Interest Management for Distributed Virtual Environments: A Survey

ELVIS S. LIU, IBM Research, Ireland, and University College Dublin
GEORGIOS K. THEODOROPOULOS, IBM Research, Ireland

The past two decades have witnessed an explosion in the deployment of large-scale distributed simulations and distributed virtual environments in different domains, including military and academic simulation systems, social media, and commercial applications such as massively multiplayer online games. As these systems become larger, more data intensive, and more latency sensitive, the optimisation of the flow of data, a paradigm referred to as interest management, has become increasingly critical to address the scalability requirements and enable their successful deployment. Numerous interest management schemes have been proposed for different application scenarios. This article provides a comprehensive survey of the state of the art in the design of interest management algorithms and systems. The scope of the survey includes current and historical projects providing a taxonomy of the existing schemes and summarising their key features. Identifying the primary requirements of interest management, the article discusses the trade-offs involved in the design of existing approaches.

Categories and Subject Descriptors: I.6.7 [**Simulation and Modeling**]: Types of Simulation—*Distributed, gaming, parallel*; C.2.4 [**Computer-Communication Networks**]: Distributed System—*Client/Server, Distributed applications*; I.3.2 [**Computer Graphics**]: Graphics Systems—*Distributed/Network graphics*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Virtual reality*

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Interest management, distributed virtual environments, massively multiplayer online games, high-level architecture, data distribution management

ACM Reference Format:
Elvis S. Liu and Georgios K. Theodoropoulos. 2014. Interest management for distributed virtual environments: A survey. ACM Comput. Surv. 46, 4, Article 51 (March 2014), 42 pages.
DOI: http://dx.doi.org/10.1145/2535417

1. INTRODUCTION

A virtual environment is a computer simulation system that provides users with sensory information in such a way that they can readily visualize, explore, and interact with the entities in a virtual landscape. A Distributed Virtual Environment (DVE) is intended for multiple users to interact within a virtual world in real-time even though they are at geographically different locations. In recent years, the emergence of high bandwidth and low latency network infrastructures have fostered the development of large-scale DVE applications, which aim to support hundreds of thousands, if not millions, of participants; Massively Multiplayer Online Games (MMOGs) [Smed et al. 2002] is an indicative class of such systems. The complexity of these systems are increasingly crossing all known boundaries, presenting enormous design, performance,

Authors' addresses: E. S. Liu, IBM Research, Ireland. Mulhuddart, Dublin 15; email: elvisliu@ie.ibm.com; G. Theodoropoulos, IBM Research, Ireland. Mulhuddart, Dublin 15; email: theogeorgios@gmail.com.

DOI: http://dx.doi.org/10.1145/2535417

management, and usability challenges. At the forefront of this effort is providing scalable data distribution services. As DVEs become larger, more data intensive, and more latency sensitive, scalability becomes a crucial element for their successful deployment.

The simplest data distribution approach for a DVE would be to have each host broadcast the state of each virtual entity (e.g., the position of an avatar) that it maintains. This will include, however, data that are not of interest to some receivers. Consider a DVE with n virtual entities and m participants, in which using state broadcasting would result in the system regularly sending $O(nm)$ entity states through the network. As the scale of DVE grows, this approach could consume significant network resources.

An approach to address the scalability problem in this context is to build systems in a way that the flow of data is optimised to reflect the interests of the user population—a paradigm generally referred to as *interest management*. To achieve this goal, interest management systems filter unneeded data before delivering updates to the appropriate participants for processing. This usually involves a process called *interest matching*, which matches the interests of data senders and receivers and hence determines what data should be sent to the participant as well as what data should be filtered.

Over the past two decades, interest management for DVEs has been studied extensively in different domains such as military and academic simulation systems, social media, and commercial applications such as MMOGs, and numerous schemes have been proposed for different application scenarios. Interest management techniques used in some of the early DVEs, mainly in the military domain, were surveyed by Morse [2000] and Morse et al. [2000]. In Boukerche and Roy [2000], the authors described and classified interest management schemes utilised by High-Level Architecture (HLA)-compliant systems. More recently, Ahmed and Shirmohammadi [2009] discussed issues related only to the use of a particular class (zone based) of interest management in MMOGs. Consequently, the need for an up-to-date comprehensive review on the topic of interest management is largely unmet.

It is this conspicuous gap that this article aspires to fill surveying the state of the art in the design of interest management algorithms and systems for DVEs. This review covers current and historical DVEs, ranging from military systems to commercial applications. It is important to note that DVEs are actually one class of distributed simulation systems [Fujimoto 2000] that focus more on real-time and interactive requirements and usually involve some virtual space and moving virtual entities. It is this particular class of systems that this survey is targetting. Many of the interest management approaches surveyed in this article can and have been applied in both DVEs and distributed simulations. Interest management approaches specifically targetting the latter have also been developed (e.g., Suryanarayanan et al. [2010]), but these are considered outside the scope of this survey.

The article first briefly introduces the key design elements of DVEs in Section 2. It then provides a historical overview of the development of interest management techniques in Section 3 before it proceeds to identify and discuss the primary requirements and the trade-offs involved in their design in Section 4. The article then classifies existing interest management algorithms and systems into six major categories, namely *zone-based*, *aura-based*, *class-based*, *visibility-based*, *hybrid*, and *content streaming*. These are reviewed in detail in Sections 5 through 10, respectively. Finally, the article concludes in Section 11. Unless otherwise specified, the terminology used in this article is defined in Appendix A. When HLA-compliant systems are discussed, the HLA terminology is used. The key features of the reviewed approaches for each class are summarised in Appendix B, in Tables I to VI, respectively.

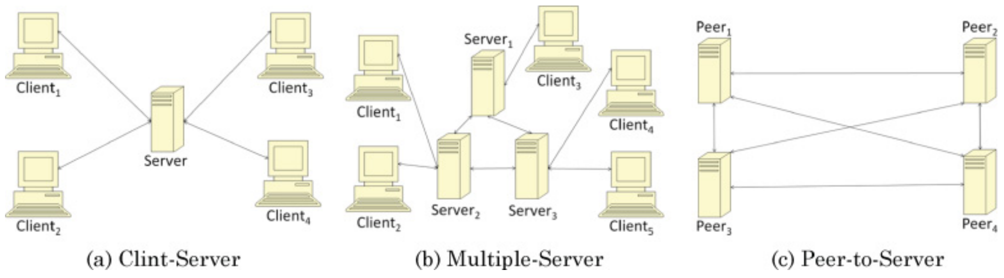

Fig. 1. Communication architectures.

2. DISTRIBUTED VIRTUAL ENVIRONMENTS

Since the development of the first DVEs in the early 1970s, there has been an increasing explosion of interest in the field accompanied by an intense research effort in areas such as data distribution, synchronisation, rendering, and network security. Comprehensive reviews on the subject can be found in Singhal and Zyda [1999] and Steed and Oliveira [2010]. In this section, we briefly outline three pertinent issues on the design of DVEs, which form the basis for the discussion in the subsequent sections: resources, communication architectures, and multicasting.

2.1. Resources

Scalability is perhaps the most crucial requirement for DVE development. A DVE system is said to be scalable if it would remain effective when there is a significant increase in the number of resources and the number of participants. Common DVE resources include network bandwidth and processing power, which have been identified by Singhal and Zyda [1999] as the two most significant bottlenecks in a DVE. Therefore, minimising the demands on these resources is essential to achieve improved scalability and performance.

Based on the resource analysis model given in Singhal and Zyda's book, the interest management technique effectively reduces bandwidth consumption, but at the cost of increasing processor cycles. This in turn introduces a second problem: if the filtering process consumes too many resources, it would be impractical to apply on real-time systems. Therefore, minimising the use of resources, particularly processor cycles, is one of the primary design requirements for a practical interest management scheme. In Section 4, we give a detailed discussion of these design requirements.

2.2. Communication Architectures

The communication architecture is another important component of a DVE because it controls how data is distributed and how participants are synchronised. Numerous architectures have been proposed throughout the years. The three most typical models, *Client-Server*, *Multiple-Server*, and *Peer-to-Peer*(P2P), are illustrated in Figure 1.

2.2.1. Client-Server. Most of the commercial applications, such as MMOGs, adopt the client-server or multiple-server architectures. In the single-server model, a master server plays the most important role in the system. Each client is connected to the server and can only communicate with other clients via the server (Figure 1(a)). The server collects participants' actions from the clients, performs simulations, computes entity states, and sends updates to the clients. If interest management is applied, all filtering processes would be carried out at the server because it has full knowledge of entity states. Obviously, this is the least scalable model among the three, since the master server may eventually become a bottleneck. Moreover, the server is a potential single point of failure—if it fails, the entire DVE would become unavailable to the participants.

2.2.2. Multiple-Server. As its name suggests, the multiple-server architecture (Figure 1(b)) involves multiple servers, with each server serving a number of clients. A common practice is to partition the virtual world into several space subdivisions and assign one or more subdivision(s) to each server. Hence, the workload of simulations and state updates can be shared among the server cluster. Participants who frequently communicate with each other are supposed to connect to the same server. If a partitioning approach is used, the participants connecting to a particular server usually have avatars residing in the subdivision(s) that are controlled by the server. This model is more scalable than the client-server architecture.

2.2.3. Peer-to-Peer. In the P2P model (Figure 1(c)), there is no central repository of entity states. Instead, each peer (client) maintains its own copy of the entity state based on its own simulation results and update messages from all other peers. The virtual world is usually partitioned into a number of space subdivisions, which are controlled by peers. This is similar to the multiple-server architecture with the exception that servers are also clients. The P2P architecture is the most scalable model among the three. It is also more fault tolerant than the two server-based models, since the failure of some peers may not render the whole system unavailable to the participants. However, putting all of the data and simulations on geographically distributed clients introduces management challenges and makes the system more vulnerable to attacks and information exposure.

Examples of running DVEs on P2P networks can be found in Singhal and Zyda's book and in Rueda et al. [2007]. In recent years, building DVEs on structured P2P networks based on Distributed Hash Tables (DHTs) has become increasingly common; representative examples include Knutsson et al. [2004] and Bharambe et al. [2006].

2.3. Multicast Protocols

Multicasting is a technique that is commonly used in DVEs and is closely related to interest management. Many of the interest management schemes that are reviewed in the subsequent sections adopt multicast protocols in their implementation. Multicasting supports one-to-many real-time communication, which delivers a message to a group of computers simultaneously in a single transmission from the source. Multicasting in DVE allows the system to create multicast groups, which consist of participants (called *members*) that are interested in the same set of data and/or events. During runtime, participants may join or leave the multicast groups according to their interests.

The implementation of IP multicast uses specially reserved multicast address blocks in IPv4 and IPv6. The Internet Group Management Protocol (IGMP) adopted on IPv4 networks is used by hosts and adjacent routers to establish multicast group memberships, where multicast management on IPv6 networks is handled by Multicast Listener Discovery (MLD). Detailed descriptions of IP multicast implementations can be found in Singhal and Zyda [1999] and Steed and Oliveira [2010]. In Ahmed et al. [2009], the authors pointed out that IP multicast is not fully deployable on a wide scale on the Internet even in IPv6, and they advocated Application Layer Multicasting (ALM), which implements multicasting functionality at the end hosts instead of network routers.

3. HISTORICAL OVERVIEW OF INTEREST MANAGEMENT IN DISTRIBUTED VIRTUAL ENVIRONMENTS

Although there is no clear consensus on the origin of the interest management paradigm, developers have long been aware that limiting interaction and communication between participants is essential for scaling up the DVE system. In fact, this idea existed even in one of the earliest DVEs in history—the Multi-User Dungeon (MUD) [Bartle 2003], which was created in 1978 and is regarded by many as the origin of today's MMOGs. To enhance scalability, MUD partitions the virtual world into 'rooms' and restricts the participants to see only those virtual entities that are in the same room. This approach was later classified as an example of *zone-based* schemes, which are surveyed in Section 5.

Much of the work in large-scale DVEs for military simulations began with the Simulator Networking (SIMNET) project [Calvin et al. 1993] in the 1980s. To enhance scalability, SIMNET addresses the $O(nm)$ state broadcasting problem by utilising *dead reckoning*, which employs extrapolation models to reduce the frequency of position update of the virtual entities. This approach, however, induces a trade-off between scalability and data consistency. The successor of SIMNET, Distributed Interactive Simulation (DIS) [Neyland 1997], inherits many of SIMNET's features, including the dead reckoning models. In addition, it defines a more generic protocol for data communication—the Protocol Data Units (PDUs). The Entity State Protocol Data Unit (ESPDU) is the most common type of PDUs, which describes the format of an entity state and includes fields such as entity type, position, and velocity. Although DIS itself does not provide a systematic approach for interest management, the ESPDU forms a basis for the design of message filtering mechanisms of the future DIS-based systems, such as those of Srinivasan and Supinski [1995] and Sorroche and Szulinski [2004].

The first academic paper on the use of message filtering in DVE simulation that can be traced is that of Bassiouni et al. [1991]. This paper introduced a filtering scheme based on the circular visibility of simulated vehicles. It asserts that the filtering process can be carried out either at the receiver's or the sender's network gateway. A more generalised version of this approach, the *aura-based* schemes, was later used in some early DVE systems, such as Distributed Interactive Virtual Environment (DIVE) [Carlsson and Hagsand 1993] and Model, Architecture, and System for Spatial Interaction in Virtual Environments (MASSIVE) [Greenhalgh and Benford 1995]. This class of schemes are surveyed in Section 6.

The development of the HLA [DMSO 1998; Morse and Petty 2004] was another major milestone in the evolution of DVE technology and has attracted a vast amount of research work in interest management. HLA provides a generic framework for data filtering, which includes two filtering mechanisms: *value-based* filtering and *class-based* filtering through HLA's Data Distribution Management (DDM) and Declaration Management services, respectively. The DDM was designed to support both zone-based and aura-based schemes. Over the years, a large number of DDM implementations have been proposed, the most influential of which are reviewed in subsequent sections.

Although early efforts in DVE were pioneered by the defence and the academic research communities, large-scale commercial applications, such as MMOGs, have tended to dominate this field since the late 1990s. Popular MMOGs like EverQuest [Sony Online Entertainment 1999], Final Fantasy XI [Square Enix 2002], and World of Warcraft [Blizzard Entertainment 2004], all use zone-based interest management to reduce bandwidth consumption. Later games, such as Guild Wars [NCsoft 2005] and Second Life [Linden Lab 2003], employed interest management schemes for content streaming, which distributes not only the dynamic entity states to the relevant clients but also the static data such as geometry models, scenes, audio files, textures, and animations. This has initiated a new trend in creating very large scale, fully dynamic DVEs. We survey the approaches of content streaming in Section 10.

4. REQUIREMENTS OF INTEREST MANAGEMENT

From the perspective of a participant in a DVE, the semantics of receiving relevant state updates from the system is typically that of a persistent subscription to a uniquely identified stream of updates/events or to all updates/events matching some predicate. To successfully support this operation, we have identified three primary design requirements of interest management schemes, namely *filtering precision*, *runtime efficiency*, and *event-capturing ability*.

4.1. Filtering Precision

As the scale of the DVE grows in terms of participants and virtual entities, using state broadcasting could consume significant network resources such as bandwidth. Therefore, interest management schemes should aim to provide precise filtering mechanisms, which ensure that the participants receive the minimal set of data that are relevant to them, in order to reduce bandwidth consumption. This is in fact the primary goal of interest management.

Apart from the network resources issue, filtering precision is also important for controlling information exposure, especially in commercial applications. For example, in a certain MMOG, a player is not supposed to know the Hit Point (HP) value of other characters even though he or she can see them in the virtual environment. If the interest management system is not carefully designed, all states of nearby characters, including their HP value and position, would be sent to the player's client computer. Although the official game client does not show the HP values on the screen, the player may still use third-party programs to access such information, thus gaining an unfair advantage. A quantitative analysis of information exposure attacks on interest management can be found in Hao and Cai [2012].

4.2. Runtime Efficiency

As described in Section 1, interest management systems have to match the 'interests' of data senders and receivers and hence determine what data should be sent to the participants. If the interests of most of the data senders and receivers are static, the computational overhead of interest matching would be insignificant. However, in real-time DVEs, participants' interests are frequently changed. To reflect these changes, the interest matching process should be carried out frequently, which introduces a significant computational overhead. If the cost of matching is too high, it would be unsuitable for real-time applications such as MMOGs, for which runtime performance is important. Therefore, interest management schemes should provide a way to efficiently minimise the computational overhead of the matching process.

Complex filtering mechanisms may provide good filtering precision; however, their matching process usually introduces more computational overhead. This creates a trade-off between runtime efficiency and filtering precision. Over the years, a number of efficient interest matching algorithms have been proposed, which sought to solve this trade-off. Section 6.2 gives a detailed review of these efforts.

4.3. Event-Capturing Ability

Since DVE participants rely on the interest matching process to determine what events (i.e., messages) to receive, if the process fails to report some of the relevant events, the participants would most likely render incorrect scenes or perform incorrect simulations. Therefore, the interest management schemes should aim to capture and report all relevant events generated by the DVE.

As simulation in most of the DVEs is based on discrete timesteps, the 'missing events' may occur between two discrete interest matching processes. Section 6.2.1 offers an insight into this problem and discusses the existing solutions.

5. ZONE-BASED SCHEMES

Zone-based filtering schemes[1] are perhaps the most widely used approaches for interest management. These schemes limit the participants' interactions and communications within a small number of space subdivisions, or zones. They typically partition the virtual world into a number of zones, with each zone containing a subset of entities. Participants in the DVE are connected to these zones in order to receive events and updates that are generated from them.

The zone-based schemes can be classified into two major categories, namely *disjoint zones* and *seamless zones*.

5.1. Disjoint Zones

The disjoint zone-based approach is the most widely used in the world of MMOGs. The most important feature of this approach is that each participant is restricted to *one* zone at a time. Therefore, they are only allowed to interact with the entities or other participants within their zone of residence, which greatly reduces the update messages that they receive and generate. In addition, this partitioning approach allows the size and shape of the zones, which are usually application dependent, to be freely chosen by DVE developers. Each zone has a number of exit points that allow the participants to change their zone of residence. When a participant's avatar moves into an exit point, it would be disconnected from the old zone and transported to the corresponding exit point of a new zone. Since this is a static one-to-one mapping process, no interest matching is required.

The primary advantage of using disjoint zones is its simplicity of implementation, as the DVE system does not need to perform interest matching at runtime. However, this approach has a very poor filtering precision. Once participants are connected to a zone, they will receive all events and updates from that zone which, naturally, might include irrelevant messages. In addition, this approach also has poor migration transparency. When participants change their zone of residence, they would be halted for a few seconds and would see a *loading screen*. This seriously breaks down the illusion of presence of virtual reality.

As mentioned in the previous section, the MUD was the first multiplayer online game to adopt disjoint zones (called *rooms* in the game). Participants access the game database from inside the rooms, seeing only those entities that are in the same room and moving between rooms via the 'exits' that connect them [Curtis and Nichols 1994]. Many of the most subscribed MMOGs, such as EverQuest [Sony Online Entertainment 1999] and Final Fantasy XI [Square Enix 2002], still use the same idea to partition the virtual world in order to limit the data communication between the participants and entities.

In academic research, the MASSIVE system [Greenhalgh and Benford 1995] is one of the earliest DVE systems that adopts this partitioning approach. The universe is structured as a set of disjoint zones called *worlds*. Each world defines an infinitely large virtual space, which may be inhabited by many concurrent participants. The worlds are connected by *portals*, which allow participants to jump from one world to another.

5.1.1. Instancing. A somewhat different approach utilised by a number of popular MMOGs, such as World of Warcraft [Blizzard Entertainment 2004], Guild Wars [NC-soft 2005], and Final Fantasy XIV [Square Enix 2010], is *instancing*. This approach makes use of *instance zones*, which are special areas, typically dungeons, that generate a new copy of the location for a certain amount of participants that enter the area.

[1]Various other terms have been used in the literature to describe this generic approach, most noticeably *cell-based*, *grid-based*, and *region-based*.

The advantages of using instancing are similar to the original disjoint zones. Having participants in instances tends to spread out the population, instead of concentrating them, which limits the number of potential interactions between participants and virtual entities. Since the participants in the instance do not need to receive updates that are generated outside the instance, and vice versa for the participants outside the instance, there is an overall reduction in network communications. In addition, in some MMOGs, making an instance private for a group of players would prevent them from being disturbed by other players in the same area, depending on the objectives of the game and the experience that the game designer wishes to provide the players.

The main disadvantage of this technique is also similar to the original disjoint zones—when participants enter or leave a zone instance, they would inevitably suffer from the 'loading screen' problem. Moreover, if two participants are in two different instances of the same location, they would not be able to see each other. Through certain means of communication (e.g., chat messages), they might know that they are both in the same place; therefore, not seeing each other would break down the illusion of presence in the virtual reality.

Although the discussion on instancing is less common in the academic literature, a variant approach called *dynamic individual instance* has been proposed in Botev et al. [2008]. In this approach, an instance is created for each participant according to his avatar's *social bias*—an information that is generated automatically based on the avatar's history of interaction with other avatars—whenever avatar density in a certain area exceeds a certain threshold. Interaction across multiple instances is possible because each participant is present in individual instances of all participants that share a similar social bias. This approach preserves the benefits of instancing and, at the same time, allows the participants to interact with other familiar participants without affecting the illusion of presence.

5.2. Seamless Zones

The seamless zone-based approach enables participants to specify an Area Of Interest (AOI) in order to subscribe to multiple partitions. A typical AOI consists of a radius of zones where the participant is joining new zones at the leading edge and leaving old zones at the trailing edge as their avatar moves around the virtual world. In Section 5.2.1, different types of AOIs are discussed in more detail.

The primary advantage of using seamless zones over disjoint zones is that they provide a *seamless* view of the virtual world. In other words, this approach has a better migration transparency—participants would not see a loading screen when they join a new zone. Interest matching, however, is required for this approach. Whenever the AOI moves, the system needs to determine which zone(s) it overlaps. If the number of zones is constant, the computational complexity of the matching process would be $O(m)$, where m is the number of AOIs.

Many systems compliant to HLA DDM [DMSO 1998] adopt this type of scheme. The DDM provides a flexible mechanism for publishing and subscribing to the interest of a participant through the use of multidimensional routing spaces. The basic structure of the routing space is defined as follows:

—*Routing space*: A routing space is a collection of dimensions.
—*Dimension*: Dimensions are used to define regions.
—*Extent*: An extent is a bounded range defined along each dimension of a routing space.
—*Region*: Each region is defined in terms of a set of extents.

The DDM configurations contain two types of regions: a subscription region represents the AOI declared by a federate, whereas an update region represents the set of data made available to other federates. An object is said to be of interest to a federate if and only if at least one update region associated with the object overlaps with at least one subscription region specified by the federate. A typical DDM implementation for the seamless zone-based approach is to define the AOI as subscription regions and the zones as static update regions, and carry out overlap tests between the subscription and update regions during the matching process. Tan et al. [2000c] provides an example of such an implementation.

Unlike disjoint zones, dividing the virtual world into seamless zones requires a proper partition approach. Although most of the virtual worlds are three-dimensional, partitions are usually two-dimensional and can be classified into two main categories: *uniform* and *nonuniform* partitions. These partition approaches are discussed in Sections 5.2.2 and 5.2.3, respectively.

5.2.1. AOI Types. NPSNET [Macedonia et al. 1994, 1995] adopts the circular AOI, which is the most common type of AOI used for seamless zone-based approaches. Typically, the centre of the circular AOI is determined by the coordinates of the avatar's position. If the distance between the centre and a zone is smaller than the radius of the AOI, the participant who controls the avatar subscribes to the updates of the zone. For systems that are compliant to HLA DDM, rectangular AOIs (and zones) are often in use because regions in DDM are all axis aligned. Rak and Van Hook [1996] have published an experimental comparison between circular and square AOIs. Based on the results, they argued that smaller zones more closely approximate the circular AOI, resulting in improved filtering precision.

A different AOI shape, the Field Of View (FOV), is discussed in Bezerra et al. [2008]. The area within an FOV is defined as a circular sector, which is more refined than a circular AOI in that the FOV only subscribes to what the participant can actually *see* (i.e., entities in front). However, this approach has a disadvantage that when the avatar turns 180 degrees, a 'lag' effect might occur to the participant's screen. Since all of the things that it can see now were outside his FOV at the previous moment, a large amount of entity states would suddenly be subscribed by its AOI. As a result, the DVE might not have enough time to deliver all needed data to the participant due to latency and bandwidth limitation. This lag effect would become more serious if the participant keeps turning his or her avatar around very quickly.

Bezerra et al. [2008] also proposed the A^3 AOI, which is a combination of circular AOI and FOV. The A^3 AOI consists of a small circular AOI that subscribes to events within a close area around the avatar, and a large FOV that subscribes to the events that the avatar can actually see. If the avatar turns 180 degrees quasi-instantly, the only entities that would be affected negatively due to abrupt turning are the more distant ones. Therefore, the lag effect on screen would only happen to those distant entities. The authors argued that this is acceptable for most games.

Figure 2 illustrates the four kinds of AOIs that are described in this section. Detailed experimental comparisons of circular, FOV, and A^3 AOIs can be found in the paper of Bezerra et al. [2008], which indicates that A^3 has the best filtering precision among all these AOIs shapes.

5.2.2. Uniform Partitioning. Interest management schemes that adopt uniform partitioning divide the virtual world into zones that are static, are regular, have a uniform orientation, and have uniform adjacency. The most common shapes adopted by the existing approaches are rectangles, hexagons, and triangles, illustrated in Figure 3. For different DVEs, the reasons for adopting one of these shapes are usually application dependent.

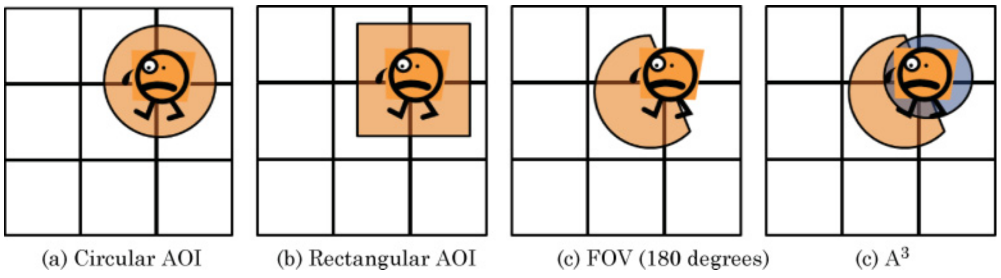

Fig. 2. AOI shapes for seamless zone-based interest management.

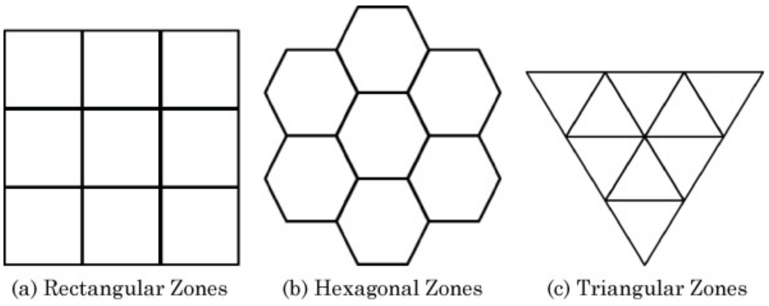

Fig. 3. Uniform partition shapes for seamless zone-based interest management.

5.2.2.1. Rectangles. Van Hook et al. [1994] introduced the concepts of using square zones (coined *grid-based filtering* in the paper). In this approach, multicast addresses are associated with zones defined by a square system overlaid on the terrain. State updates are transmitted to the multicast address of the zone in which an entity is located. Updates may be received from an area by joining the multicast groups for the zones that are in the area. Relevant data is selected indirectly by joining groups for the square zones that fall within an AOI. In a later paper [Rak and Van Hook 1996], a detailed experimental evaluation of this approach is published, focusing on different parameters including size of zones, zone alignment, multicast group join rates, and AOI shapes.

Several variants of rectangular partitioning exist in the literature to further enhance the filtering precision. Smith et al. [1995] proposed a multicast implementation for Modular Semi-Automated Forces (ModSAF), which is a set of DIS-compliant applications. This approach creates seamless zones at two update resolutions. A given ESPDU is updated through either the high-fidelity (i.e., high update frequency) zone address or the low-fidelity zone address, depending on the amount of elapsed time since the last ESPDU was transmitted. All other types of PDUs are updated through the low-fidelity address. Depending on the application, a simulator can subscribe to either or both of high- and low-fidelity information. Smith et al. argued that this approach is especially suitable for wide area viewers (AOIs), such as a long-range radar. Since a wide area AOI may be able to subscribe to all zones, if additional filtering is ignored, it could cause a flood of traffic due to its promiscuous subscriptions. However, by only subscribing to the low-fidelity version of traffic, as Smith et al. proposed, the wide area AOI receives a greatly reduced set of data.

Srinivasan and Supinski [1995] present an approach that defines static multicast groups that are overlaid on square zones. Regular and irregular overlays (Figure 4) are discussed in the paper. Srinivasan et al. asserted that using regular overlays might

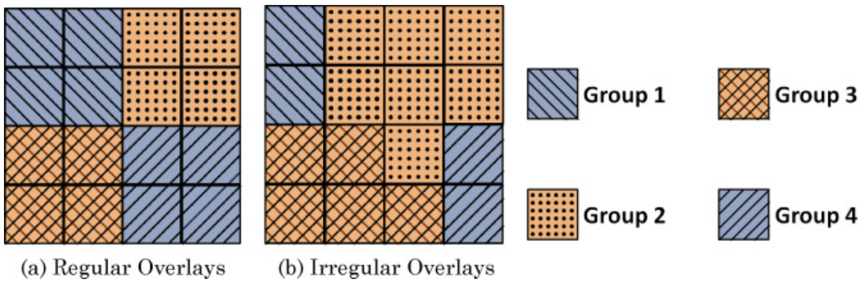

Fig. 4. Multicast group overlays.

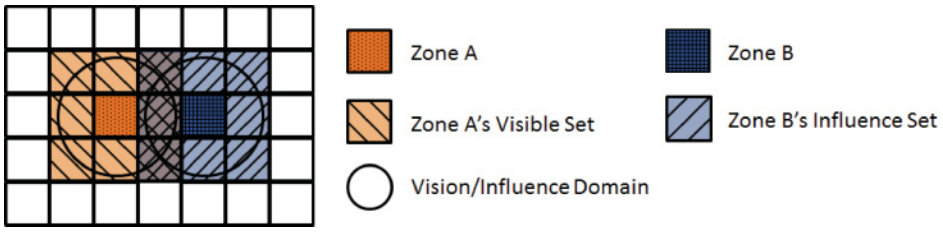

Fig. 5. Tracking-needless grouping.

be wasteful when the groups are being allocated to areas where little or no activity will take place. On the other hand, the irregular overlays can be arbitrarily shaped groups of adjacent square zones, which allow the developers to define multicast groups according to the terrain information. For example, if an impassable mountain separates two groups of entities, the two sides of mountain can be assigned separate multicast groups. Moreover, if a large area has little activity, it can be assigned a single multicast group.

One drawback of this approach, as discussed in the paper, is that using irregular overlays may require relatively more computation overhead. This is because there is no trivial solution for the overlap test between the circular AOI and the multicast groups of arbitrary shape. This problem can be solved by partitioning the virtual world with hierarchical data structures, which offer similar advantages as the irregular multicast groups with efficient matching algorithms (traversal algorithms). These algorithms are described in more detail in Section 5.2.3.

Liu et al. [2004] proposed a *tracking-needless* grouping approach for uniform square zones. This approach assumes that a zone can actually *see* other zones through a circular vision domain. Zones that are overlapped with the vision domain form a visible set, which is precomputed before simulation begins. Similarly, each zone also has an influence domain (and a precomputed influence set) that indicates the area that the zone can influence. During runtime, each participant only subscribes to the visible set of his avatar's residing zone, and the avatar's state update is sent to all members of the corresponding group in the influence set. With this concept, the system does not have to frequently find out what zones are overlapped with the AOIs as the entities move around the virtual world. As a result, the computational overhead of the interest matching process can be saved. However, one assumption of this approach is that all AOIs are the same size as the vision domains. This would be inflexible and might generate many irrelevant update messages. Figure 5 illustrates this approach in a two-dimensional space. If a participant has an avatar located in Zone A, he would receive updates from Zone B since Zone A's visible set overlaps Zone B's influence set.

Zhai et al. [2005] proposed a slightly different approach called the *adaptive grouping* scheme. Unlike tracking-needless grouping, this scheme divides the visible set of a zone into several subsets called *subscription sets*. The zones in the subscription set are visible to participants whose AOI radius is within the range when their avatar resides in any position of the zone. Zhai et al. argued that this approach can reduce the irrelevant messages by lessening the visible zones of the participants.

5.2.2.2. Hexagons. The developers of NPSNET prefer hexagonal over square zones [Macedonia et al. 1995]. The argument put forth is that since the AOI is typically defined by a radius, if squares were used, it would either need to include more area than was necessary (and thus include more entities in the AOI) or use smaller zones (which requires more multicast groups) and compute which zones the AOI should be associated with. On the other hand, using hexagons can more closely approximate a round-shape AOI. When the AOI moves through the virtual space, it uniformly joins and leaves the same number of hexagon zones, which would increase the filtering precision. However, Macedonia et al. did not present a quantitative analysis to support this argument. According to the experimental comparisons presented in Prasetya and Wu [2008], using hexagons actually causes more zone crossing than squares, resulting in an increase in management overhead. Moreover, Prasetya and Wu pointed out that the interest matching process with hexagons is more complicated than squares, since the overlap test between circular AOIs and hexagons usually involves a point-to-line distance formula.

Many variations of hexagonal partitioning exist in the literature Jaramillo et al. [2003], Yu and Vuong [2005], Kazem et al. [2007], and Lu et al. [2010]. Next, we briefly describe these schemes.

Similar to the overlay approach for rectangular zones [Srinivasan and Supinski 1995], Jaramillo et al. [2003] proposed an overlay approach that joins the adjacent hexagonal zones. Participants would subscribe to the updates of a group of hexagonal zones if their AOIs overlap with any zone in the group. The potential benefit of this approach is that there would be less chance of subscribing to and unsubscribing from multicast groups as the participants move around the virtual world. However, in doing so, the participants might receive irrelevant updates from the zones that are not of interest to them. To solve this problem, Jaramillo et al. proposed a load-balancing algorithm to minimise the irrelevant updates by making the zones that generate the most update messages (coined *noisy cells* in the paper) as individual multicast groups. Hence, the chance of subscribing to the irrelevant noisy cells would be reduced.

Kazem et al. [2007] proposed a seamless partitioning approach in a multiple-server architecture. Similar to NPSNET, this approach geographically partitions the virtual world into hexagonal zones and performs message filtering based on the positions and radius of visibility of participant. One key feature of this approach is that it provides a two-layer partitioning. Once the population of a zone is over a load threshold, it triggers a load-balancing process for the zone. This process partitions the zone into fixed number of partitions, as illustrated in Figure 6. When a load-balancing mode is triggered from zone 1, servers 2, 3, 4, 5, 6, and 7 immediately assume responsibility for managing the new adjacent partitions. If, after a period of time, the load drops below the threshold, then the load-balancing mode would be undone.

Lu et al. [2010] proposed a variation of the hexagonal partitioning that employs a hierarchical structure to organise the zones. The authors argued that this would enhance the scalability and managing efficiency of interest management. Moreover, this work addresses the problem of an avatar frequently bouncing in and out at the border of two zones by introducing buffer borderlines.

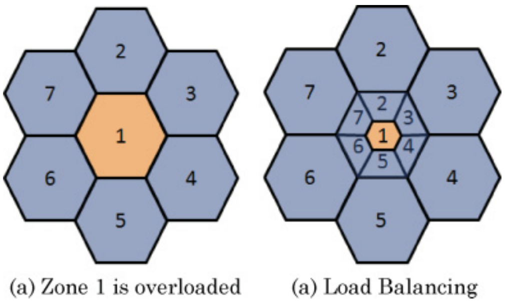

Fig. 6. Two-layer partitioning.

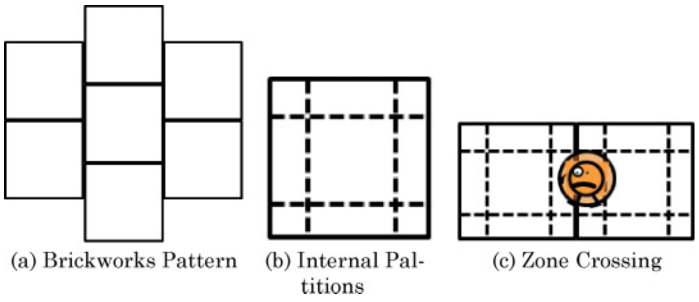

Fig. 7. Brickwork with internal partition.

5.2.2.3. Brickwork with internal partition. Prasetya and Wu [2008] proposed a Brickwork with Internal Partition (BIP) approach, which is designed specifically for mobile gaming. BIP begins with a brickworks pattern, as illustrated in Figure 7(a). This approach can reduce the number of neighbour zones of the original rectangular partitioning approach (Figure 3(a)) and thus achieve the same result of hexagonal partitioning (Figure 3(b)). Each single zone in BIP contains internal partitions, which divide the zone into nine rectangles (Figure 7(b)). Similar to a typical seamless zone-based approach, a participant receives state updates of the entire zone of residence as his avatar moves around the virtual world. However, when the avatar is going to change zones, the participant would not receive all information on possible future zones. Instead, he will only receive partial information as his AOI overlaps with the internal partitions of the new zones (Figure 7(c)). In this way, bandwidth consumption should be lower, as there is less data to transfer during zone crossing.

Prasetya and Wu [2008] also defined a metric for performance analysis specifically for the partition shapes. They performed detailed experimental comparisons between BIP and other typical shapes such as uniform triangles, rectangles, and hexagons based on the rate of (1) zone migration—events when an avatar moves to another zone and (2) zone pre-migration—events when the AOI of an avatar crosses a zone border.

According to the experimental results, for the rate of zone migration, BIP is the lowest among all partitioning shapes and triangular zone is the highest, where hexagonal zone is slightly higher than the rectangular zone. For the rate of zone pre-migration, the rectangular zone is the lowest among all shapes and the hexagonal zone is the highest, where BIP and the triangular zone have no significant difference.

5.2.3. Nonuniform Partitioning. Apart from partitioning the virtual world by some regular pattern, a number of schemes create zones with different shape, size, and even relative orientation. Individual zones can be chosen freely and may be modified dynamically

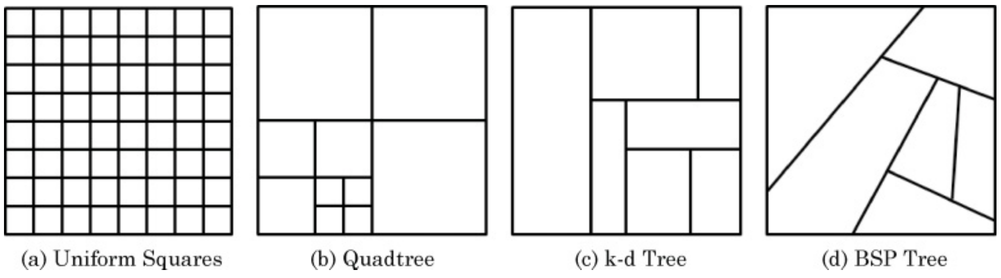

Fig. 8. Partition patterns.

during runtime based on whatever is most convenient from the perspective of designing the individual zones themselves.

One of the earliest examples of nonuniform partitioning is presented in Broll [1997]. The author argued that the uniform partitioning model of NPSNET does not apply well to general-purpose virtual environments, where objects are complex and self-contained. Broll instead proposed a hierarchical structure, which supports zones with arbitrary shapes and individual boundaries. Depending on the 'camera view' of the participant's avatar, connections are established between the participant and all visible zones. This provides a seamless view of the virtual world. However, Broll's approach does not allow virtual entities other than the avatars to travel between different zones. This is somewhat similar to the restriction of the disjoint zones. Another problem of this approach is that since it supports zones with arbitrary shapes, interest matching between the camera view (or AOI) and the zones is not straightforward and may be computationally intensive. This problem, however, is not discussed in Broll's paper.

The majority of nonuniform partitioning schemes employ hierarchical data structures such as Binary Space Partitioning (BSP) trees, k-dimensional (k-d) trees, and quadtrees for space partitioning. Unlike Broll's approach, they place certain restrictions on the construction method of the hierarchy. In this way, however, the computational overhead of the interest matching process can be significantly reduced.

Figure 8 illustrates the partition pattern of uniform square, quadtree, k-d tree, and BSP tree. The following subsections reviews some notable examples of the hierarchical data structures.

5.2.3.1. BSP tree. The Spline platform [Barrus et al. 1996; Waters et al. 1996] decomposes the virtual world into various space subdivisions called *locales*, which may have an variable shape and may be linked together by arbitrary transformations. Each virtual entity resides in exactly one locale, where the participant's interactions are limited to the current locale and its immediate neighbours. Spline employs a BSP tree to describes the boundary of a partition. The basic concepts of BSP tree are presented in Naylor et al. [1990]. While constructing a BSP tree, the system recursively uses a partition plane to split the virtual space into two subdivisions. Each node of the BSP tree represents a partition in virtual space. Each child represents one subdivision of its parent. At the bottom of the hierarchy are the smallest size subdivisions, called *leaf nodes*. Choosing proper partition planes is important for BSP tree construction. It is always desirable to partition the virtual space into two subdivisions that contain the same number of virtual entities in order to keep the tree balanced. A balanced BSP tree allows us to process a query of interest matching in $O(\log n)$ time, where n is the number of entities in the virtual space. The query algorithm starts by comparing the AOI with the root node in order to determine which child(ren) the AOI lie on. It then visits recursively the corresponding child(ren), and when it reaches a leaf node, it checks each of the associated entities for intersection with the AOI in question. For m AOIs, the computational complexity of the matching process is $O(m\log n)$. However, if the tree is unbalanced, in the worst case, each query could be processed in $O(n)$ time. Hence, the whole matching process would become $O(nm)$.

If there is only one participant moving in a static environment, the interest matching process would be very fast. However, a problem with hierarchy is updating it for multiple moving entities, since it changes the structure of the hierarchy. This update process would become time consuming when the number of moving entities is large. In the worse case, this may lead to a total reconstruction of the BSP tree, which takes $O(n\log n)$ time.

5.2.3.2. k-d tree. Other hierarchical data structures, which place more restrictions on the choice of partition planes than the BSP tree, are also common in the literature. In Steed and Abou-Haidar [2003], four partition schemes are presented, including quadtree, k-d tree, constrained k-d tree, and region growing. A k-d tree has similar properties to a BSP tree, except that its partition planes must be axis aligned. In other words, after the partition process, all edges of the space subdivisions in the k-d tree would be parallel or perpendicular to the coordinate axes. Although this is less flexible and may be harder to keep the tree balanced than the BSP tree approach, the query processes can be faster, since comparing an AOI with axis-aligned rectangles is much simpler than with zones of largely unregulated (but still convex) shapes.

Another feature of the approach of Steed and Abou-Haidar [2003] that differs from Spline is that the partition process is only performed when the density (i.e., number of entities per node) of a leaf node is greater than a certain threshold. The choice of threshold is in fact a trade-off and is application dependent. Choosing a large threshold can reduce the height of the k-d tree and make the tree more balanced. As a result, the runtime efficiency of tree construction and matching queries would be increased. However, in doing so, a leaf node may contain more entities, which may reduce the filtering precision.

Choosing a proper partition plane is also important in building a k-d tree. Steed and Abou-Haidar [2003] pointed out that very thin partitions are potentially problematic since the entities may frequently move between partitions. They instead proposed a constrained k-d tree, which specifies that no partition will have one edge longer than a given multiple of the other edge. This is implemented in the recursion by constraining the axis of the partition plane and also the positioning of that plane.

More k-d tree partition strategies can be found in Rieche et al. [2007]. This paper presents a k-d tree approach that supports FOV-type AOI and places the space partitions on a P2P overlay. Zones with too many entities inside can be split, and one part can be sent to another server. Rieche et al. proposed five different partitioning algorithms for the k-d tree:

—*SplitCenter* splits areas along the centre horizontally or vertically in exchange (i.e., if area *A* was split into areas *B* and *B*' horizontally, then area *B* will be split into *C* and *C*' vertically).
—*MaxDistToBorders* splits along the centre. This algorithm decides whether to split horizontally or vertically by maximising the average distance of the entities to the newly created border. The idea behind this is to minimise internal traffic (i.e., messages exchanged between servers), as looking and walking over borders always creates overhead.
—*IntelliDistance* splits along the centre and decides whether to split horizontally or vertically so that as few entities as possible can see the other side of the new border.
—*EqualNumbers* splits along the centre and decides whether to split horizontally or vertically in a way that makes the number of entities in the new zone as equal as

possible. The expectation is that this algorithm will perform better than *SplitCenter* in terms of load balancing.

—*VarAreas* splits horizontally or vertically as *SplitCenter* does. However, it places the border not along the centre but at the centroid of the entities. As with *EqualNumbers*, the expected result is better load balancing.

In addition to splitting, adjacent zones with low numbers of entities can be merged for a lower number of internal messages. The experimental results presented in the paper show that the *VarAreas* algorithm can effectively minimise internal traffic, where *IntelliDistance* performs best on balancing the load (i.e., number of entities) between servers.

5.2.3.3. Quadtree. The quadtree partitioning scheme presented in the paper of Steed and Abou-Haidar [2003] can be traced back to an early paper [Van Hook et al. 1997]. Similar approaches can also be found in a recent paper [Pan et al. 2013] and in an HLA DDM implementation [Eroglu et al. 2008]. A quadtree partitions the two-dimensional space by recursively subdividing it into four equal-size, axis-aligned subdivisions. Since the partition planes are fixed, partitioning by quadtree is less flexible than the BSP tree and k-d tree, but the runtime efficiency of processing a matching query is better. In addition, in Van Hook et al.'s approach, a maximum depth of partitioning is used to limit the height of the tree. Therefore, the tree traversal time would be bounded by a constant, even if the tree is unbalanced. This type of control is stricter than Steed and Abou-Haidar's threshold approach in the sense of limiting the processing time of tree construction and matching queries.

A more generalised version of quadtree, called *N-tree*, is proposed in Shi et al. [2008]. Its underlying partition mechanism is similar to the approach of Van Hook et al. [1997], except that it is designed for multidimensional space. In addition, a predictive method is employed to calculate the level of awareness between entities in order to reduce the frequency of update messages. This method is generally referred to as multiresolution filtering, which is common among aura-based interest management schemes (see Section 6.1).

5.2.3.4. Region growing. Region growing is the last approach presented in the paper of Steed and Abou-Haidar [2003]. It constructs the virtual world from the bottom up, which is based on a standard image processing algorithm for image partitioning. It starts from a seed point and adds adjacent points until a threshold is reached. Once the threshold is reached, a new seed point is chosen. The choice of partitions is according to the actual structure of the virtual world, which is of irregular pattern and arbitrary shape.

5.2.3.5. Voronoi diagram. Although hierarchical data structures are common, they are not the only variable partitioning approaches presented in the literature. In a Voronoi-based Overlay Network (VON) [Hu et al. 2006], a Voronoi diagram is used to define the virtual world and solve the data distribution problem. Given a number of points, or 'sites,' in a two-dimensional world, a Voronoi diagram partitions the world into the same number of zones (Voronoi regions), such that each Voronoi region contains all of the points closer to the region's site than to any other site. For a given participant, an *AOI neighbour* is defined as the Voronoi region whose position is within the participant's AOI. Each Voronoi region is enclosed by a set of other Voronoi regions, known as its *enclosing neighbours*; a participant's AOI may also partially overlap with some set of Voronoi regions, this set is termed its *boundary neighbours*. This Voronoi diagram is continually recomputed by each avatar. Figure 9 illustrates an example of a Voronoi diagram.

When an avatar moves, the update is broadcasted to all connected neighbours. In addition to rendering the update to the participant, each enclosing neighbour will

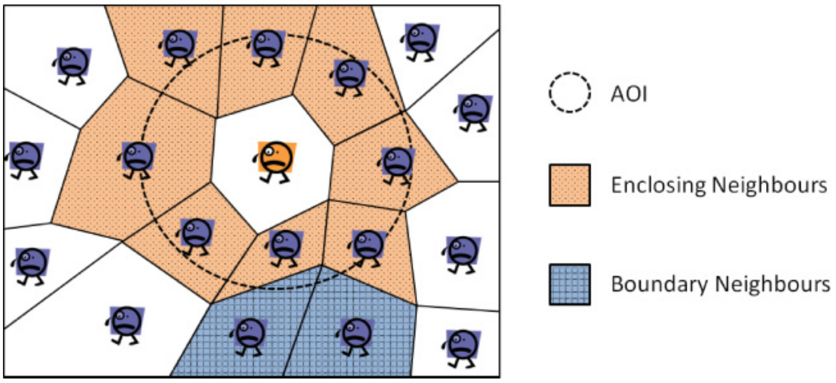

Fig. 9. Voronoi diagram.

perform checks to determine if the move requires adding new boundary neighbours (of which it may not currently be aware). In this manner, no individual participant need ever be aware of the entire diagram, but only that covered by his or her own boundary and enclosing neighbours. This scheme is an elegant way of localising update traffic between participants in line with the geometric relationships between avatars. However, it does come at the significant overhead of continual recomputation of the Voronoi diagram itself, an operation requiring $O(m \log m)$ time, for m neighbours in the boundary set. Once again, in this approach, the basic trade-off encountered in interest management is repeated: filtering precision versus runtime efficiency.

Buyukkaya and Abdallah [2008] presented another Voronoi diagram approach that is similar to VON. In addition to the partitioning scheme, this work focused more on the discovery and interaction of both static and dynamic entities, making it a more practical solution for data distribution.

5.3. Granularity

Choosing a proper granularity is one of the major considerations for all zone-based schemes. For a static partitioning of the virtual world, a significant trade-off must be made. If the zones are large, each zone would contain a large number of virtual entities, and thus the participants might receive a large amount of irrelevant data. On the other hand, if the zones are small, the number of zones as well as the number of multicast groups would become large, and therefore the entity movement between zones would be more frequent. This increases the chance of subscribing to and unsubscribing from multicast groups as the participants move around the virtual world, resulting in an increase in management overheads.

The hierarchical data structures surveyed in Section 5.2.3 also suffer from the same problem but in a different form—a trade-off must be made when choosing a proper granularity of the leaf nodes or a proper height of the hierarchy. Researchers such as Van Hook et al. and Steed and Abou-Haidar tried to maintain a balanced hierarchy by setting a maximum height or a population threshold. These are practical solutions; however, one should also consider the characteristic of the application, as well as the processing power and communication speed of the entire DVE system, when choosing the optimal granularity.

A study presented in Tan et al. [2000c] argued that in a system with fast CPUs and slow communication network, the optimal zone size should be rather small. On the other hand, in a DVE with slower CPUs and faster communication, the optimal zone size would be rather large. In Deen et al. [2006], the authors suggested that adaptive reassignment of zones during runtime may be the best way to achieve and maintain an optimal use of resources. Early experimental investigations on the optimal granularity of certain types of seamless zone-based schemes can be found in Rak and Van Hook [1996] and Ayani et al. [2000]. In the former, Rak and Van Hook were able to identify a certain range of zone size that provides the optimal results in terms of filtering precision and multicast group join rates. These results, however, entirely depend on the simulation settings.

5.4. Communication Architectures

The implementation of zone-based interest management on client-server architecture is straightforward. Each zone represents a multicast group and is assigned a multicast address. Whenever an entity changes its position, the master server determines its residing zone and sends update messages to all members in the corresponding multicast group.

In a multiple-server architecture, the zones are usually distributed across a cluster of servers, and thus each server is responsible for the interest matching and data distribution processes for the zone(s) that they host. It is important to point out that most of the DVEs that implement virtual space partitioning on multiple-server architecture are, in a way, using zone-based interest management for data distribution. For example, some Grid-enabled DVEs such as Gamelet [Wang et al. 2006] and OptimalGrid [Deen et al. 2006] assign each space subdivision to a Grid server, which enables the workload of simulation to be shared among a server cluster. Although these systems do not explicitly address the technique of zone-based interest management, in doing so they essentially reduce the $O(nm)$ message broadcasting problem to a $O(\sum n_i m_i)$ problem, where n_i and m_i are the number of entities and participants, respectively, in each space subdivision i.

Implementing zone-based interest management on structured P2P networks has become increasingly common in DVEs. One influential example of such implementation is reported in Knutsson et al. [2004], wherein the game world is divided into disjoint zones (termed *regions* in the paper). Each player in the game can be a member of only one zone at a time, receiving all updates that occur within the zone. Each zone is hashed (by unique, globally known, game-assigned handles) into a Pastry overlay [Rowstron and Druschel 2001]. The node in the overlay to whom a zone is assigned becomes that zone's coordinator, synchronising all updates to shared entities in the zone. It also becomes responsible for disseminating updates to members of the zone using a Pastry-based ALM infrastructure called *Scribe*.

Another example is the zone federation model proposed in Iimura et al. [2004]. In this model, the entire game world is divided into several seamless rectangular zones, each maintained by an owner node and one or more member nodes. The owner provides zone-local state updates by aggregating modifications from all member nodes and by sending state-change notifications to them. Moreover, it ensures consistency of changes that are committed by other member nodes. DHT harnesses this zoning layer by working as a backup medium for zone data and by providing rendezvous capability—when an independent node becomes a zone member, the new zone member registers its address with the DHT while looking for its zone owner's location by using the DHT.

In a later paper, MOPAR [Yu and Vuong 2005] presented an improvement for the approach of Iimura et al. [2004]. Three types of nodes are introduced in MOPAR: master nodes, slave nodes, and home nodes. The system divides the virtual world into hexagonal zones. Each zone is also bounded to a home node, which is a virtual node, as it does not necessarily have an avatar that resides in the zone. If a newly joined node finds that there is no master node registered for this zone after querying the home node (DHT query), it registers itself as the master node; otherwise, it becomes a slave node. The connections between the master nodes act as the backbone for data distribution, where slave nodes in each zone receive updates from the master nodes. With this approach, the role of Iimura et al.'s owner node is shared by the master and home nodes. The authors argued that this can minimise the use of the DHT, which has overhead for the $O(\log n)$ hops from the source node to the destination node, where n is the number of nodes. Moreover, they proposed a neighbourhood dissemination algorithm that can further minimise the overhead of message dissemination.

6. AURA-BASED SCHEMES

Aura-based is another large class of interest management approaches. It was originally proposed in Bassiouni et al. [1991] (see Section 3) and was later used in the DIVE system [Fahlén and Brown 1992; Carlsson and Hagsand 1993]. The basic idea of this approach is to use *auras* to represent the interests of each entity or participant. An *aura* can be defined as a spatial scope, similar to the AOI described in the previous section. When two auras overlap, a connection between the two owners of the auras is established, and messages are exchanged through the connection. This approach provides a much more precise data-filtering mechanism than the zone-based approaches; however, since the system needs to periodically test the overlap status for the auras, more computational effort is required for interest matching. The computational complexity of the matching process is $O(nm)$ by brute force, where n is the number of entities and m is the number of participants. A detailed experimental comparison between a seamless zone-based scheme and an aura-based scheme can be found in Zou and Diot [2001], which focuses on evaluating the cost of leaving/joining multicast groups as well as the filtering precision.

The MASSIVE system [Greenhalgh and Benford 1995] adopts an alternative version of the aura-based approach called *focus and nimbus* [Benford et al. 1993]. The focus represents the allocation of attention of the participant (or his avatar), whereas the nimbus represents the observed entity's manifestation or observability. A participant (or their avatar) is aware of an entity if and only if his focus overlaps its nimbus. During runtime, focus and nimbus are attached to the entity or the avatar's current position and therefore move dynamically. The system needs to perform overlap tests frequently to keep track of their overlap status.

In HLA DDM, aura-based interest management can be implemented via the update and subscription regions in a way that is similar to the focus and nimbus approach. However, since all regions in DDM are formed by a set of bounded ranges on the dimensional axes, the auras must be defined as an axis-aligned 'orthotope' (i.e., a generalisation of n-dimensional rectangle). An interest matching process must be carried out periodically during runtime in order to determine which update-subscription region pairs overlap. Many of the algorithms surveyed in Section 6.2 have been designed specifically for the DDM interest matching problem. Experimental comparisons for zone-based and aura-based DDM schemes in terms of computational efficiency and filtering precision have been presented in Boukerche and Dzermajko [2001] (note that in the paper, the terms *region-based* and *grid-based* are used to refer to aura-based and zone-based schemes, respectively). The reported results confirm that the aura-based scheme in DDM provides a higher filtering precision than the zone-based scheme but at the cost of computational efficiency.

The three typical aura-based schemes described in this section are illustrated in Figure 10.

Apart from the visual information, aura-based approach can also be applied on the distribution of spatialised audio communications in a DVE. Zimmermann and Liang [2008] present one of such applications, which uses aura-based interest management with stream mixing to find an optimal solution to the many-to-many audio streaming problem.

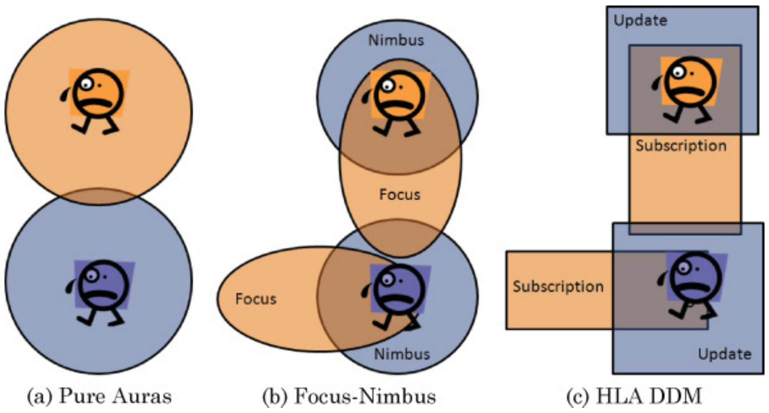

Fig. 10. Aura-based schemes.

6.1. Multiresolution Filtering

In addition to focus and nimbus, another key feature of MASSIVE is that it takes into account the level of awareness—the more an entity's nimbus is within a participant's focus, the higher the level of awareness of it. The system hence is able to calculate the awareness value of information such as graphics and audio, which is used as the basis for managing the interaction between the entity and the avatar. In general, a participant would devote more bandwidth and computational resources to the entities with higher awareness values.

The concept of multiresolution filtering has since been adopted in a few other aura-based approaches. In Donnybrook [Bharambe et al. 2008], participants receive high-fidelity updates only from the entities in their interest set. This is based on an assumption that a human participant can only focus on a constant number of entities at once. Updates of the entities that are not in the participant's interest set are sent at lower fidelity since he is not paying attention to them. The value of attention that a participant is paying to other participants or entities is calculated based on three metrics:

—*Proximity*: Participants are more likely to pay attention to entities nearby. The attention values are inversely proportional to the distance between the participant's avatar and the entity in question. This is essentially similar to the original concept of level of awareness adopted by MASSIVE.
—*Aim*: Participants are more likely to pay attention to entities they aim at. This metric is based on the assumption that a participant's attention is highest on entities in the middle of the screen and falls off for entities are closer to the edges where it becomes unnoticed.
—*Interaction recency*: Participants who recently interacted are more likely to pay attention to each other.

Bharambe et al. [2008] pointed out that different metrics are more important in different situations; therefore, the weights of the three metrics should be varied based on the application scenario. Experimental evaluations presented in the paper suggest that in a first-person-shooter (e.g., Quake III), it is suitable to weight interaction recency 1.5 times more than proximity and aim. Moreover, avatars with melee weapon should be given a higher weight in proximity since they always focus on hitting nearby enemies, whereas avatars with a sniper weapon should be given a higher weight in aim since they always focus on shooting enemies in a distance.

As with most of the interest management approaches, design trade-offs are made by Donnybrook. First, sending updates of the entities that are not in the interest set in an infrequent manner might reduce the event-capturing ability of the interest management system, resulting in a higher chance of incorrect simulation. Second, although such a complex filtering mechanism may achieve a better filtering precision than a typical aura-based approach, it also introduces much computational overhead to calculate the attention values of the entities and participants. Therefore, this approach is only desirable to apply to DVEs with limited bandwidth and/or intensive data communication. In Section 10, we survey interest management approaches used for content streaming, which is a desirable application scenario for multiresolution filtering due to the large amount of data communication.

6.2. Interest Matching Algorithms

Interest matching algorithms are designed to address the trade-off between runtime efficiency and filtering precision, as they seek to ensure that the participants receive the minimal set of data that are of interest to them while reducing the computational overhead involved in the process.

Interest matching is applicable to zone-based systems as well, and several references to this process have been made in the previous section. However, in this survey, we opted to discuss it in detail under the aura-based interest management, as it forms an essential, crucial, and defining characteristic of this approach.

The first paper to investigate the computational complexity of the matching process of the aura-based approach (referred to as *object-based approach* in the paper) was Van Hook et al. [1994]. To address this problem, Van Hook et al. suggested the use of crude zone-based filtering to cull out many irrelevant entities before a more compute-intensive procedure is carried out for finer discrimination. Essentially, this becomes a hybrid approach to combine aura-based and zone-based interest management.

There exist a number of papers in the literature that study the nature of the interest matching problem. An effort has been made in Petty and Morse [2000] to prove that the interest matching process in DDM is not NP-complete by exhibiting a naive polynomial time algorithm (i.e., brute-force interest matching) for the process. In a related effort, Petty and Morse [2004] have provided a theoretical analysis of the computational complexity of the interest matching process. The paper showed by reduction from binary search that the matching process requires a total time with a lower bound in $\Omega(n\log n)$ and a upper bound in $O(n^2)$, where n is the number of times that the federates declare the data that they will send and wish to receive during runtime.

Morgan et al. [2004] have pointed out that collision detection algorithms can be used to aid the process of interest matching in order to enhance runtime efficiency. Collision detection is in fact a well-established field of research in interactive computer graphics [Lin and Gottschalk 1998]. The collision detection algorithms are designed to efficiently determine collisions between virtual objects (usually in the form of polygons). The nature of collision detection and aura-based interest matching is very similar, except the former focuses more on primitive/polygon level overlap tests. In Morgan et al.'s paper, a collision detection algorithm for aura-based interest matching is proposed, which uses circular auras to represent the participants' interests. The authors argued that this algorithm more accurately reflects the groupings of entities that may be interacting than existing collision detection algorithms and provided performance results to demonstrate that this approach is more computational scalable than brute-force interest matching.

Certain features of existing collision detection algorithms have influenced the design of interest matching algorithms. For example, a number of robust interest matching algorithms based on dimension reduction have been proposed in the literature [Raczy

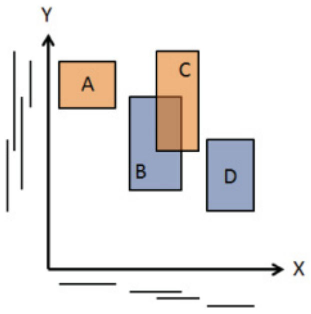

Fig. 11. Dimension reduction.

et al. 2005; Liu et al. 2005; Pan et al. 2007; Ahn et al. 2011]. These algorithms are designed specifically for HLA-compliant systems and thus adopt the use of axis-aligned orthotopal auras. The concept of dimension reduction has first been used in collision detection [Cohen et al. 1995] approaches. The basic idea behind this concept is to reduce the multidimensional overlap test to a one-dimensional problem:

Two axis-aligned orthotopes overlap in n-dimensional space if and only if their orthogonal projections on the 1^{st}*,* 2^{nd}*, . . . , and* n^{th} *dimension overlap.*

Figure 11 illustrates how the concept of dimension reduction works in two-dimensional space. In the figure, B-C overlaps on the x axis, and A-C, A-B, B-C, B-D, and C-D overlap on the y axis; hence, B-C overlaps in two-dimensional space.

Apart from the adoption of dimension reduction, each of these algorithms has some unique features. In Raczy et al. [2005], the algorithm projects the regions on each axis and uses heap-sort to sort the projections in order to find out the overlap information. The computational complexity of such sorting is $O((n+m)log(n+m))$, where n is the number of update regions and m is the number of subscription regions. Liu et al. [2005] proposed a matching algorithm based on caching, which has been adopted by the Lucid Platform—a commercial middleware for MMOG development [Liu et al. 2006]. Similar to Raczy et al.'s approach, it projects the regions on each axis and performs sorting. However, instead of finding the overlap information every time, it caches the matching results of the previous timesteps. In environments where entities make relatively small movements between consecutive timesteps, interest matching can be processed in linear time. Pan et al. [2007] also proposed a sort-based matching algorithm. The efficiency of this approach relies on the assumption that only a small portion of the entities are updated at each timestep of simulations. Pan et al. argued that this algorithm has better storage and computational scalability than Raczy et al.'s algorithm in many cases. In Ahn et al. [2011], a binary partition-based matching algorithm that is based on the divide-and-conquer approach is proposed. The basic idea of this algorithm is similar to a quick-sort, which recursively divides the regions into two partitions that entirely cover those regions. In addition, it adopts a concept called *ordered relation* to avoid unnecessary comparisons within regions in different partitions. Experimental evaluation conducted by Ahn et al. has shown that this approach performs better than Raczy et al.'s matching algorithm across a variety of workloads.

These are by far some of the fastest interest matching algorithms; however, they are designed as sequential processes appropriate for execution on a single processor. As the problem size grows, using these algorithms does not satisfy the scalability requirement of DVE because the single processor may eventually become a bottleneck. Moreover, performance gain would be limited if they are deployed on multiprocessor computers. To solve this problem, in Liu and Theodoropoulos [2009], a parallel

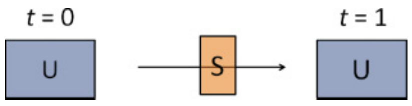

Fig. 12. Missing event.

interest matching algorithm for shared-memory multiprocessors is proposed. It divides the interest matching process into a number of work units by using a hash table and then distributes them across multiple processors. The experimental results show that this approach is more computationally efficient than the serial algorithms when running on shared-memory multiprocessor computers. In a later paper, Liu and Theodoropoulos [2011] presented another parallel algorithm for distributed-memory systems, where communication among processors take places via message-passing protocols. This algorithm is assisted by two associated load-balancing schemes to perform workload redistribution.

6.2.1. The Missing Event Problem. The interest matching algorithms discussed so far focus on enhancing the computational efficiency of the matching process; they have, however, a fundamental disadvantage—they perform interest matching at discrete time intervals. This might lead to 'missing events' in large-scale DVEs that contain virtual entities of greatly varying types. For example, military simulations may involve virtual entities such as infantry men, fighter aircraft, artillery shells, and warships, which are all different in speed and size. If an entity moves around the virtual space at a high speed such that the distance travelled is sufficiently large per timestep, it might move across a small entity without notice. In other words, the events between two consecutive timesteps may be ignored. Since DVE participants rely on the interest matching process to determine what data to receive, if missing interactions are ignored, the participants would most likely render incorrect scenes or perform incorrect simulations. Hence, the participants would often see avatars walk through walls and bullets penetrate soldiers without wounding them. Obviously, the illusion of presence in virtual reality would break down seriously when this happens.

Figure 12 illustrates the *missing event problem* in two-dimensional space. An aura U moves across a static aura S over the time interval [0, 1]. However, neither at time $t = 0$ nor at $t = 1$ can a discrete interest matching algorithm determine that they indeed overlap each other. The event is thus *missed*.

In general, there are three contributing factors to the missing event problem: infrequent interest matching, rapid entity movement, and small aura size, which have been discussed in an early paper [Morse and Steinman 1997]. This paper has also described two simple solutions to the missing event problem, but each must make a trade-off. The first solution is aura expansion (called *sensor range expansion* in the paper), which expands the aura size by a large amount in order to reduce the probability of missing events. For example, an avatar's original line of sight is a 500m radius. If we expand it to 1km, many of the missing events (generated by fast-moving auras) could be captured. However, expanding the line of sight may result in the avatar seeing things that it is not supposed to see. Therefore, the participant controlling the avatar may receive a large amount of irrelevant messages resulting in a bandwidth overheard. A similar solution is discussed in Section 8.6.5. of Fujimoto [2000].

The second approach is frequent matching (called *frequent sampling* in Morse and Steinman [1997]), which reduces the timestep of the simulation and increases the frequency of performing discrete interest matching. For example, suppose that the original timestep is 1 second. If we reduce it to 0.1 second and carry out interest matching 10 times within 1 second, we could avoid many of the missing events. This is, however, a very time-consuming process. Furthermore, it would be meaningless to perform extra matching without aura updates. Additional overhead would be introduced when the frequency of aura update is increased.

Morgan and Lu [2003] proposed a predictive interest management approach to address the missing event problem, which is a message exchange policy based on predictive modelling of entity movements. It aims to vary message exchange between nodes based on the likelihood that entities will influence each other in the near future. Hence, it can avoid missing interactions involving aura and regionalisation by enabling the DVE to ensure that message exchanges occur at an appropriate frequency before, during, and after overlap of auras. The performance of predictive interest management is, however, unclear since the authors have not published the evaluation (in terms of runtime efficiency and the ability of capturing missing events) of this approach.

In Liu and Theodoropoulos [2010a], the authors presented an interest matching algorithm that aims to capture missing events by performing a space-time overlap test for aura pairs. The basic idea of the test is to use bounding volumes called *swept volumes* to bound the trajectory of the auras over each time interval and perform a divide-and-conquer overlap test for the swept volumes in order to increase the accuracy of the overlap tests. Although this approach requires additional effort to compute the swept volumes, a scalable insertion-sort algorithm based on dimension reduction is employed to cull out the aura pairs that are unlikely to overlap, and thus significantly enhances the computational efficiency. In a later paper, Liu and Theodoropoulos [2010b] presented a parallel algorithm that further enhances the computational efficiency of the space-time overlap test by distributing the workload across shared-memory multiprocessors.

6.3. Communication Architectures

In a single-server architecture, the master server is responsible for performing interest matching for all participants and entities; therefore, employing the serial interest matching algorithms (i.e., algorithms that are designed for serial processing) surveyed in Section 6.2 is straightforward.

The serial algorithms are unsuitable for the multiple-server architecture since they are only designed for a single processor. A hybrid implementation of aura-based and zone-based schemes is usually adopted for this scenario. Typically, each server would be assigned a zone, and aura-based interest matching would be performed within that zone. Examples of these hybrid schemes are presented in Section 9.

pSense [Schmieg et al. 2008] presents an example of a structured P2P implementation for aura-based filtering, which has no predefined zones but uses one single continuous space. Every node in a pSense network maintains two lists:

—*Near node list*: A list containing nodes whose avatars are within the visibility range of the participant. This list is updated frequently to keep it as accurate as possible.
—*Sensor node list*: A list containing nodes whose avatars are just outside the visibility range of the participant. The purpose of the sensor node list is to avoid network partitions via communicating with more distant nodes and to detect nodes whose avatar moves closer to the participant.

pSense uses a method called *localised multicast* for position updates in the P2P network. When a participant changes its position, it sends an update message to all nodes in the near node list and the sensor node list. When these nodes receive the update, they forward it to those nodes they know that reside within the visibility range of the participant. The multicast can be indirectly used to determine the relative distance of nodes and thus rearrange the overlay dynamically. However, this method also generates redundant messages in the forwarding process, resulting in an increase in bandwidth consumption.

The HyperVerse [Botev et al. 2008] is another implementation of aura-based filtering on a structured P2P network. Since it is designed for content streaming, we describe it in Section 10.

7. VISIBILITY-BASED SCHEMES

In some cases, a virtual entity would not be of interest to the participants even though it is physically close to their avatar. This usually happens when the entity is behind an obstacle (e.g., a wall) that the participant cannot possibly see. However, the zone-based and aura-based approaches discussed in the previous two sections would still consider that this entity is relevant to the participant since auras, zones, and AOIs are only calculated based on spatial distance.

Visibility-based interest management has been introduced to solve this problem. This class of schemes performs interest matching based on the participant's visibility scope instead of spatial distance. An update message would not be delivered to the participant if the corresponding event happens too far from his avatar or behind an obstacle. In general, the filtering precision of visibility-based approaches is higher than the aura-based approaches. However, the computational overhead is also higher due to the complexity of visibility-based interest matching. In Hosseini et al. [2002], the authors pointed out that visibility-based interest matching can be saved because visibility culling—a process to cull out invisible virtual entities in the scene—is already done by the computer graphics renderer of each client computer. If the rendering results are available to the DVE application, Hosseini et al. argued that they can be used for relevance filtering. However, this approach has a potential problem in which before the local renderer performs culling, the spatial information (e.g., position) of potentially visible entities must be available to the client. If the hosts of these entities broadcast their spatial information to all clients, it would again become the $O(mn)$ message broadcasting problem as described in Section 1.

In Section 7.1 and Section 7.2, we survey several approaches that are designed to reduce the computational overhead of visibility-based interest matching through the adoption of zone-based partitioning.

7.1. Visibility Approximation Using Zones

Since it is in general difficult to compute the precise visibility scope of the participants, a number of the visibility-based schemes decompose the virtual space into zones and use them to approximate the visibility scopes.

The RING system [Funkhouser 1995] is an early DVE that adopts visibility-based interest management. Before simulation starts, the virtual world is decomposed into static, nonuniform, rectangular zones. The system also precomputes the visibility in which the set of zones potentially visible to each zone is determined by tracing beams of possible sight lines through transparent zone boarders. During runtime, servers keep track of which zones contain which entities by exchanging periodic update messages when entities cross zone boarders. Real-time update messages are distributed only to servers and clients containing entities inside some zones visible to the one containing the updated entity. Since an entity's visibility is conservatively overestimated by the precomputed visibility of its containing zone, this approach allows servers to distribute update messages quickly using zone visibility rather than more exact real-time entity visibility computations (which would be computationally intensive).

In Boulanger et al. [2006], the authors argued that the cost of interest matching of perfect visibility-based filtering is relatively expensive. Moreover, the optimal dataset is not always desired since real-time applications (such as MMOGs) often prefetch entity states that are soon to be discovered in order to prevent the 'lag' effect caused by network latency. They instead proposed a zone-based approach, which partitions the

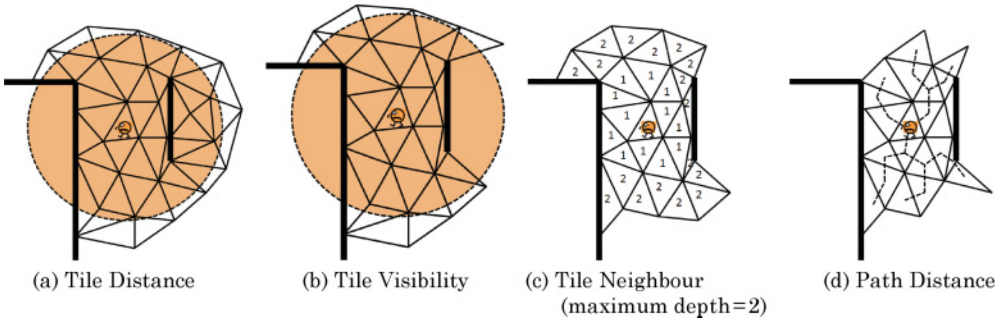

Fig. 13. Visibility algorithms based on Delaunay triangulation.

virtual space by *Delaunay triangulation*, to increase the runtime efficiency of visibility-based filtering. The basic idea of Delaunay triangulation is that the triangles (called *tiles* in the paper) follow the lines of obstacles. An avatar residing in one tile can see entities residing in other tiles in a close neighbourhood. Four algorithms are proposed to determine which tiles are within the visibility scope of the avatar:

—*Tile distance*: The algorithm computes a set of tiles that is an approximation of the participant's AOI (see Figure 13(a)).
—*Tile visibility*: The algorithm computes the visibility between each tile. It takes advantage of the fact that the visibility of tiles is static, as opposed to the visibility of avatars that changes with their position (see Figure 13(b)).
—*Tile neighbour*: The algorithm performs a breadth-first search from the current tile of the participant's avatar and collects all the tiles until it reaches a given maximum depth (see Figure 13(c); the maximum depth is set to 2, and the depth is denoted by the numbers in the triangles).
—*Path distance*: The basic idea of the algorithm is that the subscriber is interested in the tiles within a certain reachable distance from his avatar's current position defined as the path length (see Figure 13(d)).

Based on the Delaunay triangulation approach, a follow-up paper [Denault et al. 2011] presented a mechanism that can distribute the workload (both the workload due to simulations and the workload incurred though interest management) on a multiple-server architecture. Responsibility for triangles can be distributed across servers and dynamically adjusted through load balancing if workload changes.

7.2. Invisible Zones

Apart from approximation, a number of visibility-based approaches calculate the invisible zones of the virtual space from the participant's viewpoint. Entities within the invisible zones cannot be seen by the participant, and therefore no update messages would be generated.

Sudarsky and Gotsman [1996, 1997] proposed the notion of Temporal Bounding Volumes (TBVs). A TBV is an area of space that completely contains an entity for a determined period of time. If an entity moves along a preset trajectory and its maximum velocity and maximum acceleration are known, then a TBV can be computed for the entity based on hierarchical structures such as octree (a three-dimensional version of quadtree) or BSP tree. An entity within its TBV is invisible to a certain participant; therefore, no update would be generated during the validity interval of the TBV. This approach is similar to the occlusion culling technique in computer graphics, which

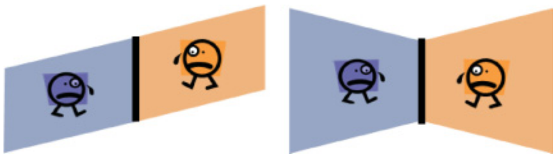
Fig. 14. Examples of UFRs.

prevents invisible objects or surfaces (e.g., they lie behind obstacle objects such as walls) from rendering.

As discussed in Section 5.2.3, the main disadvantage of using hierarchical structures is that updating the hierarchy for multiple moving entities in real time is time consuming. Sudarsky and Gotsman's approach, however, avoids wasting time on hidden moving entities by utilising TBVs. The hidden entities require processing only if their TBVs expire or become exposed. Therefore, the runtime efficiency of hierarchy update can be increased.

Whereas the TBV deals with the visibility culling problem for the entities and the viewpoint of a certain participant (or avatar), an Update-Free Region (UFR) [Makbily et al. 1999] implements a similar concept for mutual visibility of avatars: if two avatars are irrelevant, it is possible to compute two UFRs containing the avatars, such that as long as each avatar is in its UFR, they remain irrelevant. The process for updates between each pair of avatars are as follows: as long as each avatar is inside its UFR, updates need not be sent. When an avatar exits its UFR, a mutual update is required. From that time on, updates are sent between the corresponding participants, and interest matching is performed at every timestep. As soon as the avatars become irrelevant again, new UFRs are computed, and the process repeats. Figure 14 shows some possible examples of UFRs. In the figure, a wall blocks the vision of the avatars, preventing them from seeing each other and thus defining two UFRs in each case.

Faisstnauer et al. [2000] proposed a variant of TBV to manage the priority of update messages via multiresolution filtering. This approach first employs TBV to perform visibility culling in order to reduce the number of relevant entities. The system then uses a Priority Round-Robin (PRR) scheduler including visibility information in the priority of the entities. This method reduces the effort for the system to determine which updates should be sent to each participant. As each participant has its own FOV, the system must frequently examine all entities for each participant. The PRR algorithm shifts part of the matching effort to the scheduler. It repeatedly schedules as many entities (k) as the network permits, achieving an overall effort of $O(k*m) = O(m)$ for m participants. Whenever an entity is selected, the scheduler checks whether it is visible or not. For a visible entity, updates are transmitted; otherwise, the algorithm continues its selection, looking for visible entities, with the highest speed permitted by the computing power and the network bandwidth. The visibility information has an impact on how the entities' priority is determined: visible entities get a priority equal to their velocity; if an entity is invisible, the priority is chosen so as to let the entity be rescheduled when it is expected to become visible again after a period of time. The implementation is based on the prediction of when an entity will be visible again on the shortest path from the actual position to the next visible area (and the velocity of the entity).

7.3. Psychologically Oriented Approaches

A psychologically oriented approach for visibility-based interest management was proposed in Beeharee et al. [2003]. This approach is based on a number of visual attention models that exploit the fact that only feature information (e.g., colour, orientation, and participant activity) can catch the attention of a human. It also describes a phenomenon called *change blindness*, which can be defined as the phenomenon whereby human subjects are unable to notice what should be obvious changes in the scene or images to which they have been directly exposed. By applying the visual attention models, the DVE system can easily prioritise the updates to be sent to the participants and reduce the frequency of low-priority updates. This method is similar to the multiresolution filtering approach of Faisstnauer et al. [2000]. Experiment results suggested that this approach is appropriate for highly populated DVEs, significantly reducing network traffic in such environments.

7.4. Per-Pixel Visibility Test

El Merhebi et al. [2008] present a visibility-based interest management scheme (called *perception-based filtering* in the paper) that is based on the computation of the participant's vision angle and computer screen pixels. They observe that, in general, an entity is not visible due to three properties: (1) it is outside of the participant's FOV, (2) it is hidden by other objects, and (3) it covers less than one pixel on the screen. Based on these properties, they proposed a mathematical framework and algorithms to perform per-pixel visibility tests. They also compared the performance between the proposed scheme and three aura-based schemes. The experimental results suggests that El Merhebi et al.'s approach has the best filtering precision among all of these approaches.

8. CLASS-BASED SCHEMES

Zone-based, aura-based, and visibility-based schemes do not consider filtering aspects other than spatial information. What would happen, for instance, in a military simulation scenario wherein radar should detect all nearby aircraft except stealth fighters? In this case, the position of the *class* of stealth fighters should not be sent to the radar. Normally, the three types of schemes surveyed so far do not support this kind of filtering; however, one exception is the third-party objects [Benford and Greenhalgh 1997] introduced in MASSIVE-2. These objects are created on the basis of spatial information but can also be used for other aspects such as attribute of entities. In this section, we survey some notable filtering schemes that filter events and state updates based on their class.

8.1. Message and Behaviour Classes

A naive approach for class-based filtering is to classify the messages sent by a participant in order to filter the class(es) of messages that are not of interest to other participants. Such a filtering approach based on message class is part of e-Agora [Masa and Žára 2002, 2004], a testbed DVE system aimed at social interaction and cultural content dissemination. In this system, participants can see each other by the help of avatars and communicate by chat and gestures. An approach that applies 'domains' on top of an aura-based scheme is used. It defines eight domains including (W)orld, (S)pace, (R)ooms, logical (G)roups, (C)hat, (N)avigate, (P)lay games, and (E)dit objects. The first three domains are related to spatial information, whereas the others are based on message or action class. A participant can issue the sets: {C, G} for a chat to a group of participants, {E, G, W} for editing within a specific group and a world, or {E, W} for cross-group editing. The combination of domains restricts the scope of the data variable and thus reduces the data communication within the DVE.

Another early example, based on participant's behaviours, is described in Ding and Zhu [2003]. Typical behaviours such as *point*, *grasp*, *touch*, *gaze*, and *talk* are predefined, which describe the interaction between the participants. An interest degree

$I \in [0, 1]$ is used to represent the degree by which an entity is interested, which is somewhat similar to the level of awareness of the MASSIVE system. According to certain application, a user-defined threshold $\delta \in [0, 1]$ is used to judge whether $I > \delta$. If this is the case, the status update of the corresponding entity-participant pair would be communicated. When there is more than one kind of behaviour between two participants, an overall influence is calculated as $I^* = \sum I$.

8.2. Object-Oriented Approaches

The object-oriented approach is a more sophisticated class-based filtering scheme that supports attribute-level filtering and inheritance. The declaration management services of HLA [DMSO 1998; Fujimoto 2000] is the most notable example of this type of schemes. In declaration management, a federate can subscribe to attributes of an object class, indicating that it wishes to receive notification whenever that attribute of any object instance of that class is modified. Furthermore, a federate may subscribe at any point in the class hierarchy. Attributes of a class are inherited by the subclasses of that class. Subscribing to an attribute of a class at a certain level in the hierarchy automatically subscribes the federate to that attribute, as it is inherited by all the subclasses of that class.

One important difference between declaration management and the DDM schemes described in the previous sections is that interest matching of the former can be precomputed, and thus the matching results can be cached and are not modified dynamically during runtime. The DDM schemes, on the other hand, should perform interest matching at runtime due to frequent region updates.

The Interest Operator (IO) [Bartlett 2006] is another example of an object-oriented approach that supports attribute-level filtering. An IO is an operator that accepts parameters, performs a calculation, and returns results. Results are based on whether interest criteria are met. One key difference between IO and the attribute of HLA declaration management is that different classes (with no common class ancestry other that the hierarchy root) may support the same IO. Moreover, in the HLA, an object cannot specify its interest in any attribute or interaction, where the IOs allow developers to establish criteria that may be of interest to both objects or participants.

9. HYBRID SCHEMES

Many systems attempt to combine different interest management schemes to achieve fine-grain filtering and reduce computational overhead. In fact, *pure* zone-based or aura-based schemes are rare and only exist in early work (e.g., Bassiouni et al. [1991] and Van Hook et al. [1994]). Many schemes reviewed in the previous four sections are (or have evolved to be) hybrid schemes, although they may have more features of one scheme than the other. For example, the latest version of DIVE [Benford et al. 1994] uses a 'world hierarchy' to provide zone-based filtering and performs aura-based filtering within each world. The MASSIVE system originally combined disjoint zones and aura-based filtering; in its third generation, MASSIVE-3 [Purbrick and Greenhalgh 2000] integrated the 'locales' method of Spline with the original system to facilitate seamless zone-based filtering. The HLA itself is in fact a good example of a hybrid interest management system. In the HLA interest matching process, an object is said to be of interest to a federate if and only if the following two conditions are satisfied:

(1) At least one of the object's attributes is subscribed to by the federate (through declaration management services); and
(2) at least one update region associated with the object overlaps at least one subscription region of the federate (through DDM).

Essentially, these rules indicate that the HLA provides a combination of class-based and value-based filtering. In this section, we survey some representative hybrid schemes that have been proposed in the literature.

9.1. Zone/Aura-Based Hybrid

The majority of hybrid schemes employ a combination of zone-based and aura-based filtering. Besides DIVE and MASSIVE, there are other notable examples. Abrams et al. [1998] proposed a three-tiered architecture that provides three tiers of message filtering hierarchically. The first and second tiers perform zone-based and aura-based filtering, respectively; in the third tier, protocol-dependent filtering is carried out. A similar zone/aura-based hybrid approach is also reported in Boukerche et al. [2005]. In Pan et al. [2010], the authors proposed a slightly different zone/aura-based hybrid for P2P DVE. This work focuses on reducing not only the computational overhead of aura-based interest matching but also the communication overhead of exchanging aura (or AOI) information between peers.

In HLA DDM, the use of 'region' is flexible—it can be employed for both zone-based and aura-based filtering. This has facilitated a series of hybrid implementations of the two approaches [Boukerche et al. 2000; Tan et al. 2000b; Boukerche and Roy 2002; Park et al. 2004; Boukerche and Lu 2005]. In MGRID [Lo et al. 2009], a dynamic zone/aura-based hybrid approach for DDM interest matching is proposed. MGRID solves the granularity problem (as described in Section 5.3) by using a matching cost model with the profiling information about regions to evaluate the cost of interest matching and dynamically adjust the size of zones during runtime. The experimental results presented in the paper has shown that this approach is more runtime efficient than the matching approaches proposed in Tan et al. [2000b] and Pan et al. [2007].

As described in a lecture delivered at GDC2008 [Guðjónsson 2008], the MMOG EVE Online [CCP Games 2003] also adopts a zone/aura-based hybrid implementation. EVE Online uses a BSP tree to partition the virtual world. Each space ship in the game is characterised by a spherical aura, which can be used to generate a swept volume (called *time-extruded sphere*) over a time interval as the ship moves around the virtual world. The swept volume can be used to traverse the BSP tree, thus determining which space partition it can interact with. Overlapped swept volumes are called *causality bubbles*, which represent a set of events that could potentially influence one another. Information is only shared between the participants in the same set of causality bubbles, resulting in a decrease in bandwidth consumption.

9.2. Other Hybrid Filtering Schemes

The combination of zone- and aura-based scheme has given rise to a clear set of systems described in the previous section. Several other schemes and systems have been proposed that combine several different filtering strategies. These hybrid schemes are reviewed in chronological order next.

The DARPA's Synthetic Theater of War (STOW) programmes that began in the mid-1990s were intended to support military training exercises with tens of thousands of entities. *Hierarchical filtering* [Mellon 1996] is one of the interest management approaches that has been used to improve the scalability of the STOW exercises. In this approach, filtering schemes are organised into three tiers. The first tier uses a multicasting network technology, which routes only the relevant entity states to the simulations that might need that data. The second tier filters execute a Modelling Interest Language (MIL), which carries out filtering on the receiving host before passing the entity states to the local simulation. The third tier filters are a set of user-defined boolean functions, which further optimise the dataset that passes the MIL filters. The first generation of STOW exercises use DIS protocols with customised PDUs. After DIS was succeeded by the HLA standard, an HLA-compliant implementation called *RTI-s* was developed [Rak et al. 1997].

Tan et al. [2000a] argued that the filtering precision of traditional zone-based and aura-based approaches is typically insufficient and would thus cause irrelevant data communication. To address this problem, they proposed an agent-based DDM filtering mechanism that uses intelligent mobile agents to perform *exact* data filtering on the data senders' side. When a subscription is made, agents are launched to the publishers of those attributes and interactions. Whenever the publishers update their own object attributes or interactions, the agents associated with them would perform data filtering locally before sending the relevant data to the subscribers. When a subscription region is modified, the subscribing federate would notify its agents in order to keep internal filtering parameters up-to-date. Experimental evaluation included in Tan et al. shows that the agent-based approach has significant better filtering precision than the zone-based and aura-based approaches. However, in a follow-up paper [Tan et al. 2001], the authors described some of the disadvantages of this approach. First, the data senders must keep an internal table of other federates' publishing interest. Therefore, extra storage space is required. Second, a large number of agents residing on one federate would cause a heavy load to the federate's machine. Therefore, the agent-based mechanism would become computationally inefficient in a large-scale DVE. This is confirmed by the experimental results reported in the follow-up paper.

The DIS Filter-Analyzer (DFA) [Sorroche and Szulinski 2004] developed by the Air Force Distributed Mission Operations Center of Excellence (DMOC) is a set of software filters for DIS-based exercises. It reads DIS PDUs from both LANs and determines if they pass or fail the filter criteria set by the user. Several filter types are available within the DFA: DIS PDU type, DIS Site, Application, and Entity ID, Enumeration filter, and the range filter. The other capabilities include DIS Version translation, a teleport function, a frequency filter, and a delay for PDU processing. Multiple filters can be applied at the same time. For a PDU to be forwarded, it must pass all filters. This approach can provide optimal data transmission; however, the computational overhead of processing all of the filters could be very large.

The Push-Pull framework by Minson and Theodoropoulos [2005] derives the DVE data distribution process at the infrastructure level in a *bottom-up* manner. In this framework, data are represented by a set of Distributed Shared Variables (DSVs). The DVE data distribution process can be translated into read and write operations to the DSVs. Each DSV is associated with a particular owner node (e.g., a server). All writes to a variable are sent to that variable's owner, thus ensuring consistency via this master copy. All nodes who are not the owner of a particular variable are termed *replicators* of that variable. These maintain a proxy locally, which can be read and written by the hosted application in a transparent way. The state of an edge connecting masters and proxies can be maintained in one of two processing modes:

—*Push*, wherein each write at the master results in an update message sent to the proxy
—*Pull*, wherein each read at the proxy results in a read-request/read-response exchange initiated by the proxy

Under the Push-Pull framework, several algorithms were proposed that sought to minimise the number of message passing of the read-write operation. Later papers [Minson and Theodoropoulos 2007a, 2007b] show how the Push-Pull framework can be combined with a zone-based interest management approach, which partitions the virtual world into uniform three-dimensional grids. A quantitative study of the behaviour of this combined approach when it is used in a load-imbalanced DVE is given in Minson and Theodoropoulos [2008].

10. CONTENT STREAMING

So far, this article has focused on how interest management systems distribute dynamic data (e.g., states of moving entities) to virtual world participants. The static data, such as geometry model of the virtual entities, scenes of the virtual world, pre-recorded audio files, textures, and cut scenes, are beyond the concern of these approaches. For most of the existing DVEs, if the developers want to modify the static data, they usually do it in an offline manner. They would either send out a new copy of software to the participants, or simply ask them to download an update patch. Therefore, the participants' machine is required to preinstall all up-to-date static data before simulation starts. It is important to note, however, that most of this data is not of interest to the participants during most of the runtime. As the scale of the virtual world grows, storing a large amount of irrelevant static data may consume significant storage space.

The content streaming technique was introduced to address this problem. This technique distributes the content of the virtual world, including both static and dynamic data, to the participants in a real-time fashion. It allows the participants to enter the virtual world without a complete installation of content. During runtime, content is only transmitted to the participants that are interested in it. One important characteristic of static data is that its size is usually many times larger than dynamic data, resulting in a serious bandwidth consumption. Therefore, interest management schemes with high filtering precision are often preferred by the content streaming systems.

Some systems in this category adopt conventional interest management techniques. An example is Flowing Level-of-Details (FLoD) [Hu et al. 2008, 2010], which adopts a typical seamless zone-based interest management approach that partitions the virtual world into uniform squares. However, the majority of content streaming systems utilise multiresolution filtering to prioritise the transmission of geometry models, thus greatly reducing the bandwidth consumption of streaming. The next section describes the most influential of these systems.

10.1. Utilising Multiresolution Filtering

As discussed in Section 6.1, the multiresolution filtering approach adopted by MASSIVE and Donnybrook is suitable for DVEs that require a large amount of data transaction. This is precisely the requirement of content streaming since the size of geometry data is in general very large.

One of the earliest efforts in this area was reported in Schmalstieg and Gervautz [1996] in the context of strategies for managing network transmission of geometry data in DVEs. Their approach was based on a client-server architecture, which allowed the participant (client) to request geometry from the server based on individual Level Of Detail (LOD) instead of downloading the complete geometry model of virtual entities or even the entire scene. A typical aura-based interest management scheme was used to filter irrelevant contents. When the participant's avatar approaches an entity and the auras overlap, the LODs of the entity's geometry model are transmitted to the participant from coarser to finer resolution. This avoids the transmission of high-resolution geometry data that is never actually used and thus reduces the bandwidth and rendering costs.

Apart from the geometry models and LOD models, multiresolution filtering was utilised by Teler and Lischinski [2001] to stream image-based representations of entities (e.g., texture mapping the image of a complex entity model onto some simple geometry). This approach includes an online optimisation framework for remote rendering, which uses path prediction and a cumulative benefit (i.e., transmission priority) function in order to more efficiently exploit the available bandwidth. The online optimisation algorithm is based on a greedy optimisation strategy. The algorithm repeatedly

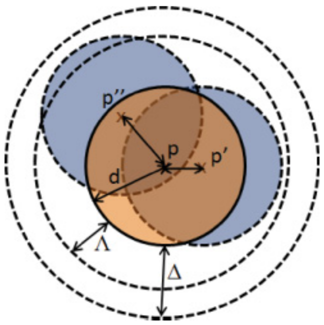

Fig. 15. Auras of HyperVerse.

computes an added benefit integral for all relevant representations of entities that are predicted to become visible and then transmits the representation with the best benefit to cost (i.e., duration of transmission) ratio.

Another interesting content streaming system is CyberWalk [Chim et al. 2003], a Web-based DVE based on on-demand transmission of content. The system employs multiresolution modelling techniques for caching and prefetching entities in a client at various granularities, with nearby entities at higher resolution and distant entities at lower one. Similar to the approach of Schmalstieg and Gervautz [1996] presented earlier, CyberWalk uses an aura-based interest management scheme to filter irrelevant geometry models. However, there are some major differences between these two approaches. First, as Schmalstieg and Gervautz's approach transmits multiple models of the same object at different resolutions, redundant information will be sent. CyberWalk solves this problem by applying a progressive mesh technique for model transmission. Furthermore, in Cyberwalk, an entity is calculated based not only on the distance of the entity from the participant's avatar but also on the size of the entity concerned, the depth of sight of the participant, and the resolution of the display device, allowing the entity models to be transmitted at a low cost. CyberWalk also employs a prefetching mechanism that predicts entities that may be accessed in the near future.

HyperVerse [Botev et al. 2008] is another influential Web-based content streaming system. The client of HyperVerse is similar to a Web browser, which is heterogeneous and contains no predistributed information of the virtual world. Interest management in HyperVerse is somewhat similar to a multiresolution aura-based scheme with the adoption of aura expansion (see Section 6.2.1) as illustrated in Figure 15.

Each participant has three circular auras with radii d, $d + \Lambda$ ($\Lambda < \Delta$), and $d + \Delta$ ($\Delta \geq 0$), respectively, around the position of their avatar (position p). The inner aura represents the visibility scope of the avatar. Initially, all entity and terrain data within the outer aura are delivered to the participant. The primary goal of this scheme is to mitigate the effects of message latency. It allows the avatar to move within a distance Λ (e.g., position *p*') without requesting further data. Whenever the avatar has moved more than Λ (e.g., position *p*"), the auras must be recalculated and all entities and terrain data within the new outer aura need to be delivered to the participant. The introduction of the threshold Λ allows for more time for requesting these data since the participant's inner aura (i.e., visibility) is still $\Delta - \Lambda$ away from areas for which no information has been prefetched. Δ and Λ are chosen by the participants' machines according to their individual capabilities. If they are chosen carefully, message latency can be tolerated without having visual effects. However, using large outer auras would result in the participants receiving a large of amount of irrelevant data. This disadvantage is similar to that of the aura expansion approach. Last but not least, HyperVerse exploits data locality by applying caching techniques. Entity and terrain data with high ‘hit rate’ can be cached locally in order to prevent needless data communication.

10.2. Commercial Applications

The content streaming technique has been used in commercial applications to minimise the size of software distribution. This subsection reviews two representative examples of such systems.

In the social DVE Second Life [Linden Lab 2003; Rosedale and Ondrejka 2003], the content of its virtual worlds is user created and real-time editable. Therefore, updating static data by releasing a new patch or new copy of software is nearly impossible. Second Life solves this problem by employing content streaming as its key feature. Besides the geometry model of the virtual entities, texture and audio data are also delivered to participants through streaming. Progressive techniques are exploited to allow participants to put thousands of entity models, large textures, and a large number of audio sources into a scene and then to stream only the LOD that are needed. In addition, Second Life employs a seamless zone-based scheme to partition the world into uniform square zones. Each zone is managed by a ‘simulator machine’ that handles not only data distribution but also different kinds of simulations within the zone. The simulator machine communicates only to the four nearest neighbours; therefore, it scales as the world becomes large. As the participant moves around the virtual world, a streaming connection is maintained only to the nearby simulator machines. The simulators compute the entities and information that are relevant to the participant and only transmit the data of those entities that are either newly created or that have changed. This allows a thin client software to be the only thing a participant needs to download and install.

In the popular Guild Wars MMOG [NCsoft 2005; Strain and Adams 2005], players only need to download a 90KB thin game client (launch program) before starting the game. When they join the game world, the relevant data continue to stream to them in the background. Generally, players can play the game without seeing a loading screen. Traditional MMOGs such as EverQuest [Sony Online Entertainment 1999] and Final Fantasy XI [Square Enix 2002] update game content by performing scheduled updates or server maintenance. During that time, the players are forced to exit the game and have to download an update patch after the maintenance is finished. Guild Wars, on the contrary, updates the game content dynamically without interrupting the players’ game-play experience or causing the normal patching delays. This provides very good ‘patching transparency’ and the ability to update only the specific files that need to be changed. It is necessary to note that, unlike the DVE systems that are mentioned previously in this section, in Guild Wars, once players have received the files on their hard drive, they will not have to reacquire the data unless they are changed. In this way, the size of the game on the computer will grow as the game content grows, but the relevant data will only be downloaded once.

10.3. Communication Architectures

Both client-server and P2P architectures have been used for content streaming systems. As already mentioned, Schmalstieg and Gervautz [1996] is an example of the former.

Structured P2P overlays based on DHT present a more promising data transmission backbone for content streaming, as they are well suited for sharing vast amounts of data. The HyperVerse is an example of this type of systems, which uses a loosely structured P2P client overlay for data distribution. The basic idea is that each client makes accessible cached information through a torrent-based protocol. When a client requests data of a certain area of the virtual world, a number of other clients can be used to concurrently transfer pieces of data that are already in their caches. For this purpose, the system provides a number of servers to keep track of the clients' auras. A client can retrieve a subset of nearby peers (along with their auras) from the servers and uses this information to determine which peers host the data that it needs. According to the interest management model described in the previous section, by choosing an appropriate value for Λ, the update frequency in individual servers can be kept manageable.

FLoD is another example of P2P streaming framework. All content in FLoD is initially stored at a server, and clients obtain it by streaming from either the server or other clients through P2P overlay. Each peer is able to carry out interest matching between AOI locally. An important assumption of the design of FLoD is that avatars tend to see each other or crowd at certain hotspots. Therefore, a participant may have overlapped visibility with his neighbours. When he needs content data, the neighbours might already possess them. By requesting data from the neighbours first, the server can be relieved from sending the same data repetitively. FLoD exploits this property by using a Voronoi-based approach that is similar to VON [Hu et al. 2006] as its P2P overlay. This approach organises the virtual space into a Voronoi diagram in order to support neighbour discovery, which has been described in Section 5.2.3.

In the realm of commercial systems, although structured P2P overlays are suitable for intensive data transactions, this paradigm has not flourished. This is mainly due to administration and security issues but also problems related to latency and hard real-time constraints [Miller and Crowcroft 2010]. Predominant approaches in the commercial world are server based (peer servers). Nonetheless, there have been efforts in the P2P direction. Recently, Varvello et al. [2009] proposed an approach to deploy Second Life on a structured P2P network, which adopts a partitioning scheme similar to k-d tree.

11. CONCLUSIONS AND FINAL THOUGHTS

The evolution of computer graphics and network technologies over the past three decades has dramatically expanded the application range of DVEs, which have thus diverged into various different platforms such as personal computers, video game consoles, mobile phones, handheld game consoles, and Web-based environments. Each of these platforms has different resources and its own unique requirements. The design of an interest management system highly depends on the platform that adopts it. For instance, a complex filtering scheme used by personal computers may be unsuitable for handheld game consoles due to limited processing power. Therefore, it is important to note that there is no one-size-fits-all approach, which works well in all application scenarios, to interest management. As new DVE domains and computational infrastructures continue to emerge, the problem of interest management will continue to call for new approaches and techniques.

In this article, we have defined the three primary design requirements of interest management algorithms and systems. Based on these requirements, we have discussed the trade-offs that different existing approaches have made and summarised their unique features. We have provided a taxonomy that classifies the existing approaches into different categories, namely zone-based, aura-based, visibility-based, class-based, hybrid, and content streaming schemes. We have also summarised the experimental comparisons of some of these schemes where they have been provided in the literature.

We have tried to focus on the fundamental principles of interest management approaches as they have been proposed and developed through research efforts over the years, describing them in the context of the respective research prototype systems where appropriate. There are relevant exciting developments in the field of commercial games as well; however, these tend to be proprietary systems, and there is a lack of relevant publications. For example, we were only able to obtain a brief description of the design of EVE Online from the GDC2008 lecture. As a result, we have not been able to provide an extensive coverage of such systems in this survey. We still feel, however, that this is an important area of the landscape, and we will continue our efforts to explore it and, hopefully, report our findings in the future.

The application of interest management has evolved considerably since the beginning of this field, becoming increasingly relevant to domains other than traditional DVEs. Content streaming is just an example of a new class of applications. Recently, filtering mechanisms have been applied to mixed reality systems. One representative example is vehicle navigation systems [Yu et al. 2010], which employ AOI to filter the useless spatial information and highlight the relevant ones. Another example is the social network applications for mobile systems [Safar et al. 2009], which allow users to search for friends within the area they are in. We anticipate that the emerging applications of virtual and mixed reality environments, especially on new mobile devices, will further extend the use of the interest management techniques.

Another area where interest management techniques are envisioned to play a critical role is exascale computer systems. As we are moving to the exascale computing era, there is a pressing need to support extreme-scale simulations that will be accessing exascale historical and streaming data. Scalable data distribution will be the key factor to meet the energy and scalability requirements of this new generation of data-intensive analytics approaches [Amarasinghe et al. 2009].

Finally, we note that approaches with goals somewhat similar to interest management, with different terminology and semantics, can be encountered in other areas and at different levels in computer systems research, from hardware design (e.g., cache coherent NUMA approaches [Eggers and Katz 1989] and content addressable memories [Pagiamtzis and Sheikholeslami 2006]) to operating systems (e.g., Linda [Gelernter 1985]) to applications (e.g., relational databases research).

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

REFERENCES

Howard Abrams, Kent Watsen, and Michael Zyda. 1998. Three-tiered interest management for large-scale virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*. 125–129.

Dewan Tanvir Ahmed and Shervin Shirmohammadi. 2009. Zoning issues and area of interest management in massively multiplayer online games. In *Handbook of Multimedia for Digital Entertainment and Arts*. Springer, 175–195.

Dewan Tanvir Ahmed, Shervin Shirmohammadi, and Jauvane C. Oliveira. 2009. A hybrid P2P communications architecture for zonal MMOGs. *Multimedia Tools and Applications* 45, 1–3 (October 2009), 313–345.

Junghyun Ahn, Changho Sung, and Tag Gon Kim. 2011. A binary partition-based matching algorithm for data distribution management. In *Proceedings of the 2011 Winter Simulation Conference (WSC 2011)*.

Saman Amarasinghe, Dan Campbell, William Carlson, Andrew Chien, William Dally, Elmootazbellah Elnohazy, Robert Harrison, William Harrod, Jon Hiller, Sherman Karp, Charles Koelbel, David Koester, Peter Kogge, John Levesque, Daniel Reed, Robert Schreiber, Mark Richards, Al Scarpelli, John Shalf, Allan Snavely, and Thomas Sterling. 2009. ExaScale software study: Software challenges in extreme scale systems.

Rassul Ayani, Farshad Moradi, and Gary Tan. 2000. Optimizing cell-size in grid-based DDM. In *Proceedings of the 14th Workshop on Parallel and Distributed Simulation (PADS'00)*. IEEE Computer Society, 93–100. http://portal.acm.org/citation.cfm?id=336146.336164

John W. Barrus, Richard C. Waters, and David B. Anderson. 1996. Locales and beacons: Efficient and precise support for large multi-user virtual environments. In *Proceedings of the IEEE 1996 Annual International Symposium on Virtual Reality (VRAIS'96)*. 204–213.

Richard Allan Bartle. 2003. *Designing Virtual Worlds*. New Riders Games.

Robert Bartlett. 2006. Interest operators: Facilitating attribute interest criteria for formula-based interest management in distributed virtual environments. In *Proceedings of the 20th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2006)*. 111–118.

Mostafa Bassiouni, H. Williams, and Margaret Loper. 1991. Intelligent filtering algorithms for networked simulators. In *Proceedings of 1991 IEEE International Conference on Systems, Man, and Cybernetics. 'Decision Aiding for Complex Systems.'*

Ashweeni Kumar Beeharee, Adrian J. West, and Roger Hubbold. 2003. Visual attention based information culling for distributed virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*. ACM, 213–222.

Steve Benford, John Bowers, Lennart E. Fahlen, and Chris Greenhalgh. 1994. Managing mutual awareness in collaborative virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*. 223–236.

Steve Benford, Adrian Bullock, Neil Cook, Paul Harvey, Rob Ingram, and Ok-Ki Lee. 1993. From rooms to cyberspace: Models of interaction in large virtual computer spaces. *Interacting with Computers* 5, 2, 217–237.

Steve Benford and Chris Greenhalgh. 1997. Introducing third party objects into the spatial model of interaction. In *Proceedings of the 5th European Conference on Computer-Supported Cooperative Work (ECSCW'97)*. Kluwer Academic Publishers, 189–204.

Carlos Eduardo Bezerra, Fábio R. Cecin, and Cláudio F. R. Geyer. 2008. A3: A novel interest management algorithm for distributed simulations of MMOGs. In *Proceedings of the 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications (DS-RT 2008)*. IEEE Computer Society, 35–42.

Ashwin Bharambe, Jeffrey Pang, and Srinivasan Seshan. 2006. Colyseus: A distributed architecture for online multiplayer games. In *Proceedings of the 3rd Conference on Networked Systems Design & Implementation—Volume 3 (NSDI'06)*. 12.

Ashwin R. Bharambe, John R. Douceur, Jacob R. Lorch, Thomas Moscibroda, Jeffrey Pang, Srinivasan Seshan, and Xinyu Zhuang. 2008. Donnybrook: Enabling large-scale, high-speed, peer-to-peer games. In *Proceedings of the ACM SIGCOMM 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. 389–400.

Blizzard Entertainment. 2004. World of Warcraft. http://www.worldofwarcraft.com/ (accessed October 4, 2013).

Jean Botev, Er Hohfeld, Hermann Schloss, Ingo Scholtes, and Markus Esch. 2008. The HyperVerse: Concepts for a federated and torrent-based "3D Web." In *Proceedings of the 1st International Workshop on Massively Multiuser Virtual Environments (MMVE)*.

Azzedine Boukerche and Caron Dzermajko. 2001. Performance comparison of data distribution management strategies. In *Proceedings of the 5th IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT'01)*. IEEE Computer Society, Washington, DC, 67.

Azzedine Boukerche and Kaiyuan Lu. 2005. Optimized dynamic grid-based DDM protocol for large-scale distributed simulation systems. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*.

Azzedine Boukerche, Nathan J. McGraw, and R. B. Araujo. 2005. A grid-filtered region-based approach to support synchronization in large-scale distributed interactive virtual environments. In *Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW'05)*. IEEE Computer Society, Washington, DC, 525–530. DOI: http://dx.doi.org/10.1109/ICPPW.2005.8

Azzedine Boukerche and Amber Roy. 2000. In search of data distribution management in large scale distributed simulations. In *Proceedings of the 2000 Summer Computer Simulation Conference (SCSC'00)*.

Azzedine Boukerche and Amber J. Roy. 2002. Dynamic grid-based approach to data distribution management. *Journal of Parallel and Distributed Computing* 62, 3 (March 2002), 366–392.

Azzedine Boukerche, Amber J. Roy, and Neville Thomas. 2000. Dynamic grid-based multicast group assignment in data distribution management. In *Proceedings of the 4th IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT'00)*. IEEE Computer Society, 47. http://portal.acm.org/citation.cfm?id=580762.875318

Jean-Sébastien Boulanger, Jörg Kienzle, and Clark Verbrugge. 2006. Comparing interest management algorithms for massively multiplayer games. In *Proceedings of the 5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames'06)*. Article 6.

Wolfgang Broll. 1997. Distributed virtual reality for everyone—a framework for networked VR on the Internet. In *Proceedings of the IEEE 1997 Virtual Reality Annual International Symposium (VRAIS'97)*.

Eliya Buyukkaya and Maha Abdallah. 2008. Data management in Voronoi-based P2P gaming. In *Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC'08)*. 1050–1053.

James Calvin, Alan Dickens, Bob Gaines, Paul Metzger, Dale Miller, and Dan Owen. 1993. The SIMNET virtual world architecture. In *Proceedings of the IEEE Virtual Reality Annual International Symposium (VRAIS'93)*. 450–455.

Christer Carlsson and Olof Hagsand. 1993. DIVE—a platform for multi-user virtual environments. *Computers & Graphics* 17, 6, 663–669.

CCP Games. 2003. EVE Online. http://www.eveonline.com/ (accessed October 4, 2013).

Jimmy Chim, Rynson W. H. Lau, Hong Va Leong, and Antonio Si. 2003. CyberWalk: A Web-based distributed virtual walkthrough environment. *IEEE Transactions on Multimedia* 5, 503–515.

Jonathan D. Cohen, Ming C. Lin, Dinesh Manocha, and Madhav K. Ponamgi. 1995. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*.

Pavel Curtis and David A. Nichols. 1994. MUDs grow up: Social virtual reality in the real world. In *Proceedings of the 39th IEEE Computer Society International Conference (COMPCON Spring'94)*. 193–200.

Glenn Deen, Matthew Hammer, John Bethencourt, Iris Eiron, John Thomas, and James H. Kaufman. 2006. Running Quake II on a grid. *IBM Systems Journal* 45, 1 (January 2006), 21–44.

Alexandre Denault, César Cañas, Jörg Kienzle, and Bettina Kemme. 2011. Triangle-based obstacle-aware load balancing for massively multiplayer games. In *Proceedings of the 10th Annual Workshop on Network and Systems Support for Games (NetGames'11)*. Article 4, 6 pages.

Dawei Ding and Miaoling Zhu. 2003. A model of dynamic interest management: Interaction analysis in collaborative virtual environment. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST'03)*. ACM, 223–230.

DMSO. 1998. High Level Architecture Interface Specification Version 1.3. http://hla.dmso.mil

Susan J. Eggers and Randy H. Katz. 1989. Evaluating the performance of four snooping cache coherency protocols. *ACM SIGARCH Computer Architecture News* 17, 3 (April 1989), 2–15.

Souad El Merhebi, Jean-Christophe Hoelt, Patrice Torguet, and Jean-Pierre Jessel. 2008. Perception-based filtering for MMOGs. *International Journal of Computer Games Technology* 2008, 1 (January 2008), 9 pages.

Omer Eroglu, H. Ali Mantar, and Fatih Erdogan Sevilgen. 2008. Quadtree-based approach to data distribution management for distributed simulations. In *Proceedings of the 2008 Spring Simulation Multiconference (SpringSim'08)*. 667–674.

Lennart E. Fahlén and Charles G. Brown. 1992. The use of a 3D aura metaphor for computer based conferencing and teleworking. In *Proceedings of the 4th MultiG Workshop*. Stockholm.

Chris Faisstnauer, Dieter Schmalstieg, and Werner 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)*. 31.

Richard M. Fujimoto. 2000. *Parallel and Distributed Simulation Systems*. John Wiley and Sons.

Thomas A. Funkhouser. 1995. RING: A client-server system for multi-user virtual environments. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics (I3D'95)*. ACM, 85–92.

David Gelernter. 1985. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 7, 1 (January 1985), 80–112.

Chris Greenhalgh and Steve Benford. 1995. MASSIVE: A collaborative virtual environment for teleconferencing. *ACM Transactions on Computer Human Interactions* 2, 3 (September 1995), 239–261.

Halldor Fannar Guðjónsson. 2008. The Server Technology of EVE Online: How to Cope with 300,000 Players on One Server. Lecture. (September 2008). Game Developers Conference.

Jianan Hao and Wentong Cai. 2012. Measuring information exposure attacks on interest management. In *Proceedings of the 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation (PADS'12)*. 133–144.

Mojtaba Hosseini, Steve Pettifer, and Nicolas D. Georganas. 2002. Visibility-based interest management in collaborative virtual environments. In *Proceedings of the 4th International Conference on Collaborative Virtual Environments (CVE'02)*. ACM, 143–144.

Shun-Yun Hu, Jui-Fa Chen, and Tsu-Han Chen. 2006. VON: A scalable peer-to-peer network for virtual environments. *IEEE Network* 20, 4, 22–31.

Shun-Yun Hu, Ting-Hao Huang, Shao-Chen Chang, Wei-Lun Sung, Jehn-Ruey Jiang, and Bing-Yu Chen. 2008. FLoD: A Framework for Peer-to-Peer 3D Streaming. In *Proceedings of the 27th IEEE International Conference on Computer Communications (INFOCOM 2008)*. 1373–1381.

Shun-Yun Hu, Jehn-Ruey Jiang, and Bing-Yu Chen. 2010. Peer-to-peer 3D streaming. *IEEE Internet Computing* 14, 2, 54–61. DOI:http://dx.doi.org/10.1109/MIC.2009.98

Takuji Iimura, Hiroaki Hazeyama, and Youki Kadobayashi. 2004. Zoned federation of game servers: A peer-to-peer approach to scalable multi-player online games. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Network and System Support for Games (NetGames'04)*. 116–120.

Juan E. Jaramillo, Lina Escobar, and Helmuth Trefftz. 2003. Area of interest management by grid-based discrete aura approximations for distributed virtual environments. In *Proceedings of the 6th Symposium on Virtual Reality (SVR'03)*. 342–353.

Ihab Kazem, Dewan Tanvir Ahmed, and Shervin Shirmohammadi. 2007. A visibility-driven approach to managing interest in distributed simulations with dynamic load balancing. In *Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'07)*. 31–38.

BjLorn Knutsson, Honghui Lu, Wei Xu, and Bryan Hopkins. 2004. Peer-to-peer support for massively multiplayer games. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*.

Ming C. Lin and Stefan Gottschalk. 1998. Collision detection between geometric models: A survey. In *Proceedings of the IMA Conference on Mathematics of Surfaces*. 37–56.

Linden Lab. 2003. Second Life. http://secondlife.com/ (accessed October 4, 2013).

Chia-Hao Liu, Chen-Hsing Wen, and Hsing-Lung Chen. 2004. Tracking-needless grouping: An efficient and scalable grouping scheme in networked virtual environments. In *Proceedings of the 1st IEEE Consumer Communications and Networking Conference (CCNC'04)*. 477–482.

Elvis S. Liu and Georgios Theodoropoulos. 2010a. A continuous matching algorithm for interest management in distributed virtual environments. In *Proceedings of the 24th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS'10)*.

Elvis S. Liu and Georgios Theodoropoulos. 2010b. A fast parallel matching algorithm for continuous interest management. In *Proceedings of the 2010 Winter Simulation Conference (WSC'10)*.

Elvis S. Liu and Georgios K. Theodoropoulos. 2009. An approach for parallel interest matching in distributed virtual environments. In *Proceedings of the 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT'09)*.

Elvis S. Liu and Georgios K. Theodoropoulos. 2011. A parallel interest matching algorithm for distributed-memory systems. In *Proceedings of the 15th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT'11)*.

Elvis S. Liu, Milo K. Yip, and Gino Yu. 2005. Scalable interest management for multidimensional routing space. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST'05)*. 82–85. DOI:http://dx.doi.org/10.1145/1101616.1101633

Elvie S. Liu, M. K. Yip, and G. Yu. 2006. Lucid platform: Applying HLA DDM to multiplayer online game middleware. *ACM Computers in Entertainment* 4, 4, 9. DOI:http://dx.doi.org/10.1145/1178418.1178431

Shih-Hsiang Lo, Cheng-An Chiu, Fang-Ping Pai, Ding-Yong Hong, and Yeh-Ching Chung. 2009. MGRID: A modifiable-grid region matching approach for DDM in the HLA RTI. In *Proceedings of the 2009 Spring Simulation Multiconference (SpringSim'09)*.

Yaping Lu, Yunjia Wang, and Houquan Liu. 2010. An interest management architecture by ALM and region partition for large-scale distributed virtual environment. *Journal of Computers* 5, 6 (June 2010), 836–843.

Michael R. Macedonia, Michael J. Zyda, David R. Pratt, Paul T. Barham, and Steven Zeswitz. 1994. NPSNET: A network software architecture for large-scale virtual environments. *Presence* 3, 265–287.

Michael R. Macedonia, Michael J. Zyda, David R. Pratt, Donald P. Brutzman, and Paul T. Barham. 1995. Exploiting reality with multicast groups: A network architecture for large-scale virtual environments. In *Proceedings of the 1995 IEEE Virtual Reality Annual International Symposium (VRAIS'95)*. 2–10.

Yohai Makbily, Craig Gotsman, and Reuven Bar-Yehuda. 1999. Geometric algorithms for message filtering in decentralized virtual environments. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics (I3D'99)*. ACM, 39–46.

Michal Masa and Jiří Žára. 2002. Generalized interest management in virtual environments. In *Proceedings of the 4th International Conference on Collaborative Virtual Environments (CVE'02)*. ACM, 149–150. DOI:http://dx.doi.org/10.1145/571878.571904

Michal Masa and Jiří Žára. 2004. Interest management for collaborative environments through dividing their shared state. In *Proceedings of the Cooperative Design, Visualization, and Engineering International Conference (CDVE'04)*.

Larry Mellon. 1996. Hierarchical filtering in the STOW system. In *Proceedings of the 14th DIS Workshop on Standards for the Interoperability of Distributed Simulations*.

John L. Miller and Jon Crowcroft. 2010. The near-term feasibility of P2P MMOG's. In *Proceedings of the 9th Annual Workshop on Network and Systems Support for Games (NetGames'10)*. Article 5, 6 pages.

Rob Minson and Georgios Theodoropoulos. 2005. An adaptive interest management scheme for distributed virtual environments. In *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation (PADS'05)*. IEEE Computer Society, 273–281.

Rob Minson and Georgios Theodoropoulos. 2007a. Adaptive support of range queries via push-pull algorithms. In *Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation (PADS'07)*. 53–60.

Rob Minson and Georgios Theodoropoulos. 2007b. An evaluation of push-pull algorithms in support of cell-based interest management. In *Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'07)*. IEEE Computer Society, 39–47.

Rob Minson and Georgios Theodoropoulos. 2008. Load skew in cell-based interest management systems. In *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'08)*. IEEE Computer Society, 43–50.

Graham Morgan and Fengyun Lu. 2003. Predictive interest management: An approach to managing message dissemination for distributed virtual environments. In *Proceedings of the 1st International Workshop on Interactive Rich Media Content Production: Architectures, Technologies, Applications, Tools*.

Graham Morgan, Kier Storey, and Fengyun Lu. 2004. Expanding spheres: A collision detection algorithm for interest management in networked games. In *Proceedings of the 3rd International Conference on Entertainment Computing (ICEC'04)*.

Katherine Morse and Mike Petty. 2004. High level architecture data distribution management migration from DoD 1.3 to IEEE 1516. *Concurrency and Computation: Practice and Experience* 16, 15, 1–17.

Katherine Lee Morse. 2000. *An Adaptive, Distributed Algorithm for Interest Management*. Ph.D. Dissertation. University of California, Irvine.

Katherine L. Morse, Lubomir Bic, and Michael Dillencourt. 2000. Interest management in large-scale virtual environments. *Presence: Teleoperators and Virtual Environments* 9, 1 (February 2000), 52–68.

Katherine L. Morse and Jeffrey S. Steinman. 1997. Data distribution management in the HLA: Multidimensional regions and physically correct filtering. In *Proceedings of the 1997 Spring Simulation Interoperability Workshop (SIW)*. 343–352.

Bruce Naylor, John Amanatides, and William Thibault. 1990. Merging BSP trees yields polyhedral set operations. In *Proceedings of the 17th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'90)*. 115–124.

NCsoft. 2005. GuildWars. http://www.guildwars.com/ (accessed October 4, 2013).

David L. Neyland. 1997. *Virtual Combat: A Guide to Distributed Interactive Simulation*. Stackpole Books.

Kostas Pagiamtzis and Ali Sheikholeslami. 2006. Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. *IEEE Journal of Solid-State Circuits* 41, 3, 712–727.

Ke Pan, Wentong Cai, Xueyan Tang, Suiping Zhou, and Stephen John Turner. 2010. A hybrid interest management mechanism for peer-to-peer networked virtual environments. In *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS'10)*.

Ke Pan, Xueyan Tang, Wentong Cai, Suiping Zhou, and Hanying Zheng. 2013. Hierarchical interest management for distributed virtual environments. In *Proceedings of the 2013 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS'13)*. 137–146.

Ke Pan, Stephen John Turner, Wentong Cai, and Zengxiang Li. 2007. An efficient sort-based DDM matching algorithm for HLA applications with a large spatial environment. In *Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation (PADS'07)*. Washington, DC, 70–82. DOI:http://dx.doi.org/10.1109/PADS.2007.14

ChangHoon Park, Koichi Hirota, Michitaka Hirose, and Heedong Ko. 2004. A flexible and efficient scheme for interest management in HLA. In *Proceedings of the Asian Simulation Conference (AsiaSim'04)*. 141–149.

Mikel Petty and Katherine Morse. 2000. Computational complexity of HLA data distribution management. In *Proceedings of the Fall Simulation Interoperability Workshop*.

Mikel Petty and Katherine Morse. 2004. The computational complexity of the high level architecture data distribution management matching and connecting processes. *Simulation Modelling Practice and Theory*, 3–4, 217–237.

Kusno Prasetya and ZhengDa Wu. 2008. Performance analysis of game world partitioning methods for multiplayer mobile gaming. In *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames'08)*. ACM, New York, NY, 72–77. DOI:http://dx.doi.org/10.1145/1517494.1517509.

James Purbrick and Chris Greenhalgh. 2000. Extending locales: Awareness management in MASSIVE-3. In *Proceedings of the IEEE Virtual Reality 2000 Conference*. IEEE Computer Society, Washington, DC, 287.

Come Raczy, Gary Tan, and J. Yu. 2005. A sort-based DDM matching algorithm for HLA. *ACM Transactions on Modeling and Computer Simulation* 15, 1, 14–38. DOI:http://dx.doi.org/10.1145/1044322.1044324

Steven J. Rak, Marnie Salisbury, and Robert S. Macdonald. 1997. HLA/RTI data distribution management in the synthetic theater of war. In *Proceedings of the 1997 Fall Simulation Interoperability Workshop (SIW)*.

Steven J. Rak and Daniel 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 Simulations*. 739–747.

Simon Rieche, Klaus Wehrle, Marc Fouquet, Heiko Niedermayer, Leo Petrak, and Georg Carle. 2007. Peer-to-peer-based infrastructure support for massively multiplayer online games. In *Proceedings of the 4th Annual IEEE Consumer Communications and Networking Conference (CCNC'07)*. 763–767.

Philip Rosedale and Cory Ondrejka. 2003. Enabling player-created online worlds with grid computing and streaming. *Gamasutra Resource Guide*. http://www.gamasutra.com/resource_guide/20030916/rosedale_pfv.htm (accessed October 4 2013).

Antony I. T. Rowstron and Peter Druschel. 2001. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware'01)*. Springer-Verlag, London, UK, 329–350.

Silvia Rueda, Pedro Morillo, and Juan M. Orduna. 2007. A peer-to-peer platform for simulating distributed virtual environments. In *Proceedings of the 13th International Conference on Parallel and Distributed Systems—Volume 2*. 1–8.

Maytham Safar, Hussain Sawwan, Mahmoud Taha, and Talal Al-Fadhli. 2009. Virtual social networks online and mobile systems. *Mobile Information Systems* 5, 3, 233–253.

Dieter Schmalstieg and Michael Gervautz. 1996. Demand-driven geometry transmission for distributed virtual environments. *Computer Graphics Forum* 5, 3, 421–433.

Arne Schmieg, Michael Stieler, Sebastian Jeckel, Patric Kabus, Bettina Kemme, and Alejandro Buchmann. 2008. pSense—maintaining a dynamic localized peer-to-peer structure for position based multicast in games. In *Proceedings of the 2008 8th International Conference on Peer-to-Peer Computing (P2P'08)*. 247–256.

Xiang-bin Shi, Yue Wang, Qiang Li, Ling Du, and Fang Liu. 2008. An interest management mechanism based on N-Tree. In *Proceedings of the 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'08)*. IEEE Computer Society, 917–922.

Sandeep Singhal and Michael Zyda. 1999. *Networked Virtual Environments: Design and Implementation*. Addison-Wesley.

Jouni Smed, Timo Kaukoranta, and Harri Hakonen. 2002. *A Review on Networking and Multiplayer Computer Games*. Technical Report 454. Turku Centre for Computer Science.

Joshua E. Smith, Kevin L. Russo, and Lawrence C. Schuette. 1995. Prototype multicast IP implementation in ModSAF. In *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Distributed Simulations*. 175–178.

Sony Online Entertainment. 1999. EverQuest. http://everquest.station.sony.com/ (accessed October 4, 2013).

Joe Sorroche and Jerry Szulinski. 2004. Bandwidth reduction techniques used in DIS exercises. In *Proceedings of the 2004 European Simulation Interoperability Workshop (EURO-SIW)*.

Square Enix. 2002. Final Fantasy XI. http://www.playonline.com/ff11us/ (accessed October 4, 2013).

Square Enix. 2010. Final Fantasy XIV. http://www.finalfantasyxiv.com/ (accessed October 4, 2013).

Sudhir Srinivasan and Bronis R. De Supinski. 1995. Multicasting in DIS: A unified solution. In *Proceedings of the 1995 Electronic Conference on Scalability in Training Simulation (ELECSIM'95)*.

Anthony Steed and Roula Abou-Haidar. 2003. Partitioning crowded virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST'03)*. 7–14.

Anthony Steed and Manuel Fradinho Oliveira. 2010. *Networked Graphics: Building Networked Games and Virtual Environments*. Morgan Kaufmann.

Jeff Strain and Dan Adams. 2005. The Tech of Guild Wars. *IGN*. http://www.ign.com/articles/2004/07/29/the-tech-of-guild-wars (accessed October 4, 2013).

Oded Sudarsky and Craig Gotsman. 1996. Output-sensitive visibility algorithms for dynamic scenes with applications to virtual reality. *Computer Graphics Forum* 15, 3, 249–258.

Oded Sudarsky and Craig Gotsman. 1997. Output-sensitive rendering and communication in dynamic virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST'07)*. 217–223.

Vinoth Suryanarayanan, Bart G. W. Craenen, and Georgios K. Theodoropoulos. 2010. Synchronised range queries in distributed simulations of multi-agent systems. In *Proceedings of the 2010 IEEE/ACM 14th International Symposium on Distributed Simulation and Real Time Applications (DS-RT'10)*. 79–86.

Gary Tan, Rassul Ayani, YuSong Zhang, and Farshad Moradi. 2000c. Grid-based data management in distributed simulation. In *Proceedings of the 33rd Annual Simulation Symposium (SS'00)*. IEEE Computer Society, Washington, DC.

Gary Tan, Liang Xu, Farshad Moradi, and Simon Taylor. 2001. An agent-based DDM for high level architecture. In *Proceedings of the 15th Workshop on Parallel and Distributed Simulation (PADS'01)*. IEEE Computer Society, 75–82. http://portal.acm.org/citation.cfm?id=375658.375683

Gary Tan, Liang Xu, Farshad Moradi, and Yusong Zhang. 2000a. An agent-based DDM filtering mechanism. In *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'00)*. IEEE Computer Society, 374. http://portal.acm.org/citation.cfm?id=580760.823771

Gary Tan, YuSong Zhang, and Rassul Ayani. 2000b. A hybrid approach to data distribution management. In *Proceedings of the 4th IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT'00)*. IEEE Computer Society, 55.

Eyal Teler and Dani Lischinski. 2001. Streaming of complex 3D scenes for remote walkthroughs. *Computer Graphics Forum* 20, 3, 17–25.

Daniel J. Van Hook, Steven J. Rak, and James O. Calvin. 1994. Approaches to relevance filtering. In *Proceedings of the 11th Workshop on Standards for the Interoperability of Distributed Simulations*. 26–30.

Daniel J. Van Hook, Steven J. Rak, and James O. Calvin. 1997. Approaches to RTI implementation of HLA data distribution management services. In *Proceedings of the 15th Workshop on Standards for the Interoperability of Distributed Simulations*.

Matteo Varvello, Christophe Diot, and Ernst Biersack. 2009. P2P Second Life: Experimental validation using Kad. In *Proceedings of the 28th IEEE Conference on Computer Communications (INFOCOM'09)*. 1161–1169.

Tianqi Wang, Cho-Li Wang, and Francis C. Lau. 2006. An architecture to support scalable distributed virtual environment systems on grid. *Journal of Supercomputing* 36, 3 (June 2006), 249–264.

Richard C. Waters, David B. Anderson, John W. Barrus, David C. Brogan, Stephan G. McKeown, Tohei Nitta, Ilene B. Sterns, and William S. Yerazunis. 1996. *Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability*. Technical Report. MERL: A Mitsubishi Electric Research Laboratory.

Anthony (Peiqun) Yu and Son T. Vuong. 2005. MOPAR: A mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and video (NOSSDAV'05)*. 99–104.

Jianwei Yu, Qingquan Li, Bisheng Yang, and Jie Yu. 2010. Spatial information filtering for adaptive visualization in vehicle navigation systems. In *Proceedings of the Canadian Geomatics Conference (CGC'10)*.

Zhengjun Zhai, Xiaomei Hu, and Xiaobin Cai. 2005. An adaptive grouping scheme in collaborative virtual environment systems. In *Proceedings of the 2005 International Conference on Cyberworlds (CW'05)*. 311–315.

Roger Zimmermann and Ke Liang. 2008. Spatialized audio streaming for networked virtual environments. In *Proceedings of the 16th ACM International Conference on Multimedia (MM'08)*. 299–308.

Li Zou, Mostafa H. Ammar, and Christophe Diot. 2001. An evaluation of grouping techniques for state dissemination in networked multi-user games. In *Proceedings of the 9th Internation Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. 33–40.

Received February 2007; revised March 2009; accepted June 2009