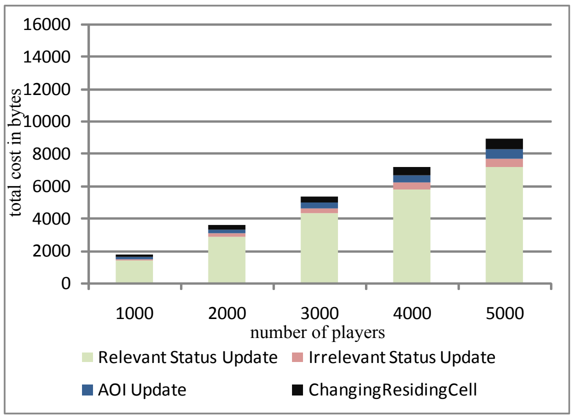

Figure 8. Scalability of the Hybrid Mechanism

the space-driven area-based mechanism can support slightly more than 4000 players in the same game as shown in Figure 6 and the cell-based mechanism can support less than 4000 players as shown in Figure 7. In contrast, the proposed hybrid mechanism is able to support more than 5000 players as shown in Figure 8.

4.4. Discussion

So far, this paper has not considered latencies between players. For physically correct filtering, AOI should be further expanded to account for latencies [33]. Since AOI updates are sent directly between peers in this paper, the latency between two peers is normally half of the Round Trip Time (RTT) between them. According to the statistics in [7], the mean inter-node latency between players in the western U.S. with longitude between 110 and 125 degrees west is 40.5ms and almost all the latencies are within 200 ms. King is a tool that accurately estimates latency between arbitrary Internet end hosts [20]. By investigating the King dataset [31], almost all the RTTs are smaller than 500 ms and thus player-to-player latencies are normally smaller than 250 ms. With the frame rate of 10 frames per second, latencies are generally shorter than 2.5 frames. An AOI expansion due to the latency should be much smaller than the AOI expansion due to the AOI update period (periodical) or the distance threshold (space-driven) in both the area-based and hybrid mechanisms at their respective optimization points as shown in Figures 2, 3 and 5. So, latencies should not affect the above experimental results much.

As varying the other experimental parameters, such as AOI size, players' moving speed and the payload size of a status update, may change the optimization point of a mechanism, the corresponding experiments may need to be carried out. In the future, we are going to carry out a theoretical analysis for each mechanism to derive their respective optimization points based on these parameters. Then, we can carry out further experiments to compare the optimized performance of different mechanisms in different scenarios.

5. Related Work

To reduce the communication overhead of the area-based IM mechanism, Tang and Cai proposed a PIR-based mechanism where PIR is a Potential Interest Region calculated for an AOI satisfying the two constraints: the PIR completely encloses the AOI (coverage constraint); all other unsubscribed entities are located outside the PIR (avoidance constraint) [44]. PIR updates are sent only when either the coverage constraint or the avoidance constraint is violated. Compared with the periodical and space-driven area-based mechanisms, the PIR-based mechanism maintains high matching accuracy with substantially lower communication overhead. However, they did not consider AOI expansions for the periodical and space-driven area-based mechanisms, which is necessary to ensure their matching accuracy.

Some existing work used the cell-based mechanism to improve the matching (computation) performance of the area-based mechanism. Tan et. al. proposed a hybrid approach which uses the cell-based approach to reduce the candidates for matching in the area-based approach [43]. It recognized the outside irrelevance condition, but it did not recognize the inside irrelevance condition. It assumes a centralized IM coordinator, which may create system bottleneck. Boukerche et. al. proposed a grid-filtered region-based mechanism [13], that also uses the cell-based mechanism to reduce the matching cost of the area-based mechanism. Different from the hybrid mechanism in [43], it assumes an AOI overlaps with every entity in a cell if the overlapping portion of the AOI with the cell exceeds a threshold. It implicitly explores the inside irrelevance filtering condition and further improves the matching performance. However, since its matching is less accurate, more irrelevant status updates are sent. It is based on the particular architecture of the dynamic grid-based mechanism [14] in which each node manages a set of cells and does the matching for these cells. This causes multi-hop matching and routing latencies which affects the consistency between different peers. All these works focus on improving the matching (computation) performance only and do not optimize the communication overhead of IM. In contrast, our proposed hybrid mechanism aims at optimizing communication efficiency (including IM overhead, relevant and irrelevant status updates) in a P2P architecture. In the mean time, it naturally inherits the benefits of optimizing matching performance in the previous works.

Some existing work used the area-based mechanism to improve the performance of the cell-based mechanism. Boukerche and Lu proposed an optimized dynamic grid-based mechanism [12], that uses the area-based mechanism to do matching for each cell to reduce the total number of multicast groups used by the cell-based mechanism. It recognized the outside irrelevance condition but did not recognize the inside irrelevance condition. It is also based on the particular architecture of the dynamic grid-based mechanism [14] and thus has the drawback of multi-hop matching and routing latencies, which affects the consistency between peers. It also assumes that the underlying network has multicast support, which is not available between most peers in the Internet.

The most relevant work to our proposed hybrid mechanism is the work in [2]. It is a multi-tiered IM mechanism in which the first tier uses the cell-based mechanism to roughly filter low fidelity data and then the second tier uses the filtered low fidelity data to do exact matching for high fidelity data with the area-based mechanism. However, it only recognized the outside irrelevance condition and did not recognize the inside irrelevance condition. It also assumes that the underlying network has multicast support, which is not available between most peers in the Internet. Our proposed mechanism should be generally more efficient and flexible for P2P NVEs in the Internet.

Also related to our work is the effort to build structured P2P overlays, such as Distributed Hash Tables (DHT), for networked games [9], [25]. However, they are based on either the traditional area-based IM mechanism [9] or the traditional cell-based IM mechanism [25]. Also, their multi-hop IM query latencies and status update delivery latencies may affect the game consistency. For example, for a game with 5000 players, both the Mercury [9] and SimMud [25] requires more than 6 routing hops. With two players on the Internet, the one hop latency between them may go up to 200ms [31]. So, the routing latencies of Mercury and SimMud may go up to 1.2 seconds. This severely violates the 150ms latency requirement for FPS games as discussed in Section 1. So, these works are not feasible to support FPS games in the Internet.

6. Conclusions and Future Work

Traditional IM mechanisms include periodical and space-driven area-based mechanisms and cell-based mechanisms. This paper proposes a hybrid mechanism that utilizes the cell-based mechanism to reduce AOI updates in the area-based mechanism so as to reduce the communication overhead. A multiplayer P2P game scenario is simulated and the performance results show that, compared to the traditional mechanisms, the hybrid mechanism reduces the upload bandwidth consumption by more than 25.28 percent, reduces the overhead ratio from more than 67.54 percent to only 25.17 percent, and allows more than 5000 players in the Internet to join the same game with today's network upload bandwidth. Although the paper focuses on the discussions of optimizing communication efficiency, the proposed hybrid mechanism naturally inherits the benefits of some previous work in the literature and should also have prominent improvement in matching (computation) performance.

As future work, we are going to carry out a theoretical analysis for each of the traditional and hybrid mechanisms. Through the theoretical analysis, we aim to derive the optimization point of each mechanism. With the analytical results, we are going to carry out more experiments to compare the performance of all the mechanisms in different scenarios by varying the experimental parameters such as AOI size, players' moving speed and the payload size of a status update. To analyze how latency affects the performance of all mechanisms, we also plan to create a network topology based on the King dataset [31] for the experiments.

Acknowledgement

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

References

[1] L. Aarhus, K. Holmqvist, and M. Kirkengen, "Generalized two-tier relevance filtering of computer game update events," in *Proceedings of NetGames 2002*, 2002, pp. 10–13.

[2] H. Abrams, K. Watsen, and M. Zyda, "Three-tiered interest management for large-scale virtual environments," in *In Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, 1998, pp. 125–129.

[3] R. Ayani, F. Moradi, and G. Tan, "Optimizing cell-size in grid-based DDM," in *In Proceedings of the 14th ACM/IEEE/SCS Workshop on Parallel and Distributed Simulation*, 2000, pp. 93–100.

[4] M. Bassiouni, M. H. Chiu, M. Loper, and M. Garnsey, "Relevance filtering for distributed interactive simulation," *Computer Systems Science and Engineering*, vol. 13, no. 1, pp. 39–47, 1998.

[5] N. E. Baughman and B. N. Levine, "Cheating-proof playout for centralized and distributed online games," in *Proceedings of INFOCOM2001, Vol.1*, 2001, pp. 104–113.

[6] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool, "The effects of loss and latency on user performance in Unreal Tournament 2003," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Network and System Support for Games*, 2004, pp. 144–151.

[7] A. Bharambe, J. R. Douceur, J. R. Douceur, J. R. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang, "Donnybrook: Enabling large-scale, high-speed, peer-to-peer games," in *Proceedings of SIGCOMM'08*, 2008, pp. 389–400.

[8] A. R. Bharambe, "Scalable and secure architecutres for online multiplayer games. Thesis Proposal, 2006."

[9] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: Supporting scalable multi-attribute range queries," in *In Proceedings of the SIGCOMM'04*, 2004, pp. 353–366.

[10] A. Bharamebe, J. Pang, and S. Seshan, "Colyseus: A distributed architecture for interactive multiplayer games," in *In Proceedings of NSDI'06*.

[11] Blizzard Entertainment, "World of Warcraft. Available via http://www.worldofwarcraft.com/index.xml."

[12] A. Boukerche and K. Lu, "Optimized dynamic grid-based ddm protocol for large-scale distributed simulation systems," in *In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, 2005.

[13] A. Boukerche, N. J. Mcgraw, and R. B. Araujo, "A grid-filtered region-based approach to support synchronization in large-scale distributed interactive virtual environments," in *In Proceedings of the 2005 International Conference on Parallel Processing Workshops*, 2005, pp. 525–530.

[14] A. Boukerche and A. Roy, "Dynamic grid-based approach to data distribution management," *Journal of Parallel and Distributed Computing*, vol. 62, no. 3, pp. 366–392, 2002.

[15] D. Chen, R. Ewald, and G. K. Theodoropoulos, "Data access in distributed simulations of multi-agent systems," *Journal of Systems and Software (Elsevier)*, vol. 81, no. 12, pp. 2345–2360, 2008.

[16] D. J. Van Hook and J. O. Calvin, "Data distribution management in RTI 1.3," in *In Proceedings of the 1998 Spring Simulation Interoperability Workshop*, 1998.

[17] J. Farber, "Network game traffic modelling," in *In Proceedings of the 1st Workshop on Network and System Support For Games*, 2002, pp. 53–57.

[18] S. Fiedler, M. Wallner, and M. Weber, "A communication architecture for massive multiplayer games," in *In Proceedings of the 1st Workshop on Network and System Support for Games*, 2002, pp. 14–22.

[19] M. Garetto and E. Leonardi, "Analysis of random mobility models with partial differential equations," *IEEE Trans. on Mobile Computing*, vol. 6, no. 11, pp. 1204–1217, 2007.

[20] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating latency between arbitrary internet end hosts," in *In Proceedings of SIGCOMM Internet Measurement Workshop 2002*, 2002, pp. 5–18.

[21] S. Hu, J. Chen, and T. Chen, "VON: a scalable peer-to-peer network for virtual environments," *IEEE Network Magazine*, vol. 20, no. 4, pp. 22–31, 2006.

[22] M. Hyett and R. Wuerfel, "Implementation of the data distribution management services in the RTI-NG," in *In Proceedings of the 2002 Spring Simulation Interoperability Workshop*, 2002.

[23] IEEE, *Standard 1516 (HLA Rules), 1516.1 (Federate Interface Specification) and 1516.2 (Object Model Template)*, 2000.

[24] S. Jose, "IBM and Butterfly to Run Playstation 2 Games on Grid. Available via http://www.hoise.com/primeur/03/articles/weekly/AE-PR-04-03-15.html, Feb. 2003."

[25] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," in *In Proceedings of INFOCOM'04*.

[26] Y. Lee, S. Agarwal, C. Butcher, and J. Padhye, "Measurement and estimation of network QoS among peer Xbox 360 game players," in *PAM 2008*.

[27] J. Lui, "Constructing communication subgraphs and deriving an optimal synchronization interval for distributed virtual environment systems," *IEEE Trans. on Knowledge and Data Engineering*, vol. 13, no. 5, pp. 778–792, 2001.

[28] M. R. Macedonia, M. J. Zyda, D. R. Pratt, D. P. Brutzman, and P. Barham, "Exploiting reality with multicast groups," *Computer Graphics and Applications*, vol. 15, no. 5, pp. 38–45, 1995.

[29] Microsoft Corporation, "Xbox Live website. available via http://www.xbox.com/en-us/live/, 2007."

[30] D. C. Miller and J. A. Thorpe, "SIMNET: The advent of simulator networking," *Proceedings of the IEEE*, vol. 83, no. 8, pp. 1114–1123, 1995.

[31] MIT King Data, "http://www.pdos.lcs.mit.edu/p2psim/kingdata."

[32] K. Morse, L. Bic, and M. Dillencourt, "Interest management in large-scale virtual environments," *Presence: Teleoperators and Virtual Environments*, vol. 9, no. 1, pp. 52–68, 2000.

[33] K. L. Morse and J. S. Steinman, "Data distribution management in the HLA - multidimensional regions and physically correct filtering," in *In Proceedings of the 1997 Spring Simulation Interoperability Workshop*, 1997.

[34] W. Palant, C. Griwodz, and P. Halvorsen, "Evaluating dead reckoning variations with a multi-player game simulator," in *In Proceedings of the 2006 International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2006, pp. 1–6.

[35] L. Pantel and L. C. Wolf, "On the suitability of dead reckoning schemes for games," in *In Proceedings of the 1st Workshop on Network and System Support for Games*, 2002, pp. 79–84.

[36] S. J. Rak and D. J. Van Hook, "Evaluation of grid-based relevance filtering for multicast group assignment," in *In Proceedings of the Distributed Interactive Simulation*, 1996, pp. 739–747.

[37] P. Rosedale and C. Ondrejka, "Enabling player-created online worlds with grid computing and streaming. gamasutra, sept. 2003."

[38] A. Shaikh, S. Sahu, M. Rosu, M. Shea, and D. Saha, "Implementaion of a service platform for online games," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Network and System Support for Games*, 2004, pp. 106–110.

[39] S. Singhal and M. Zyda, *Networked Virtual Environments: Design and Implementation*. ACM Press, 1999.

[40] S. K. Singhal and D. R. Cheriton, "Exploiting position history for efficient remote rendering in networked virtual reality," *Presence: Teleoperators and Virtual Environments*, vol. 4, no. 2, pp. 169–193, 1995.

[41] SingTel, "SingTel broadband service. available via http://www.singnet.com.sg/plans_and_services/broadband/snbb.asp?snbb=15%Matl, 2009."

[42] G. Tan, R. Ayani, Y. Zhang, and F. Moradi, "Grid-based data management in distributed simulation," in *In Proceedings of the 33rd Annual Simulation Symposium*, 2000, pp. 7–13.

[43] G. Tan, Y. Zhang, and R. Ayani, "A hybrid approach to data distribution management," in *In Proceedings of the 4th IEEE/ACM International Workshop on Distributed Simulation and Real-Time Applications*, 2000, pp. 55–61.

[44] X. Tang and W. Cai, "Communication-efficient support for spatial filtering of state updates in distributed virtual environments," in *In Proceedings of the 23rd ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation*, 2009, pp. 129–136.

[45] Valve Software, "Counter Strike. Available via http://www.counter-strike.com/."

[46] R. Waters and J. Barrus, "The rise of shared virtual environments," *IEEE Spectrum*, vol. 34, no. 3, pp. 20–25, 1997.

[47] D. D. Wood, "Implementation of DDM in the MAK high performance RTI," in *In Proceedings of the 2002 Spring Simulation Interoperability Workshop*, 2002.

[48] YVG Staff, "Video game sales break records. Available via http://us.il.yimg.com/videogames.yahoo.com/feature/video-game-sales-break-records/1181404."

[49] Zona Inc., "Terazona: Zona application framework white paper. 2002."