An Efficient Hybrid Objects Selection Protocol for 3D Streaming over Mobile Devices

by

Mohammad Alja'afreh

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the M.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Electrical and Computer Engineering
University of Ottawa

Abstract

With the rapid development in the areas of mobile manufacturing and multimedia communications, there is an increasing demand for Networked Virtual Environment (NVE) applications, such as Augmented Reality (AR), virtual walk-throughs, and massively multiplayer online games (MMOGs), on hand-held devices. Unfortunately, downloading and rendering a complex 3D scene is very computationally intensive and is not compatible with current mobile hardware specifications nor with available wireless bandwidth. Existing NVE applications deploy client/server based 3D streaming over thin mobile devices, which suffer from single point of failure, latency, and scalability issues. To address these issues, image-based rendering (IBR) and cloud-based 3D streaming have been introduced. The former introduces visual artifacts that reduce, and usually cancel, the realistic behaviors of the Virtual Environment (VE) application, while the latter is considered very expensive to implement. Peer-to-peer (P2P) 3D streaming is promising and affordable, but it has to tackle issues in object discovery and selection as well as content provider strategies. Distributing VE content over a mobile ad-hoc network (MANET) makes the system difficult to update due to the dynamic nature of the mobile clients. In order to tackle these issues, we came up with a novel protocol that combines the pros of both central and distributed approaches. Our proposed hybrid protocol, called OCTET, enables 3D scene streaming over thin devices in a way that can cope with current mobile hardware capabilities and mitigate the challenges of client/server and P2P 3D streaming. In fact, OCTET provides strategies that select, prioritize, and deliver only those objects that contribute to the user's visible scene. OCTET is implemented using the "ns-2" simulation environment, and extensive experiments have clearly demonstrated significant achievements in mobile resource utilization, throughput, and system scalability.

List of Publications Related to the Thesis

- Mohammad Aljaafreh, Haifa Maamar, Azzedine Boukerche, "*An Efficient Object Discovery and Selection Protocol in 3D Streaming-based Systems over Thin Mobile Devices"*, Proceedings of IEEE Wireless Communications and Networking Conference, China, 2013. (Accepted to be presented)

Acknowledgment

First and foremost, all thanks Allah “The Almighty” for giving me the ability and patience to do this work in a timely manner. The tremendous effort and support I received from my academic supervisor Prof. Azzedine Boukerche was the main source of my achievement; I’m really very obliged to him and would like to express my deep thanks and gratitude to him as well. With his guidance I was able to do my research work successfully, and learned how to be serious and a hard worker. He is not only a supervisor but also an example I wish to follow.

Moreover, I would like to extend my sincere thanks to Haifa Mammar whose guidance, patience, and step-by-step support were very invaluable from day one of my master's degree work. I would like also to sincerely thank all of my supportive and friendly team members at PARADISE Research Lab. Special thanks to my best friends Farouk, Maram, Mariam, Farah, Robson, and Badr for their continuous encouragements and helpful suggestions regarding technical problems. I would also like to thank NSERC DIVA research members who enriched my knowledge via their valuable seminars.

Special thanks go to Sarah Duncan-Leclair and Christine Bourbonnais-Hendley for their guidance, help, and never hesitating in answering all my registration’s questions from the first day until the last. It’s veritably a pleasure to deal with such respectful individuals.

Last but not least, many thanks go to my dear mother, father, and siblings who have always encouraged me to pursue my education and to never settle without doing something fruitful.

THANK YOU ALL.

DEDICATION

To my

PARENTS

With love and appreciation

Content

List of Tables

List of Figures

Chapter 1

Introduction

Networked Virtual Environments (NVEs) are computer simulators that integrate networked communications with 3D graphics techniques to deliver immersive, real-time interactions between users that assume their own virtual representations or Avatars [30]. NVEs were originally designed as a military simulator application. In the early 80s, Simulator Networking or SIMNET [10] was the first NVE application consisting of real-time war simulators networked worldwide. SIMNET represents the virtual world as a collection of objects, such as vehicles and weapons, that interact via events like movements. Every war simulator broadcasts its updated events and objects to all other war simulators.

The IEEE Distributed Interactive Simulation (DIS) standard formalized the protocol used in SIMNET [18]. Moreover, recent military simulation research has led to a DIS successor in the high level architecture (HLA) IEEE standard [19], which is an open technology that provides a general architecture for NVE applications. Accordingly, HLA has paved the road for the introduction of new classes of NVE.

Today, multiuser 3D virtual environments are becoming the most popular class of distributed multimedia application and are receiving increased commercial interest, particularly with applications such as massively multiplayer online games (MMOGs), virtual walkthroughs, virtual globes, and augmented reality (AR). For example, World of Warcraft[17] is a role-playing MMOG launched in November 2004 where players enter into epic battles and adventures between two opposing camps. This game had over 15 million

paying subscribers by the end of 2010. Second Life[33], a social NVE created by Linden Lab, reported more than 21 million accounts registered, about 55,000 simultaneous users, and nearly 11 million U.S. dollars worth of virtual item transactions each month by 2011. Google Earth [31] generates a virtual globe whose map and geographical information program was built by Keyhole, Inc, a Central Intelligence Agency (CIA) funded company acquired by Google in 2004. It has about 400 million users who explore the world in a 3D view from their devices.

Currently, before a user can interact with a virtual environment application, they require the entire 3D content (mesh models, texture, animations, etc.) to be stored locally. This can be achieved via full optical disk installation or network download on the client machine. The full pre-install and/or download approaches are becoming less feasible due to the following factors [2,28,30]:

1. **Bigger and more dynamic content**: In order to provide realistic behaviour, the content of VEs typically become larger and more dynamic. Today, Second Life [33] depends on delivering over 34 terabytes of user-created models, textures, and behaviour scripts. NASA World Wind[34], a virtual globe application, currently has over 4.6 terabytes of data.
2. **Massive number of environments:** Web3D is a future vision to fully display and navigate internet web sites using 3D technologies. It is almost impossible for users to pre-install millions of 3D sites, before navigating each one of them.
3. **Hardware limitation of mobile devices:** The limited memory and storage capabilities of thin mobile devices, such as the iPhone and personal digital assistant (PDA), make it extremely difficult to enable a VE class of application in such devices[1].

[1] The terms user, node, mobile, device, client and peer will be used interchangeably from now on.

To address these issues, it is desirable to only get the specific content of interest progressively through 3D streaming techniques.

1.1 What is 3D Streaming?

Continuous, real-time delivery of 3D content, such as meshes, textures, and animations, over network connections allows user interactions with a virtual world without requiring a full download or pre-installation [2,30,34]. Users can immediately begin rendering the 3D content when it is only partially received, and thus interaction with their VE could start without having to wait for the entire download to be completed.

1.2 Comparison Between 3D Streaming and Media Streaming

3D streaming is similar to media streaming in that users can immediately interact with the displayed scene while the data is gradually downloaded. However, as illustrated in Table 1.1, different characteristics exist between the two streaming approaches.

Characteristic	3D streaming	Media streaming
Content	3D Mesh Models	Image based
Access pattern	Non-linear	Linear
Transmission sequence	Unique	Same as online video streaming
Predictability	Non-predictable	Predictable

Table 1.1: 3D streaming VS media streaming

The first difference lies in the content itself. Mesh models, textures, and animations are used in 3D streaming while 2D images are used in media streaming. 3D streaming content allows the user to manipulate the scene without any additional streaming tasks because local processing can implement the changes with the current data. In other words, the user can

interact with and navigate the same scene at all possible resolutions and from any viewing angle without having to download any extra data. However, in media streaming when the user wants to change the viewing angle, extra images for the scene have to be streamed immediately.

In media streaming, the video is fragmented into sequentially ordered frames. These frames are transmitted according to the time sequence of the video. Hence, the data stream would be the same for everyone, and this media streaming access pattern is considered to be linear. This allows frame prediction to be applied. In 3D streaming, the 3D content cannot be ordered and must be fragmented based on the user's behaviour, viewing angle, and distance from the viewed objects. Different viewing angles or distances would produce individually different transmission sequences. Therefore, the 3D content access pattern in 3D streaming is considered non-linear and less predictable.

1.3 Problem Statement and Motivations

As hardware manufacturers improve hand-held device performance and capabilities through 3G and IEEE 802.11 chips, better computational capabilities, and wider screens with high resolutions, there is a growing demand from users to access NVEs interactively with these devices. Several multimedia applications, such as AR, GPS applications, and walk-throughs, try to implement 3D streaming over thin mobile devices, such as PDAs, iPhones, and head-mounted devices (HMDs). However, streaming 3D content over a thin mobile device is considered extremely challenging due to the following issues:

- **Thin mobile capabilities:** Although mobile phone manufacturing has recently evolved dramatically, thin mobile devices still have limited resources in terms of energy lifetime, processing power, storage capacity, and graphics hardware and accelerators. Even high-end mobile phones, such as the iPhone 4 and Palm Pre,

contain PowerVR SGX graphics processing units (GPUs) that can support only an average of 17.5 million polygons per second [20]. Moreover, these GPUs dynamically consume mobile device power, so rendering high quality and realistic 3D scenes on mobile devices remains a bottleneck.

- **Bandwidth limitation:** The wireless network is considered a shared medium, so no stable throughput is guaranteed. In addition to the unstable throughput, many sources of interference arise in a shared medium, which leads to high error rates and packet loss. Implementing a 3D application over a mobile wireless network can also suffer from bandwidth over-utilization, the result of broadcasting every update message to all mobile clients. Also, due to the dynamic nature of the mobile nodes, there is a need to take these fragile links into consideration.
- **Visual quality:** This is negatively affected by many factors, such as network congestion, latency, invalid requests, and long acquisition times. Furthermore, dispersing attempts, which try to enable 3D streaming over thin mobile devices via Image Based Techniques [40], significantly induce visual artifacts (anomalies), such as a lack of depth, which absolutely cancels the main objective of the VE-based application. In addition, although the Area of Interest (AoI) visibility model is the most popular visibility scheme in remote rendering based applications, it creates visual issues, such as object popping discontinuities, that disrupt the user navigation experience.
- **Architecture Issues:** In client-server (C-S) based 3D streaming, the server is the sole content provider; therefore, the applications that adopt the C-S architecture suffer from the server being the bottleneck due to request contentions and being a single point of failure. To address these issues, peer-to-peer (P2P) 3D streaming has been proposed in several studies [4,5,6,11,34]. However, P2P 3D streaming is still

considered immature since object discovery and selection, content provider strategies, and security are all issues that have to be faced.

- **Lack of Scalability:** Most of the proposed approaches that try to enable NVE applications over hand-held devices adopt remote visualization, where the VE is represented using images instead of geometry as a streaming scheme. For remote visualization, the VE content is hosted on the server, the sole entity responsible for rendering the content and sending the resultant images to the mobile users. Although remote visualization meets thin mobile capabilities, it introduces an intensive workload on the server side. In fact, Sterkin [26] reported that a high-end server can only handle 14 simultaneous sessions using this approach.

To the best of our knowledge, none of the proposed research works well. Neither remote rendering nor P2P 3D streaming provides a complete solution to enable realistic 3D streaming over thin mobile devices. Furthermore, very limited literature studies take into account prioritization of objects for transmission. Hence, it is possible for less significant objects to be transmitted before important ones, which leads to inefficient bandwidth and mobile resource consumption.

1.4 Contribution

The main contribution of this work is to propose and implement OCTET, a robust, scalable, hybrid protocol that discovers, selects, prioritizes, and streams 3D mesh models of VEs over a hierarchal, mobile ad-hoc network (MANET). This is done in a real-time manner that enhances realistic behaviours and mitigates the previously discussed issues. The list of contributions in this thesis is as follows:

1. A comprehensive survey about visibility determination algorithms used by both local and remote rendering applications and 3D streaming techniques.

2. A study of state-of-the art 3D streaming frameworks and protocols, identifying the limitations of each from a MANET perspective.
3. A novel integration design of a context-aware module with a distance-based object selection strategy. 3D objects are discovered and selected for streaming based on their proximity and the user's interaction. Mobile resources are set to play an essential role in the selection criteria.
4. Design of a double prioritization visibility algorithm that allows mobile clients to retrieve only the 3D objects based on their final scene contribution.
5. Proposition and implementation of a simple, secure, and immune content provider selection strategy that tackles the limitations of the P2P overlay network.

1.5 Thesis Organization

The rest of this thesis is organized as follows. **Chapter 2** surveys previous and related work in the area of 3D streaming techniques and visibility determination algorithms. Also, selection strategies used in setting the visible range and selecting the suitable content provider in existing 3D streaming protocols and frameworks are highlighted. **Chapter 3** presents a detailed description of our proposed hybrid protocol for enabling 3D streaming over MANETs, which we refer to as OCTET. **Chapter 4** describes the specifics of the performance evaluation methodology and the results that are obtained. Finally, **Chapter 5** concludes this thesis and presents some directions for future research.

Chapter 2

Related work

This chapter surveys and discusses different 3D streaming techniques. The pros and cons of the network architecture used for deploying earlier and current 3D streaming applications are highlighted. Next, state-of-the-art visibility determination algorithms are presented, and finally, there is a discussion of the selection strategies used in the literature for 3D streaming protocols and frameworks.

2.1 3D Streaming Techniques

In general, simplicity and progressive transmission of content are the two dominant drivers in designing and implementing an effective 3D streaming technique so it appears as if content was stored locally. Figure 2.1 illustrates the existing 3D streaming schemes, which may be grouped into four main categories, namely, object streaming, scene streaming, visualization, and image-based streaming [7,8,32,34].

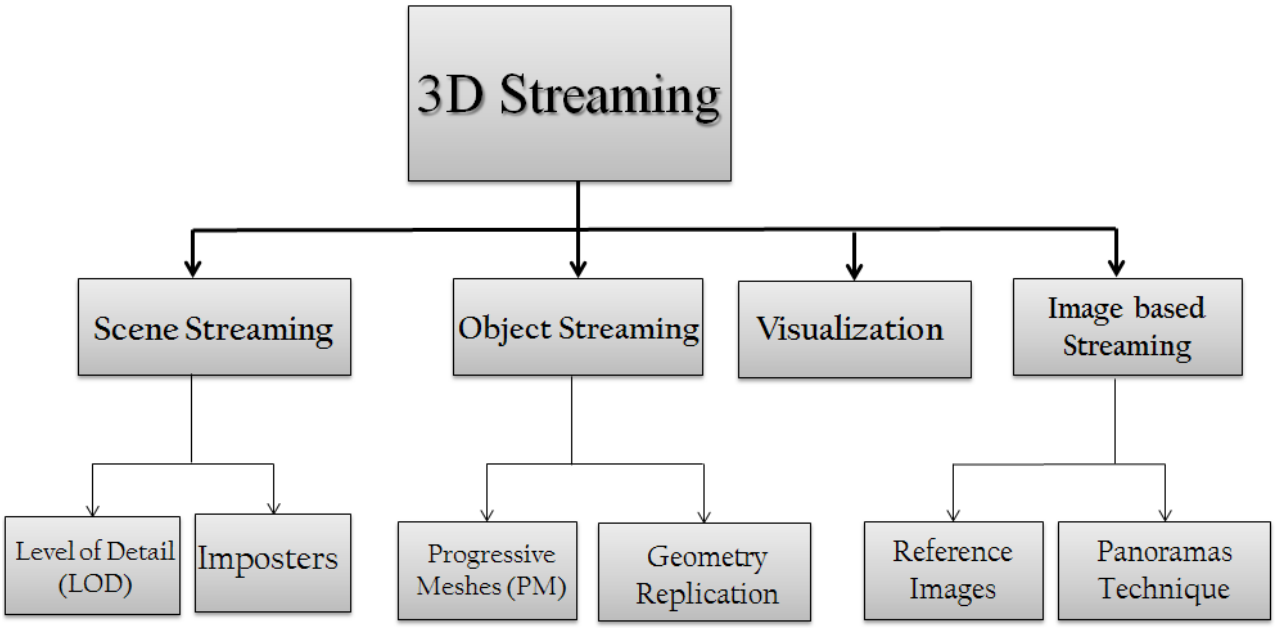

Figure 2.1: 3D Streaming Techniques

2.1.1 Object Streaming

The first type of 3D streaming technique is to stream each object individually. In 1996, Hoppe [7] introduced the concept of Progressive Meshes (PM) in which a 3D object is represented by a base layer at the lowest resolution of the 3D object, and a number of refinement layers are used to progressively improve the resolution of the object until it gets to its intended quality. PM utilizes edge collapse and edge split mechanisms [9] to increase and decrease the resolution of the object model respectively.

As shown in Figure 2.2, a remote user can instantaneously begin interacting with a coarse version of the object once the base mesh is received. Further streaming of additional pieces incrementally restore the exact original 3D object.

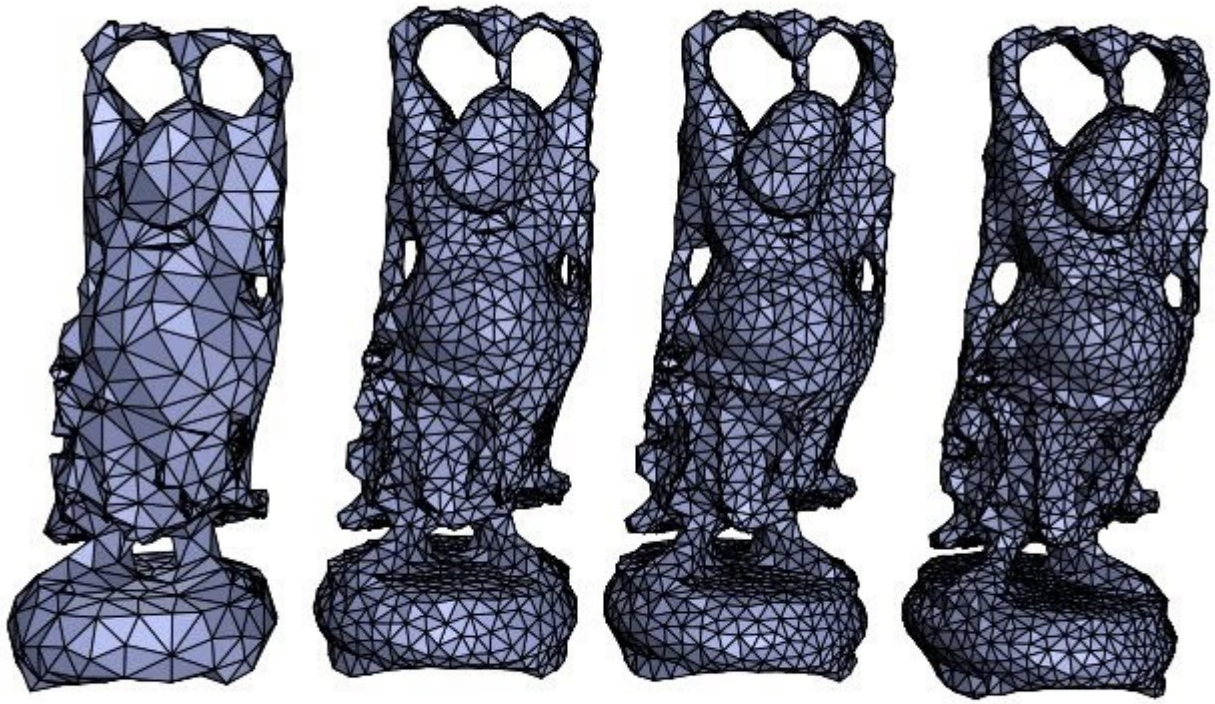

Figure 2.2: Progressive Mesh [35]

As the original PM technique is very basic, further enhancements have been proposed. Kircher *et al.*, [35] enhanced the PM model representation by using an iterative edge contraction to provide a multi-resolution representation for deforming objects with a high quality approximation. Fang *et al.*, [36] used a triangle contraction mechanism to simplify the vertex mapping of mesh models. Additionally, Pajarola and Rossignac [8] proposed the Compressed Progressive Meshes (CPM) method, which proved a 50% reduction in storage and latency over the original PM approach. CPM refines the mesh model via the implant sprays technique, which combines the vertex splits into patches. Finally, H. Li *et al.*, [37] enhanced CPM and applied it in lossy networks by attaching a smart decision module that would cull the most suitable transport protocol for each geometric sub-layer based on the network bandwidth and the loss ratio parameters.

In fact, the progressive mesh technique does not take into consideration the user's viewing angle, so multiple techniques utilize the idea of view-dependent streaming. In this way, the user's viewing angle is considered, giving priority to the currently visible areas and improving visual relevance [15].

Geometry replication [32] is another approach to 3D object streaming and is most popular in online games launched by entertainment consoles like PlayStation 3[22], Xbox 360[23], and Wii[24]. This approach is closer to a pre-installation since the user must have a copy of the 3D model geometry. This may be obtained from the user's hard drive and/or optical drives, and is rendered via local hardware. However, users will frequently download updates from the server, which may include new models, plug-ins, patches, and events, so that they share a consistent view of that scene. Consequently, geometry replication requires a powerful client machine with a robust network connection.

Last but not least, several studies [34,40,49]in the context of specific file formats, such as X3D and MPEG-4's Binary Format for Scene (BIFS), have also addressed 3D object streaming.

2.1.2 Scene Streaming

The scene streaming technique [34,30] transmits entire scenes containing objects from the server to the client instead of individual 3D objects (Figure 2.3). An important feature of scene streaming is that it is used to provide remote navigation and/or multi-user virtual environment experiences. This streaming technique is done in two stages: object determination and object transmission.

The object determination stage[34] is based on the user visibility, location, and interest. The server determines the relevant objects at different levels of detail (LODs) of model geometry to be sent to the client and discards unnecessary objects (i.e., objects out of visibility and interest). In the object transmission stage, data reduction and compression are utilized to decrease the size of the scene before sending. In addition, cached content can be used to minimize the amount of repeated data sent over the network.

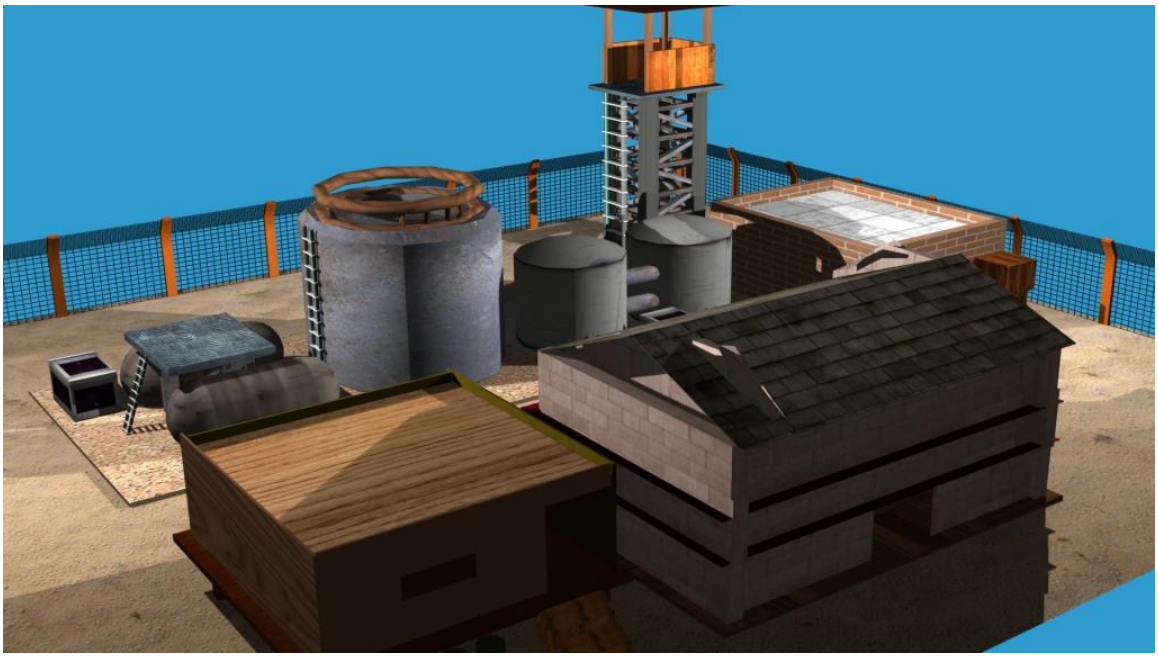

Figure 2.3: Scene streaming [34]

"Pre-rendered imposters" [20,28,32] is a scene streaming method for providing faster initial visualization by transmitting the lowest LOD of 3D objects (Figure 2.4). In this imposter technique, the rendering workload is migrated to another remote node, such as the server. The server renders the complex 3D model on behalf of the user according to the user's viewpoint and location. The user is then responsible for mapping the obtained images to simple shapes and composing them to the self-rendered object to build the final scene. Although this technique shows a significant reduction in rendering time, traffic size, and bandwidth consumption, it also produces visual artifacts, such as lack of depth, due to the transformation process.

Figure 2.4: Imposters [32]

2.1.3 Visualization Streaming

The third category of 3D streaming techniques is used to provide scientific visualizations [34]. An example of visualization streaming results is shown in Figure 2.5. Several scientific projects, such as ViSTA [39] and some Virtual Reality Mark-up Language based projects [40], produce massive amounts of data that needs to be visualized in three dimensions for analysis and comprehension.

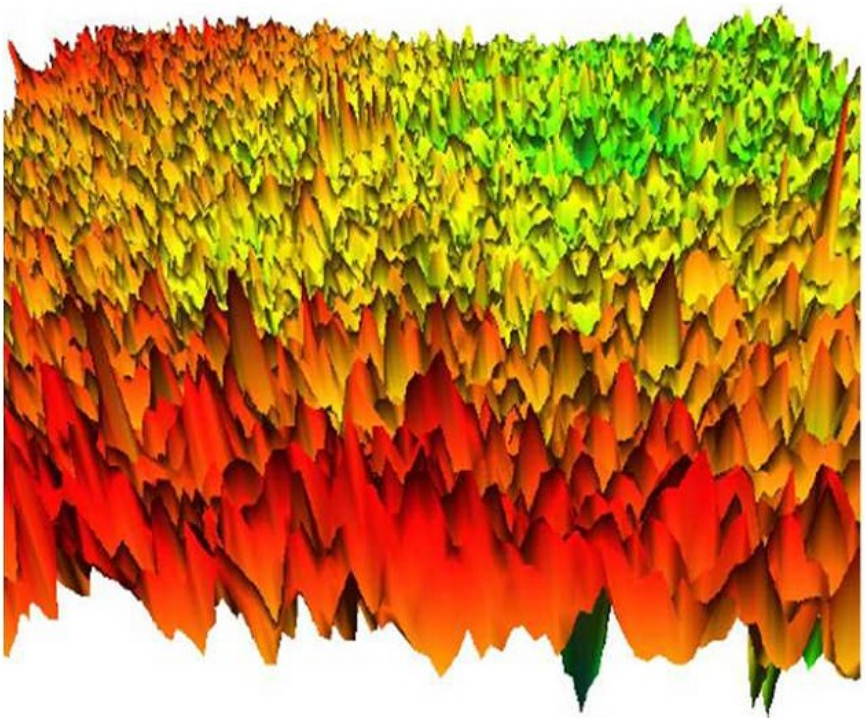

Figure 2.5: Visualization streaming [40]

Such streaming systems differ from the 3D object based streaming primarily in terms of the volume of 3D model data, which is larger and more complex. There are also time-dependent deformation mechanisms to make sure that the 3D dataset is completely refreshed. In addition, the model representation accuracy usually has a higher priority than visual aesthetics. And finally, visualization streaming requires high performance servers and network connections in order to process and transfer the vast amounts of spatial data. This cannot be attained in an internet environment where the bandwidth is often insufficient.

2.1.4 Image-based Streaming

In the image-based model [34,41], 3D content is stored at a high-end server and is not actually sent to the clients. Clients instead receive 2D rendered images generated in real-time by the server (Figure 2.6).

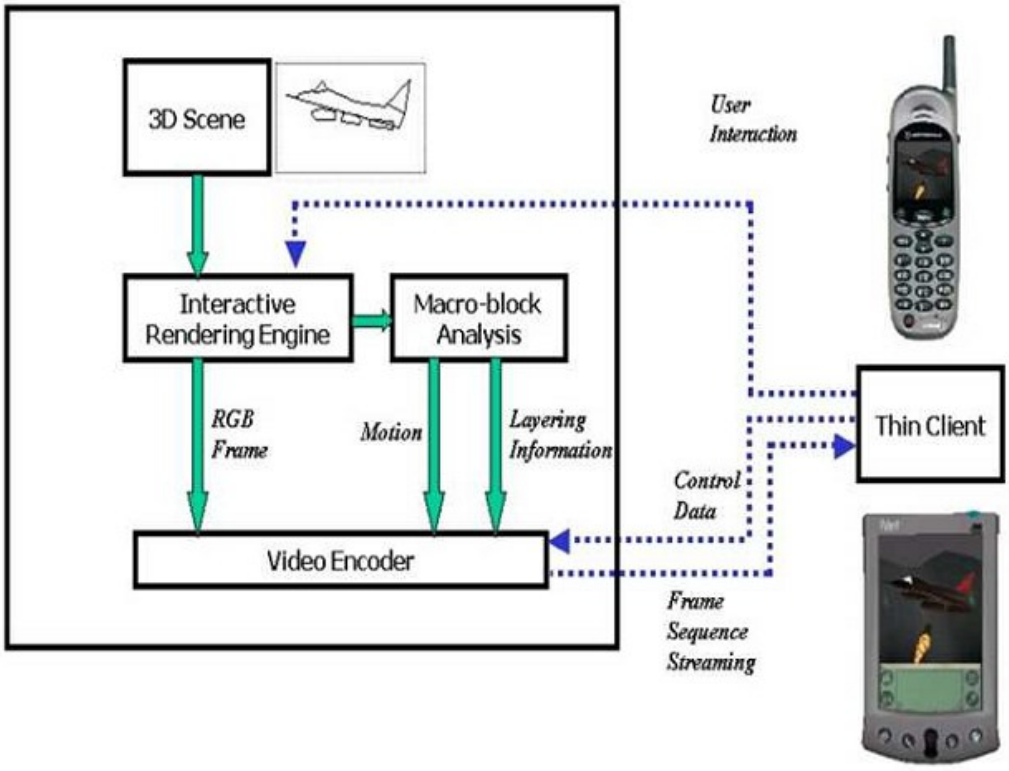

Figure 2.6: IBR [40]

Image-based streaming, also called image-based rendering (IBR), is considered suitable in dealing with thin mobile devices because they have low processing power and no 3D acceleration capability, therefore, it will not overload the client resources. Conversely, the severe processing requirements on the server restrict the system scalability and the interactivity. IBR could be implemented via two approaches: Reference Images [41] and Panoramas technique [3,32]. The former is based on a plenoptic theory, which has reported that vision ability is tightly associated with light rays tracing. The latter involves taking a set of images of a visible scene from all possible directions and combining them in a 3D cylindrical shape [3].

2.2 3D Streaming-based Application Architectures

The previously described 3D streaming frameworks and protocols [2,4,5,6,24,34,43] can be classified into two categories based on their application architecture: Client-Server 3D streaming and P2P 3D streaming.

2.2.1 Client-Server 3D Streaming

Most NVE applications [15,17,22,31,33] adopt the client-server architecture for content delivery. In C-S 3D streaming, also called remote rendering, the server is responsible for hosting the VE, generating new 3D models, and based on the client's orientation, interest, and capabilities, the server will either send a copy of 3D geometry models or thier equivalent pre-rendered images to the clients. The clients are then responsible for rendering and/or displaying the received data locally. Although C-S 3D streaming is widespread and reduces the rendering workload of the client machines, it has several limitations:

1. **Single point of failure:** Since the server is the sole content provider, the entire system will fail if the server becomes unavailable.
2. **Server bottleneck and system latency:** Remote rending is considered both data and processing exhaustive, therefore, huge amounts of server storage, memory, CPU and GPU processing, and bandwidth are needed when serving a large number of users.
3. **Scalability issue:** Current MMOGs are based on interactions among large numbers of users, so the ability of the system to expand to accommodate a growth in users is considered one of the most important requirements. Unfortunately, the server has a limited capacity.

In fact, several research works have been proposed to solve these issues. For instance, Shi *et al.* [42] proposed migrating the 3D content rendering and delivering operations to the

cloud. Despite improved results, the rendering cost increases linearly with the number of users.

2.2.2 Peer-to-Peer 3D streaming

Many successful, scalable, and affordable content sharing systems, such as BitTorrent [16], eMule [21], and Imesh [29], have been deployed in P2P networks. P2P-based 3D streaming has been proposed as an affordable way to address remote rendering issues. The idea is that given users who participate in the same 3D scene, they could possess similar 3D content due to overlapped visibility, so they may obtain content from each other in a fully distributed manner. P2P 3D streaming is promising; however, it has five main issues, namely: object discovery strategy, object selection strategy, provider peer strategy, updatability, and security.

For instance, since the VE is fully distributed among the peers (i.e., there is no single place where the entire VE's content is maintained), and the P2P networks have a dynamic nature, clients will have to follow some strategy in order to determine their visible objects. Also, other questions that must be addressed are how the visible objects will be streamed to the respective parties, who will be considered a suitable peer to provide this content to other clients, as well as how will content be authenticated between peers? These are all issues that need further study.

2.3 Visibility Determination Algorithm

In computer graphics, a visibility algorithm is used to determine the set of 3D objects that are visible from a user's viewpoint. This is one of the primary factors that have been widely studied. For instance, Visual Acuity [13] is the widely acknowledged visibility model in virtual reality systems. It is defined as the ability to recognize a given spatial pattern at a given distance (Figure 2.7).

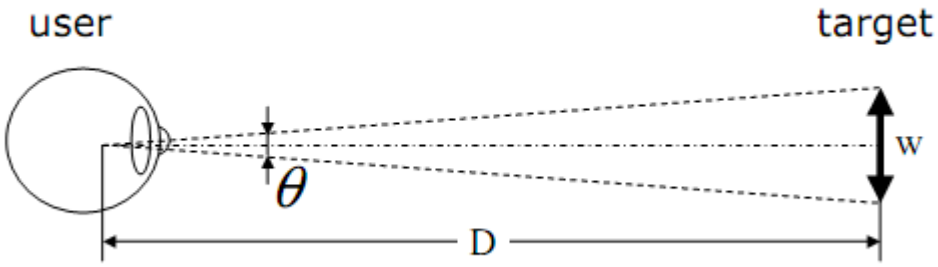

Figure 2.7: Visual Acuity model [13]

In this model, the visible condition between a user and a target object is set based on three parameters: the width of the target (W), the distance between the two entities (D), and the angle subtended at the user's eye by the target (Θ). A normal visual acuity is assumed to differentiate the target object separated by a visual angle of one minute of arc (1MOA). So let Θ_n be the threshold visual angle of the normal eye, D be the standard viewing distance, and D_n the distance at which the minimum target subtends an angle of 1MOA, which can be obtained as such[13]:

$$D_n = \frac{W}{2tan\frac{\theta_n}{2}} \qquad 2.1$$

Therefore, the visual acuity of a user (AC) [13], is equal to

$$AC = \frac{D}{D_n} \qquad 2.2$$

So far, a significant body of work [13,14,15,20] has been dedicated to the implementation of effective visibility techniques. Existing visibility algorithms have been classified based on the nature of the application they are used in. For example, culling techniques are used in local rendering applications and object selection strategies in remote rendering applications. In local rendering applications, the geometric objects are stored

locally on the client's storage medium. Once the objects are retrieved, they are considered for use in the final images. Culling techniques are used to rapidly eliminate visibly irrelevant objects. On the other hand, remote rendering applications are deployed using the client-server model, where the content repository is located at a remote node. Object selection strategies are proposed to reduce the network traffic while obtaining an acceptable visual quality.

In summary, local rendering assumes a priori knowledge on the visible set while remote rendering does not. In the following section, the visibility models in both local and remote rendering applications will be discussed.

2.4 Culling Techniques in Local Rendering Applications

To cull is to "remove from a flock" [1], and in the field of computer graphics this is exactly what culling techniques do. The "flock" is the entire scene that needs to be rendered, and those portions of the scene that are not considered to contribute to the final image are culled (removed). The remaining parts of the scene are transferred to the rendering pipeline for processing. There are many proposed culling techniques, including back-face culling, view frustum culling, and occlusion culling (Figure 2.8). [1]

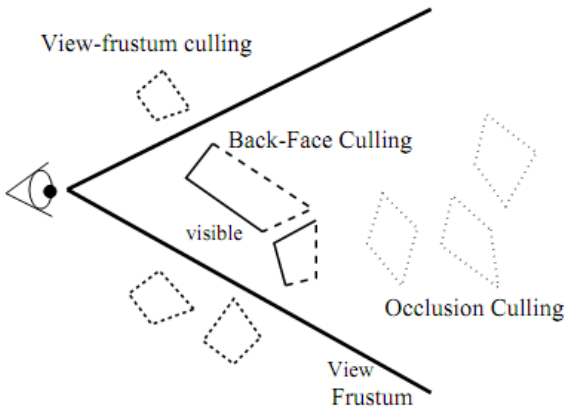

Figure 2.8: Culling techniques [1]

Back-face culling is a simple technique that works with only one polygon at a time and is used to eliminate polygons facing away from the viewer. View frustum culling eliminates groups of polygons that are located outside the view frustum of the user. Occlusion culling is the most complex culling technique because it has to calculate how objects affect each other, thereby removing those objects that are hidden by other objects. After a culling technique is applied in the local rendering pipeline, a Z-buffer algorithm is then usually used to obtain correct images.

2.5 Visibility Algorithm in Remote Rendering Systems

Culling algorithms can only be applied for local rendering applications (i.e., all objects are stored locally on the user machine). When dealing with Networked Virtual Environments, the 3D content is stored on a remote peer, and the client must decide in advance which remote objects should be requested from that node in order to render the given scene. Moreover, the NVE users are allowed to create their own world as they imagine, which may increase the object searching scope. Consequently, new visibility filtration algorithms have to be used.

In the context of NVEs, a user is interested in only a portion of the virtual world called the area-of-interest (AOI). A user's AOI is usually based on geographic proximity, but can also be associated with visibility or user device resources accordingly. The simplest visibility model, called the user AOI model, is based on downloading only those objects that fall inside the AOI of the user. Although the user-centric AOI is popular and used in most MMOGs, it suffers from "object popping discontinuities" [14].

Object popping, shown in Figure 2.9, occurs when the user does not consider rendering a large object located outside its AOI; however, when the user moves toward this object and it is large enough or the transition time Δ is small, the object will appear suddenly in a manner that disrupts the user's navigational experience.

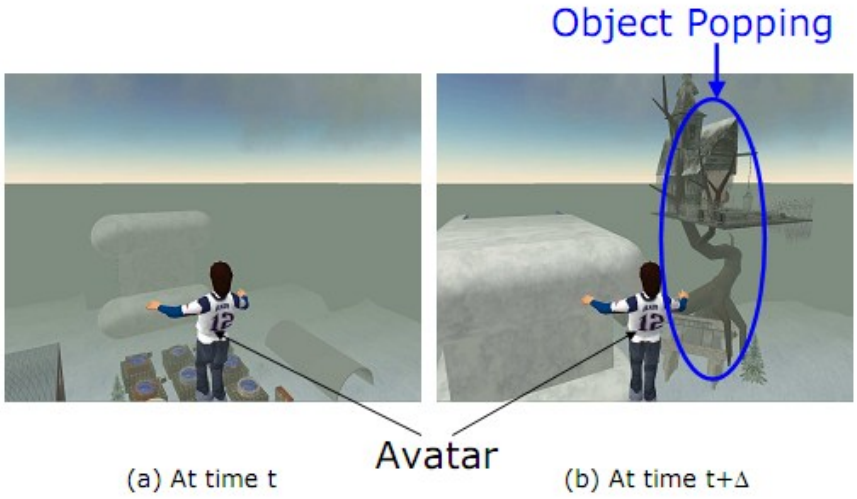

Figure 2.9: Second Life Popping Problem [13]

B. Seo *et al.*, proposed a novel model [14], called the object-initiated view model, in which a spatial indexing is implemented by partitioning the virtual space into equal sized sub-worlds, and each sub-word is further divided into smaller partitions called grid cells. This concept is seen in Figure 2.10.

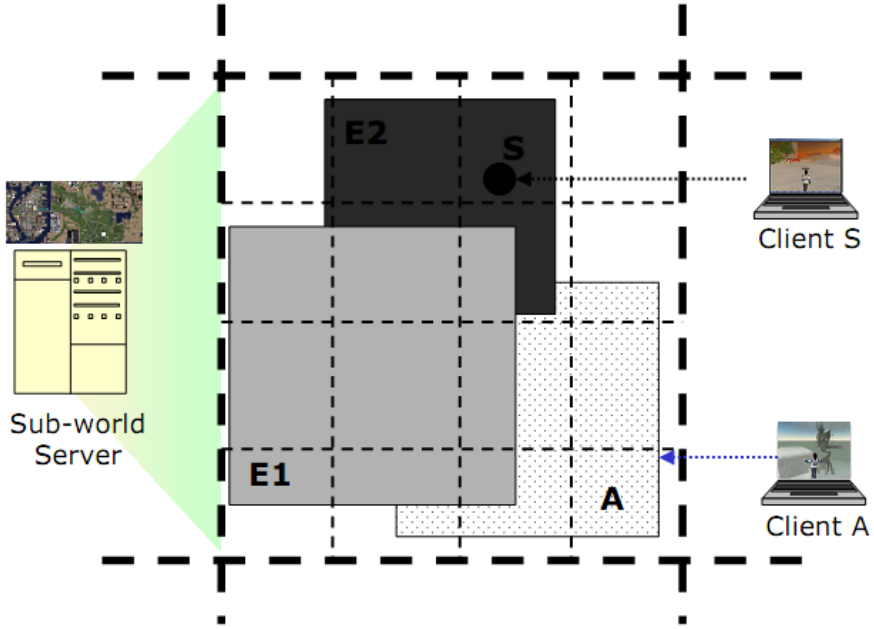

Figure 2.10: Object-initiated view model [14]

Unlike the traditional user AOI model, object entities (objects and other avatars) are assigned a square-shaped AOI. The belief is that the square-shaped spatial extension is very simple and more efficient for indexing in a grid-based virtual world. It was assumed that the user's visual scope is equal to one because every user in the VE possesses equal sight vision. An object is assumed to be visible to a user if the user is inside the AOI of the object. Although this solves the problems of the traditional AOI model, it is considered unfeasible due to its very expensive computational cost, especially when users interact with the VE via actions other than navigation.

Another proposed model is a hybrid model called CyberWalk [9], based on the Aura-Nimbus model [43]. It involves visual scopes stemming from both users and objects. The visual scope of a user is proportional to the user's visual strength, while the object's scope is proportional to its size. An object is computed to be visible to a user if their visible scopes intersect with each other. This hybrid model is imperfect as it is exposed to imprecision issues because these models were originally designed to detect collisions between spatial objects. This was intuitively extended to other spatial relationships, such as interest detection and visibility determination.

2.6 Selection Strategies in Existing 3D Streaming Protocols and Frameworks

In this section we will investigate how the object and provider selection strategies are implemented in 3D streaming frameworks. Table 2.1 summarizes the works proposed in the literature. However, to the best of our knowledge, none of these proposed protocols is intended to provide a fully affordable 3D streaming solution over thin mobile devices. Moreover, none of the literature considers selection and streaming of 3D objects in the user final scene based on both the current action taken by the user and the distance-based model,

while also taking the limitations of thin mobile devices and wireless networks into consideration.

Protocol	Object Selection Technique	Provider Selection Technique	Prioritization Algorithm	Context Aware Module	Progressive Mesh Usage	Ability to be applied on MANET
Hu *et al.*[2,28,34,36]	User AOI Model	Pull and Push based on Voronoi Overlay	X	X	√	X
Botev *et al.* [25]	User AOI Model	Server provided list	X	X	X	X
Royan *et al.* [6,11]	Object Model	TTS and push based on available data	X	X	X	X
Cavagna *et al.* [4,5]	Object Model	Proximity from the super peer	X	X	X	X
J.Chim *et al.* [9]	Hybrid Model	Server provided content	X	X	√	X
S.Benford *et al.* [43]	Hybrid Model	Server provided content	X	X	√	X
H. Rahimi *et al.* [15]	Context based	Server provided content	√	√	X	X
Zhu *et al.* [20]	Imposters	Neighbouri-ng peers	X	X	X	X
Cheng *et al.* [44]	Mesh Hierarchy	Hierarchical P2P Lookup	X	X	√	X
Chien *et al.* [24]	User AOI Model	Bandwidth reservation	√	X	√	X
H.Maamar *et al.* [34]	———	AOI neighbours with mobile available resources	X	X	√	√

Table 2.1: 3D streaming protocols and frameworks.

Hu *et al.* [2,28,34] proposed the Flow Level of Detail (FLoD) solution for exchanging 3D content between peers in an arranged 2D P2P network. In this solution, it is the

responsibility of the user to determine the list of 3D objects that need to be acquired. To do so, upon receiving the scene description from its neighbours, the user determines the list of 3D objects covered by its AOI and acquires the relevant objects in a P2P manner. FLoD improved scalability, however, it is considered inefficient when it comes to dynamic VEs, given that it is difficult to update the scene descriptions. Correspondingly, the content provider selection mechanism, which aims to find a suitable source possessing the relevant 3D object, is based on a naïve, random peer selection approach that gets the content from only the AOI neighbours. This leads to unfair workload and request contentions. Moreover, the P2P overlay in FLoD uses a Voronoi-based Overlay Network (VON) to identify the AOI neighbours, which is completely based on logical relationships without the knowledge of physical networks. To address this issue, the content provider selection strategy was enhanced by proposing multiple improvements to FLOD. For instance, a push-based approach can be used [36] to periodically exchange messages regarding the content availability between peers. The client will select suitable neighbours based on its pushed message and its location in the multi-level AOI.

In another approach, Cavagna *et al.* [4,5,6,11] propose the Level of Detail description tree (LODDT) protocol, designed to support 3D streaming in urban cityscape environments (see Figure 2.11). In this model, the VE is partitioned into specific progressive and hierarchical tree object data structures called PBtrees. Each node in the PBtree has an associated aura that determines the LOD and colours of the object it represents. In order to acquire a given object, the PBtree is traversed in a breadth-first manner using some culling rules. When the objects need to be rendered at a better resolution, child nodes are culled to increase the rendering quality.

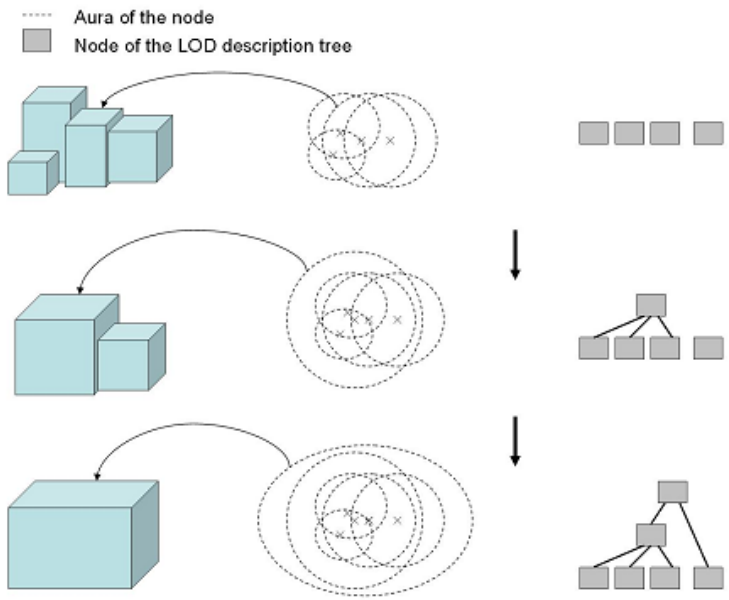

Figure 2.11: LODDT [32]

LODDT uses a similar content provider selection mechanism to FLoD but with more dependency on neighbour proximity to the super peer rather than a VON distributed design. The peers will periodically send their time to serve (TTS) to make the selection of the suitable provider easier. However, LODDT overuses the network bandwidth because redundant information needs to be sent over the network.

Chien *et al.* [24] have proposed Bandwidth-Aware Peer-to-Peer 3D Streaming BAPS, in which the content of the VE is fragmented by Level of Detail (LOD) techniques such that each piece is selected based on its visual weight and its approximation in the virtual word. However, as mentioned before, the LOD exhausts the network with multiple versions of the same data set. In addition to the AOI neighbours, a suitable content provider is selected based on the bandwidth availabilities. The major drawback of this approach is the difficulty in updating the list of potential sources. Keep in mind that the P2P network is dynamic, and the mobile node has full freedom to move within, join, or leave the network at any time.

Botev *et al.* [25] have proposed the HyperVerse protocol, which introduces a solution for providing 3D web content based on federated and torrent concepts. In this work, multiple servers are constructed in a static structure overlay. These servers are responsible for keeping

track of object and user positions in the VE. It is the responsibility of the servers to notify each user of its selected objects and neighbours based on its field of interest. However, we have to take into account the fault-tolerance of the servers so that the content provider selection strategy will stay available if the servers fail.

Cheng *et al.* [44] have worked on deploying progressive meshes while concentrating on the method that a requester can use to determine which vertices are visible before it receives the complete mesh chunk. This method is shown in Figure 2.12. This group supposes that parents of visible vertices are visible, and the parent of a vertex is the one from which the vertex is split. A unique ID is allocated to each vertex split, and they are grouped in a hierarchical way, developing a forest of binary trees.

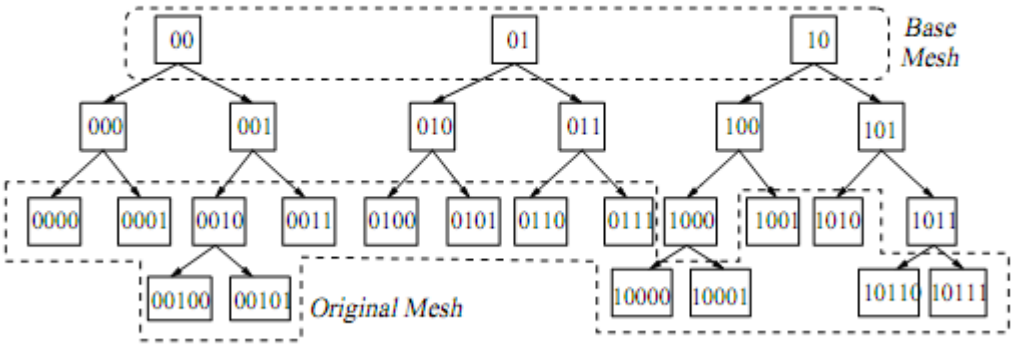

Figure 2.12: Vertex hierarchy [44]

For the content delivery phase, the author proposes a content provider selection strategy based on a hierarchical P2P lookup approach where a set of peers, called a group, is associated with each chunk of the mesh. Each group contains only those peers hosting the same chunk and one super peer, called the leader, is responsible for maintaining the hierarchical layout of the group. Once a chunk is requested by a client, it contacts the group leader who in turn selects a provider from the members of the hierarchical group.

Unfortunately, using a hierarchical P2P lookup increases the system latency and communication overhead.

Zhu *et al.* [20] proposed a peer-assisted rendering (shown in Figure 2.13), where each user has a frustum projected from its position that delimits the objects visible to the user. In this Frustum-Related Vocabulary model, the authors classify objects into three categories: far, static near, and dynamic. These categories are based on the user's position, view direction, and viewing angle. The peer-assisted approach also has two main limitations: visual artifacts due to warping the remote rendered imposters and the homogeneity of the mobile peers, as no one would like to burden its resources with others task.

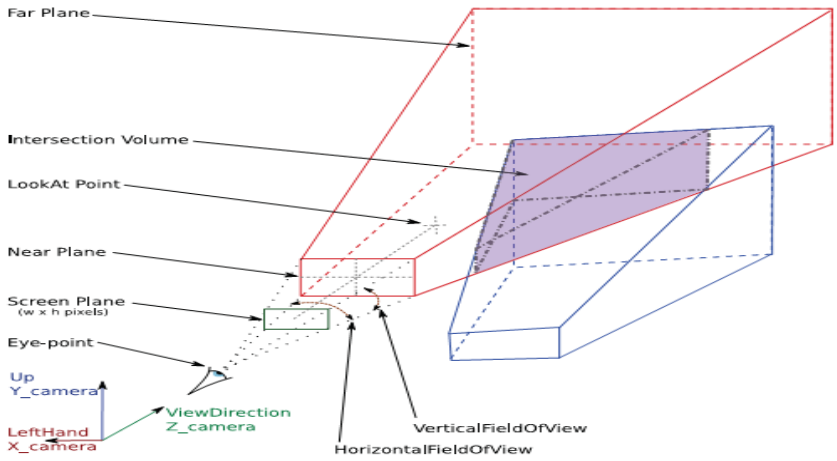

Figure 2.13: The Frustum-Related Vocabulary [20]

Rahimi *et al.* [15] have proposed an activity-centric based technique for selecting 3D objects in MMOGs over mobile devices. This approach uses a context-aware method where the objects to stream are selected based on the activity undertaken by the player. The activity centric approach relies on the game designer in assigning an importance factor for each object in the VE according to all possible activities. However, the authors use LOD as a 3D streaming technique, which is considered inefficient when dealing with thin mobile devices

(due to overhead). Furthermore, it is based on the client-server architecture where scalability is the most serious issue.

H. Maamar *et al.* proposed MOSAIC [34], a suite of supplying partner strategy protocols for P2P 3D streaming over thin mobile devices. In this work, shown in Figure 2.13, the content provider peers are selected based on the source location, the mobile device's available resources, and the mobile device's residual energy.

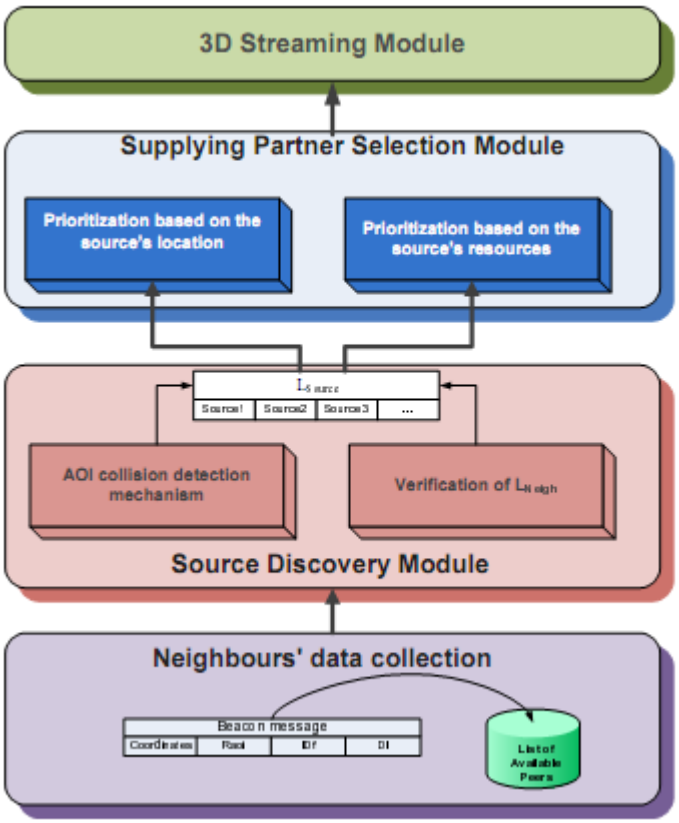

Figure 2.14 MOSAIC Architecture [34]

The author has extended the protocol to select sources based on signal strength and node mobility. Although MOSIAC is one of the preceding P2P protocols used for 3D streaming over thin mobile devices, its main focus is on how the client finds a suitable peer possessing the required 3D content. In other words, the protocol pre-assumes that clients know exactly which 3D objects are relevant to them and concentrates more on getting those

objects in a P2P approach. Therefore, no explicit objects or object discovery strategies are used in this work.

Chapter 3

OCTET: Object and Peer Selection Protocol for 3D Streaming Over Thin Mobile Devices

In this chapter, we describe OCTET, our proposed hybrid 3D streaming model based on thin mobile devices. It comprises of three stages, namely: the area layout acquisition stage, the object selection stage, and the peer selection stage. Before proceeding further into these stages, the system model will be presented.

3.1 System Model

The proposed system is intended for augmented reality applications, where the real world is mapped into a virtual world, and thin mobile devices, such as PDAs, Head Mounted Devices, iPhones, and iPADs are used to navigate the virtual environment. AR environments maintain the locations of objects both in the real world and the virtual one. Therefore, object proximity is maintained (i.e., if objects are proximate in the real world, they are also considered proximate in the virtual world).

Similar to MOSAIC [32], OCTET's VE has been divided into areas, as shown in Figure 3.1, where peers belonging to the same area have similar interests and can exchange relevant 3D data. This zoning technique has been applied in order to minimize the communication overhead. Each area has a region manager (RM) responsible for maintaining relative information, such as the properties of peers and objects. For our purposes, we assume that the 3D objects are static and that users are unable to move them.

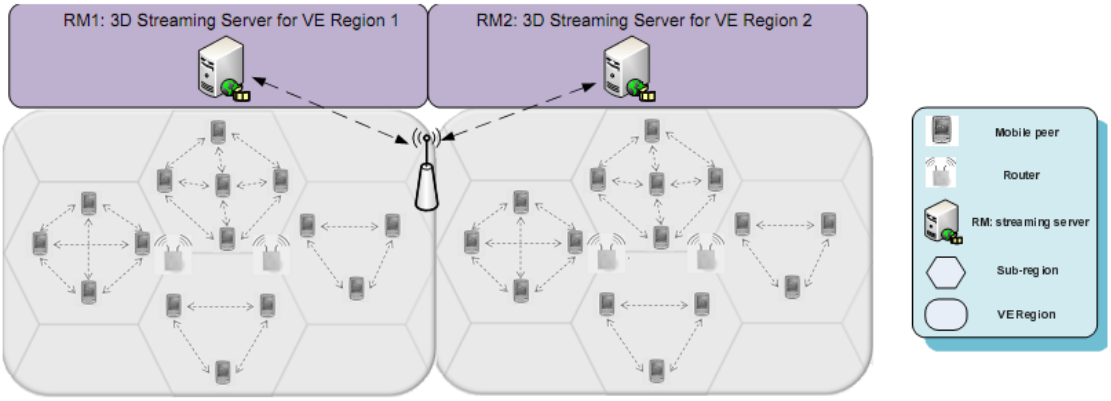

Figure 3.1: OCTET Architecture

Each area has a corresponding area layout LAY_{Area} that contains a list of the 3D objects within it as well as their coordinates and sizes. The sizes of the objects are included in order to eliminate the “object popping" problem that occurs when large objects appear suddenly on the user display. Also, given that the object locations are static, the area layout of a given area is also static and does not need to be updated. Each mobile peer periodically (T_{phase}) broadcasts a hello message to the RM and peers of its corresponding area. This message contains the (X,Y) location of the peer, its current bandwith BW, speed V, and if avaiable, the IDs of 3D objects and/or pieces found in its memory buffer. By doing so, the RM will build a list of the content providers, known as the LSP, found in its area. In order to eliminate the communication overhead, bandwidth consumption, and RM false source information, the broadcast domain of any mobile peer P is limited to its corrsponding area. If

the RM doesn't receive any hello message within the next broadcast period, the RM will remove P from the LSP.

Our proposed system is based on a hybrid architecture, where both the client-server and the P2P overlay networks are used. The area layout and the list of content providers are acquired using a centralized approach, where the RM is the entity responsible for sending the area layout and the content provider selection of a given area to a user joining that area. As for object acquisition, it is realized using a distributed approach.

3.2 Object Selection Protocol

Our proposed object selection protocol aims to determine the objects that are visible for a given user and that need to be streamed. We refer to our protocol as OCTET. Since we are dealing with thin mobile devices with limited capabilities, having the entire VE streamed to the user is not effective and may overuse the mobile device resources. There is, therefore, a need to design a protocol capable of determining specific objects that need to be streamed to the user. Our protocol is composed of two phases: the area layout acquisition phase and the object selection strategy phase.

3.2.1 Area Layout Acquisition Phase

This phase only applies when a new user joins an area. In this case, the user sends a beacon message to the area's corresponding RM, providing the user's coordinates and the type of interactions desired. It is assumed that there are a number of predefined actions that can be performed in a given area. Upon receiving the beacon message, the RM sends the corresponding area LAY_{Area} as well as a list of surrounding objects of interest L_{objInt}. As mentioned earlier, the area layout contains a list of all of the 3D objects contained in the area along with their coordinates. As for L_{objInt}, it represents the objects that the user can interact with knowing its type of interactions.

3.2.2 *Object Selection Strategy Phase*

In the OCTET protocol, the users are mobile and their views need to be updated. Therefore, the object selection strategy phase is applied as long as the user is moving. In order to select the 3D objects that need to be streamed, we propose our multi-level AOI object selection technique. The multi-level AOI has already been applied by Sung *et al.* in order to select the supplying partner in FLoD [36]. In our work, a similar approach for 3D object selection is applied. The idea behind the multi-level AOI is to subdivide the user's AOI into different levels (circles) that have a common centre (the user). Only the objects that are part of the multi-level AOI need to be streamed and are added to the set of nearby objects L_{AOIobj}. The selected 3D objects have to be part of the list of surrounding objects of interest L_{objInt}.

Given that the mobile device has limited capabilities in terms of bandwidth, energy, and memory, streaming all of the 3D objects that are part of the user's AOI is the least efficient strategy. As a matter of fact, the user's AOI has a circular shape, so some of the 3D objects may be situated behind the user and there is no need to stream these objects, especially if the user has limited capabilities. In order to limit the set of added to L_{AOIobj}, only the objects that are situated in the user's field of view shoud be considered. In our design, we represent the user's field of view by a virtual "headlight" that is maintained along the movement direction of the user. Therefore, only the objects that are part of the overlapping area between the headlight and the multi-level AOI are taken into consideration and added to the set of visible objects L_{objVis}.

Several works set the human field of view to be 120 degrees [46,50], so our proposed protocol's headlight angle is set to 120 degrees. The virtual headlight is divided into three zones: the front zone, which illustrates the objects located in the line of sight of the user, and the two border zones, which represent the 3D objects located at the borders of the headlight area. Figure 3.2 illustrates the overlapping area between the headlight zones and the multi-

level AOI. Unlike traditional visibility models, which set the radius of the visibility range based on VE's proximation, we set the radius of the multi-level AOI based on the mobile device's resources.

$$R_{multi-AOI} = \alpha ResidualEnergy + \beta Memory + \gamma bandwidth \quad 3.1$$

$$where\ \alpha, \beta, \gamma < 0$$

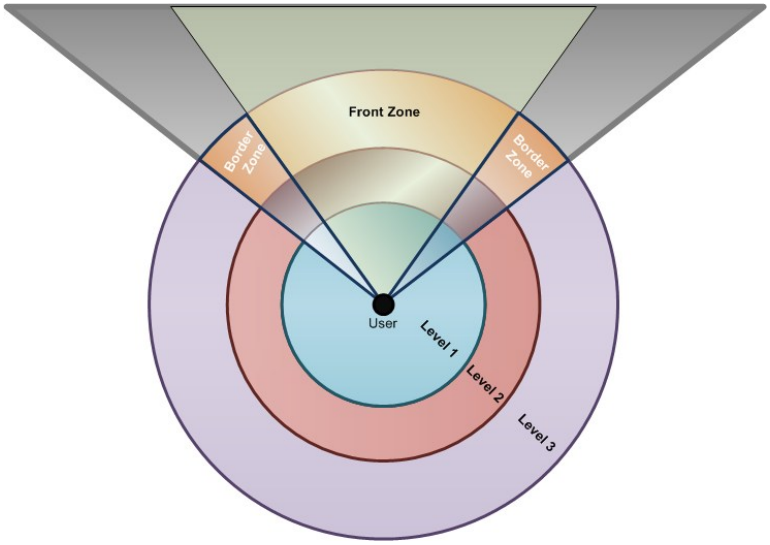

Figure 3.2: Headlight and multi-level AOI overlapping

Upon computing the set of visible objects L_{objVis}, 3D objects are prioritized based on which level of AOI they belong to and the headlight zone they are located in. We refer to the priority assigned to the 3D object O_i as *Priority*(O_i). For instance, 3D objects that lie in the first level of the multi-level AOI and in the front zone of the headlight, receive the highest priority and *Priority*(O_i) = 1. As for the 3D objects that are part of the first level of the AOI, but are located in the border zones, they receive the second highest priority, or *Priority*(O_i) = 2. In other words, the priority assigned to a 3D object decreases as the headlight zone changes towards the border zone and as the AOI level increases (i.e., as we move away from the centre). *Priority*(O_i) determines the resolution at which the 3D object O_i should be streamed. For example, if the 3D object lies in the first level and in the front zone, it has the highest

priority and has thus to be streamed to the user at a higher resolution than another 3D object that lies in a different level.

Our design, also uses the progressive mesh 3D streaming technique. A given 3D object is, therefore, represented by a base layer that is at the lowest resolution of the 3D object. A number of refinement layers are used to progressively refine the resolution of the object until it gets to its highest quality. Given that the streaming and rendering actions overuse the mobile device resources, we propose that the number of refinement layers for a given 3D object be proportional to the mobile device residual energy, and available memory. As such, a given 3D object will be rendered at its highest quality only if the mobile device has high resources. Conversely, when the mobile device resources are low, the number of refinement layers will also be low in order to not further overuse the mobile device resources.

We propose to prioritize the base mesh and refinement layers based on the headlight zone and the AOI level that the 3D object belongs to. The base mesh of a given 3D object receives a priority equal to *Priority*(O_i), and the refinement layers of the same 3D object receive a priority equal to *Priority*(O_i) + 1. For instance, if the objects O1 and O2 belong to the AOI-Level1 and AOI-Level2 respectively, and they are both located in the front zone of the headlight, the base mesh of O1 receives a priority of 1, while the base mesh of O2 receives a priority of 3. In the same example, the refinement layers of O1 receives a priority of 2, and the refinement layers of O2 receive a priority of 4. This prioritization was applied in order to schedule the delivery of the base meshes of 3D objects based on their importance in the field of view of the user. The refinement layers remain important to refining the quality and the resolution of the 3D objects, however, they receive a lower priority when compared to the base mesh. Algorithm 1 illustrates the logic behind the OCTET object selection protocol.

Algorithim 1 - Object Selection Protocol

Require: LAY_{Area}, L_{objInt}
Compute Multi-level AOI
for Each object O_i in L_{objInt} **do**
if O_i in Multi-level AOI **then**
    Add O_i to L_{AOIobj}
endif
endfor
$Area_{Overlap}$ ← Overlap(Multi-levelAOI, Headlight)
for Each object O_i in L_{AOIobj} **do**
if O_i IN $Area_{Overlap}$ **then**
    Add O_i to L_{objVis}
    LevelAOI (O_i) ← LevelAOI(R_{AOI},X_{Oi},Y_{Oi})
    if O_i IN $FrontZone_{Headlight}$ **then**
        if LevelAOI (O_i) == 1 **then**
            Priority (O_i) ← LevelAOI (O_i)
        else
            Priority (O_i) ← LevelAOI (O_i) + 1
        endif
else
        if LevelAOI(R_{AOI},X_{Oi},Y_{Oi}) == 1 **then**
            Priority (O_i) ← LevelAOI (O_i) + 1
        else
            Priority (O_i) ← LevelAOI (O_i) + 2
        endif
    end if
endif
end for
NumRefLay ←α * Resource (ResEnergy,memory)
for Each object O_i in L_{objVis} **do**
P_{res}(O_i(BaseMesh)) ←Priority (O_i)
  for Each refinement Layer (O_i) **do**
    P_{res}(O_i(RefLay)) ← P_{res}(O_i(BaseMesh)) + 1
end for
end for

3.3 Source Selection Strategy

In this section, we explain how a new mobile user can get the 3D content of interest from a suitable provider by invoking a source selection algorithm. Our proposed algorithm is efficient and selects the potential sources that have the relevant 3D data based on a set of criteria that take into consideration the limitations of MANET networks.

Once the mobile user has its prioritized list of visible objects L_{objVis}, it needs to get the 3D content for that list from the suitable sources. To do so, the user must send a request message to the RM asking it to provide a list of source providers. At the initial phase, all peers are assumed to get their 3D data from the RM, which adds a record for each requestor

in the LSP containing the identification of the requester peer (ID), its available bandwidth (BW), its current velocity (S), the time when the request was received (TR), and the IDs of the requested objects. When a new joining client requests some objects that are already send to previous requesters, the RM replies by sending an LSP prioritized by the latest TR, BW, and S of the source peer. We believe that a source peer with low speed will have a high probability of staying in the area with little modification in its view, and would receive a high priority. After that, based on the priority assigned in the objects selection phase, the client sends data request messages to obtain the content of all objects in the L_{objVis}.

In order to avoid having a single point of failure, which occurs when the tracking tasks become unavailable when the RMs fail, each client, based on a list of neighbouring peers $\boldsymbol{L_{neighbours}}$, which can be built via periodic broadcasts of hello messages. The clients then build another list of source peers based on the viewpoint similarity (VS). The list is called the LVS. The VS depends on the headlight intersection and the field of view angles of both the client and the source. As a matter of fact, the VS gives the percentage of the objects located inside the visual scopes of both client and content source peer.

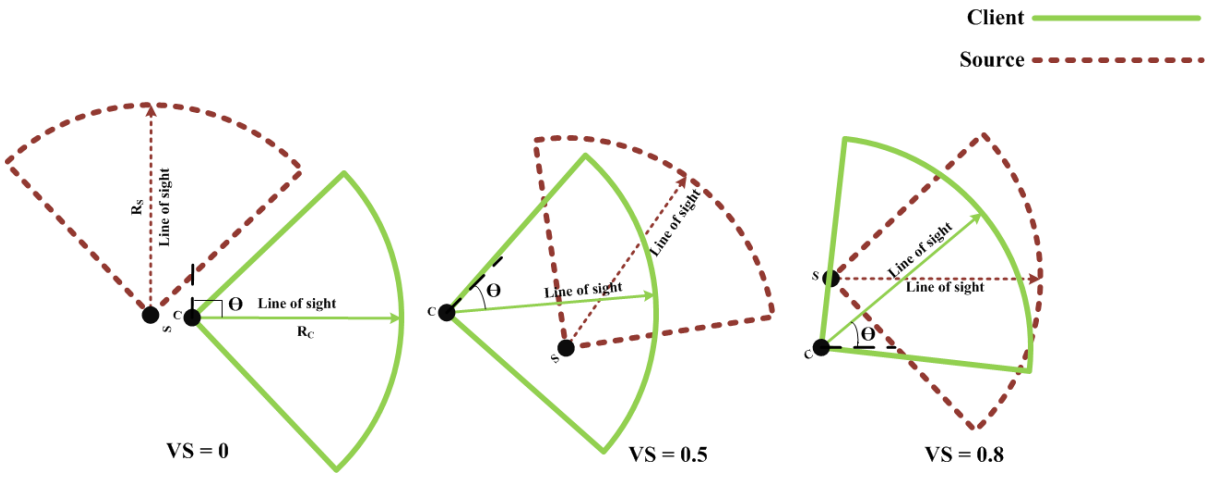

Figure 3.3: Viewpoint Similarity

For example, the viewpoint similarty, if the difference in the viewing angle is between 0^o and 89^o, could be computed as such:

$$VS = \left| \frac{\frac{R_S + R_C}{2} - \sqrt{(X_C - X_S)^2 + (Y_C - Y_S)^2}}{(\frac{R_S + R_C}{2})} * cos(\Theta) \right| \quad 3.2$$

Where:

R_c: is the multi-level AOI radius of the client
R_s: is the AOI radius of the source peer
Θ: is the difference in viewing angle between client and source.

Accordingly, if the RM becomes unavailable, the content of the selected 3D object will be obtained from those peers with a VS value greater than a predefined threshold. It is worth mentioning that the RM acts as a gateway of last resort to deliver the 3D content of interest when the client is unable to find a suitable peer in the network. The selection strategy is summerized in Algorithm 2.

Algorithim 2 - Source Selection Strategy

Require: L_{objVis}, $L_{neighbours}$(Client)
1: Request RM
2: **if** RM respond **then**
3: **for each** P_i In LSP **do**
4: Priority (P_i) ← Min(TR,S) + Max(BW)
5: **end for**
6: **else**
7: **for each** P_i in , $L_{neighbours}$ **do**
8: Compute **VS**
9: **if VS <= VS_{min} then**
10: Add P_i to LVS
11: **end if**
12: **end for**
13: **end if**
14: **for** Each object O_i in L_{objVis} **do**
15: Request O_i(BaseMesh) with Priority (O_i) from $P_i \in \{ LSP \cup LVS \}$
16: Request (O_i(RefLay) with P_{res}(O_i(RefLay)) from $P_i \in \{ LSP \cup LVS \}$
17: **if** client still needs content **then**
18: Send request to RM
19: **end if**
20: **end for**

3.4 Proof of Concept

In this section, we shall present the proof of concept for the proposed OCTET protocol, whose outcomes are fourfold:

1. It intends to quickly discover the relevant 3D objects required for the scene.

2. It takes into consideration the mobile device and network limitations.

3. It adapts the visibility of the user based on their interests.

4. It aims at optimizing the visibility of the users.

Regarding the **fast discovery** of relevant 3D objects, our approach is based on the area layout LAY_{Area} and the list of objects of interest L_{objInt} that are delivered to a newly joining user based on his/her location and activity interests. Given that the 3D objects are static, and based on their current location, the user is able to fast discover the relevant 3D objects and to prioritize them based on their locations in the overlapping area between the headlight and the multi-level AOI.

For the network and mobile device limitations, our proposed approach applies several different techniques. First of all, given that the mobile device has limited capabilities in terms of bandwidth, energy, and memory, OCTET does not select all of the 3D objects located in the AOI. This is because the user's AOI has a circular shape and some of the 3D objects may be situated behind the user. There is, therefore, no need to stream these hidden objects, especially if the user hardware has limited capabilities. As opposed to other research works [2], OCTET only selects the 3D objects located in the overlapping area between the virtual headlight and the AOI. This way, the list of 3D objects that need to be acquired is further shortened, and the required rendering is minimized. Secondly, the progressive mesh 3D streaming technique is applied and is composed of a base layer that represents the lowest resolution of the 3D object and refinement layers that are added to refine the resolution of the object as resources permit. Several works in the literature [4,5,6,11] use the LOD technique, which consists of representing the 3D object at different resolutions. However, LOD overuses the network bandwidth, given that redundant information needs to be sent over the network. Moreover, the base mesh and refinement layers are considered small compared to the LOD representations. Therefore, the applied 3D streaming technique does not overuse the mobile

device resources nor the network bandwidth. Thirdly, our approach takes into account the mobile device's available resources during the representation of the 3D object. Given that streaming and rendering actions overuse the mobile device resources, the number of refinement layers for a given 3D object are set to be proportional to the mobile device resources. Therefore, if the mobile device has low resources, OCTET does not further overuse them, but rather adapts the resolution of the 3D object to the available mobile device resources.

Regarding the visibility adaptation, the first step of OCTET is to determine L_{objInt} based on the user's intended interactions. In other research works [2,4,5,6,25,33], all of the 3D objects are streamed to the user based on their proximity, which overuses the mobile device resources. Moreover, the user would not need to interact with all of the objects within the VE. Based on these observations, and as opposed to these previous research works, OCTET adapts the visibility of the user so that it only streams the 3D objects of interest. In this way, the user's visibility is simplified and adapted to the user's interactions, and the mobile device resource usage is minimized.

As for the visibility enhancement, OCTET applies double prioritization for the 3D objects to be streamed. The first prioritization aims at classifying the 3D objects based on their locations, whether they are in the front zone of the headlight or the border zones, as well as the level of the AOI they belong to. The second prioritization aims at classifying the base mesh and refinement layers of a given 3D object based on the priority it received in the first step. This way, the most relevant 3D objects are received first and the user can start interacting with the scene while waiting for the less important objects. Figure 3.3 illustrates the sequence diagram of the OCTET protocol.

One of the main questions that may arise is why there has not been a proposal or implementation of a full P2P approach. According to a recent study [27], the 3D models in

the "Second Life" MMOG make up about 75-80% of the traffic that is communicated between the server and clients. One of the main targets in our research is to maximize system scalability, and this can be achieved by reducing the workload required at the server side by obtaining 3D models from peer clients in a P2P manner instead. Although the region master is used to get both the area layout and the list of content providers, the size of these lists are considered negligible (a few bytes) when compared to the actual 3D content, so it will not affect the scalability of our system. Furthermore, the use of a centralized list is always more secure and immune to tampering by malicious users as well as more flexible during updates.

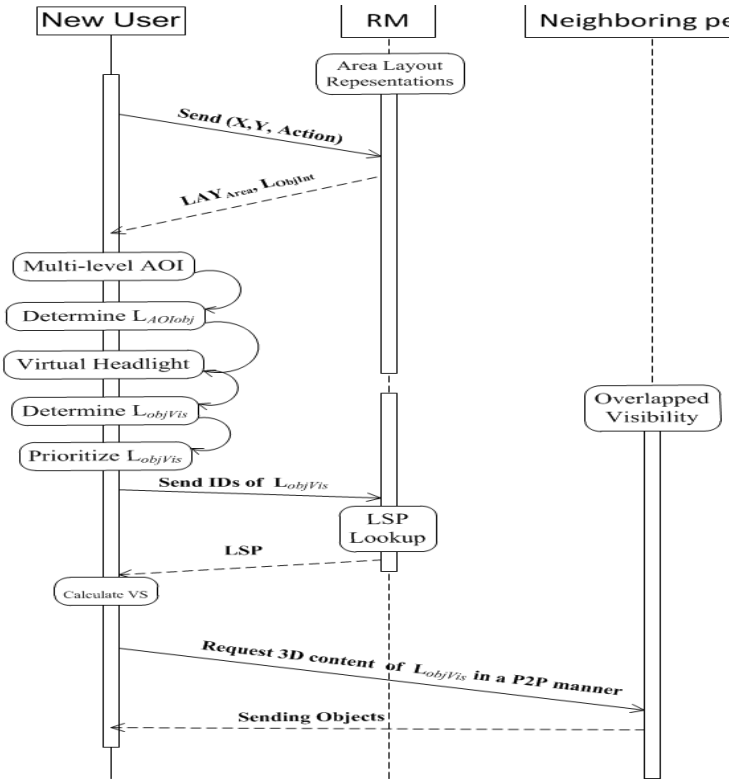

Figure 3.4: OCTET Sequence Diagram

3.5 Summary

In this chapter, we presented OCTET, our proposed hybrid protocol for 3D streaming based systems over thin mobile devices. OCTET aims to select the 3D objects of interest for the user's scene by taking into account the user's interests, the available resources on the mobile

device, and the importance of the object in the user's field of view. As shown in the proof of concept, we believe that OCTET intends to quickly discover the relevant 3D objects required for the scene using the layout area as well as the set of the objects of interest. Moreover, our proposed protocol takes into consideration the mobile device and network limitations by applying PM as a technique for 3D streaming, given that it is considered the least demanding on network bandwidth and mobile device resources. Furthermore, OCTET applies an efficient selection technique that is based on the overlap of a virtual headlight and a multi-level AOI. And finally, a content provider approach is a light and rapid approach to discover the suitable peers who host the content of interest.

Chapter 4

Performance Evaluation & Experiment Results

Network simulator 2 (ns-2) [12] was used as a discrete-event simulator to evaluate our work. Table 4.1 represents the specific simulation parameters. The virtual environment is represented by uniformly placing a number of objects in a 2D space. Each object constitutes one base piece with a size of 3KB and four refinement pieces with a size of 1KB per refinement. Therefore, the size of each object is set between 3KB to 7KB.

VE dimension	400x400m
Area size	100x100m
Object size	3-7 KB
Object Number	500
Base mesh size	3 KB
Refinement layer size	1 KB
AOI radius	10.4 m (max)
Number of nodes	25-100
Node speed	1-20 m/s
Server Upload Rate	2 Mb/s
Mac Protocol	IEEE 802.11
Simulation Time	200-1000 sec
Antenna Type	Omni Antenna

Table 4.1: Simulation Parameters

During the simulation, different numbers of mobile peers were created (10, 25, 50, 75, and 100) and randomly positioned in the VE. At the beginning of the simulation, each object peer stays at its initial position until all visible objects, according to its selection strategy, are downloaded. By doing so, we make sure that the region master of each area builds a list of content providers. In this way, the 3D contents can be acquired in a P2P manner. In client-server 3D streaming experiments this step is excluded. The mobile peers then start moving at a speed varying from 1 to 20 m/s and applying OCTET object and peer selection strategies, as long as their viewpoints change.

In order to measure the efficiency of our protocol, a comprehensive set of simulation experiments using ns-2 environments has been executed. Each simulation has been repeated several times to ensure accuracy, credibility, and reliability of our measurements. Initially, we consider a communication scenario where OCTET is compared to a non-zoning protocol, and then a comparison is carried out when we vary the time interval, T_{phase}, to periodically exchange the peer status hello messages. Then, we came up with several different scenarios. The first one adapts our OCTET object selection strategy, whereas the second one uses the AOI visibility model deployed by most remote rendering based NVE applications. Both scenarios use the same source selection strategy. An extensive set of experiments is used to compare the performance of both object selection strategies. The two strategies were compared in terms of object selection ratio, residual energy, mobile client download rate, and average response time. We also consider scalability scenarios where another set of experiments is used to evaluate our OCTET's source selection strategy when compared to client-server 3D streaming. The two streaming schemes are compared in terms of different network metrics, which are server utilization, end to end delay, and control overhead.

4.1 Communication Overhead

The first set of experiments is used to study the effect of the periodic time interval T_{phase} on the performance of our protocol. As shown in Figure 4.1, varying the value of the T_{phase} will significantly affect the number of broadcasted control messages.

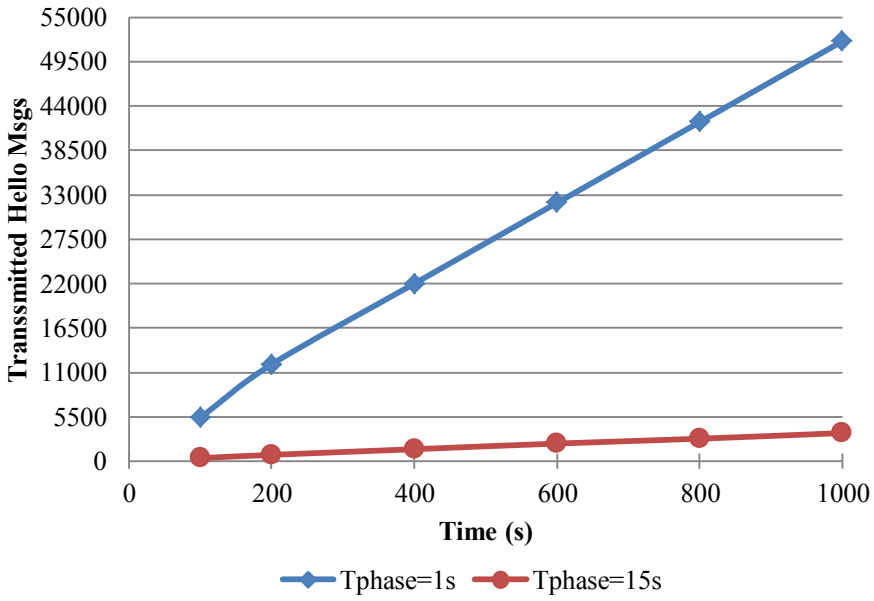

Figure 4.1: Number of Transmitted Hello Messages VS Time for different T_{phase}

The data shows that smaller values of T_{phase} produce a massive number of hello messages that will cause congestion in the network. In addition to decreasing the throughput, it also increased the energy consumption by making the mobile device send and receive messages every second. This can be clearly seen in Figure 4.2.

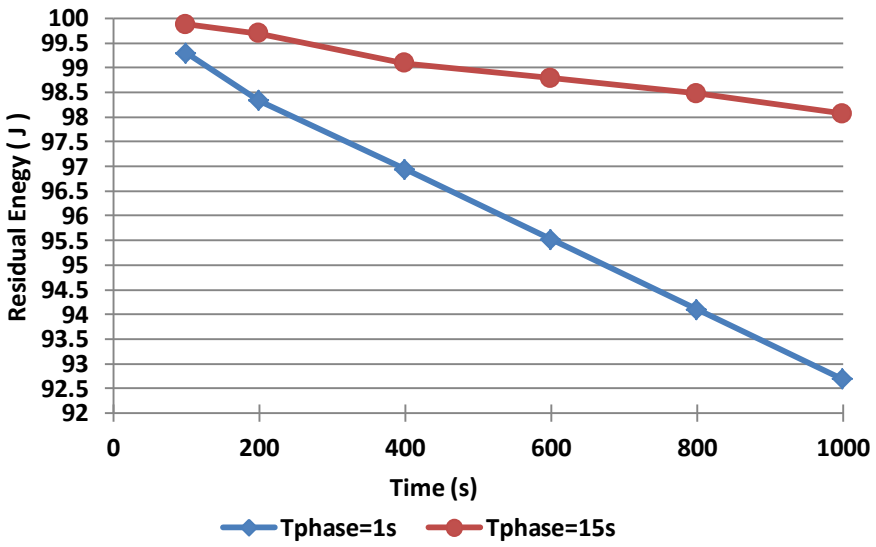

Figure 4.2: Residual Energy VS Time for different T_{phase}

On the other hand, although larger T_{phase} values reduced the number of control messages and resource utilization, it may decrease the reliability and the updatability of the system by issuing invalid requests that can adversely affect the streaming quality and the system latency. In order to tackle these issues, we set the T_{phase} value to dynamically change based on the density of the mobile nodes per area and their speeds.

4.2 Computational Overhead

The computational overhead can be expressed as the number of exchanged state hello messages received by mobile clients. We compare OCTET to the non-zoning protocol, where each user will receive and treat broadcasted messages from the entire user base in the VE. As mentioned before, the entire VE is divided into multiple broadcast domains. No broadcast message will be exchanged between peers from two different areas. As can be seen in Figure 4.3, OCTET causes a clear reduction in the computational overhead, tackling one of the MANET challenges.

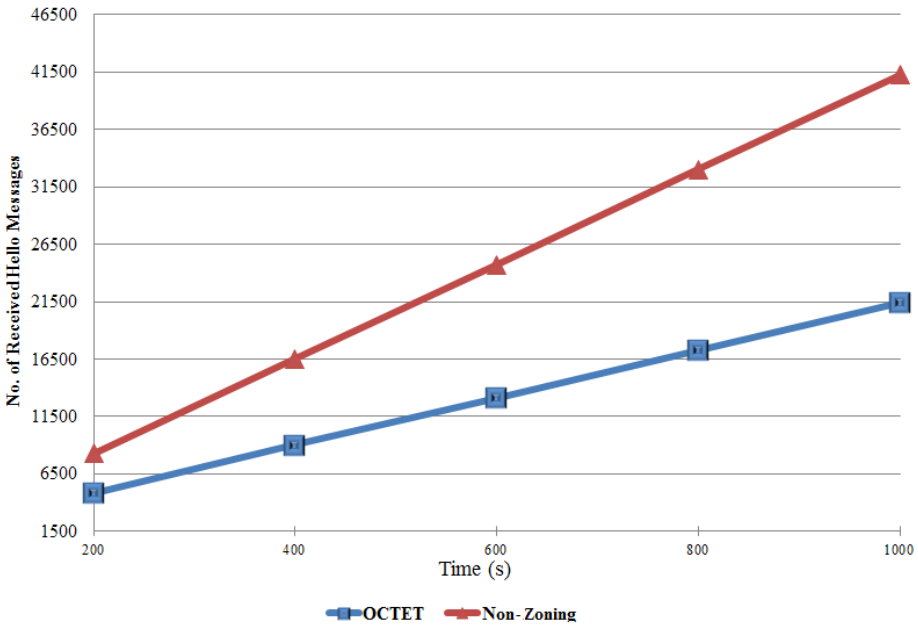

Figure 4.3: No. of Received Hello Messages VS Time

4.3 Object Selection Ratio

Figure 4.4 shows the mobile client's average number of selected objects when running OCTET versus the user AOI model as an object selection strategy. The data demonstrates that the average number of selected objects by the clients using the OCTET model varies according to time and is always less than the average number selected by the AOI model.

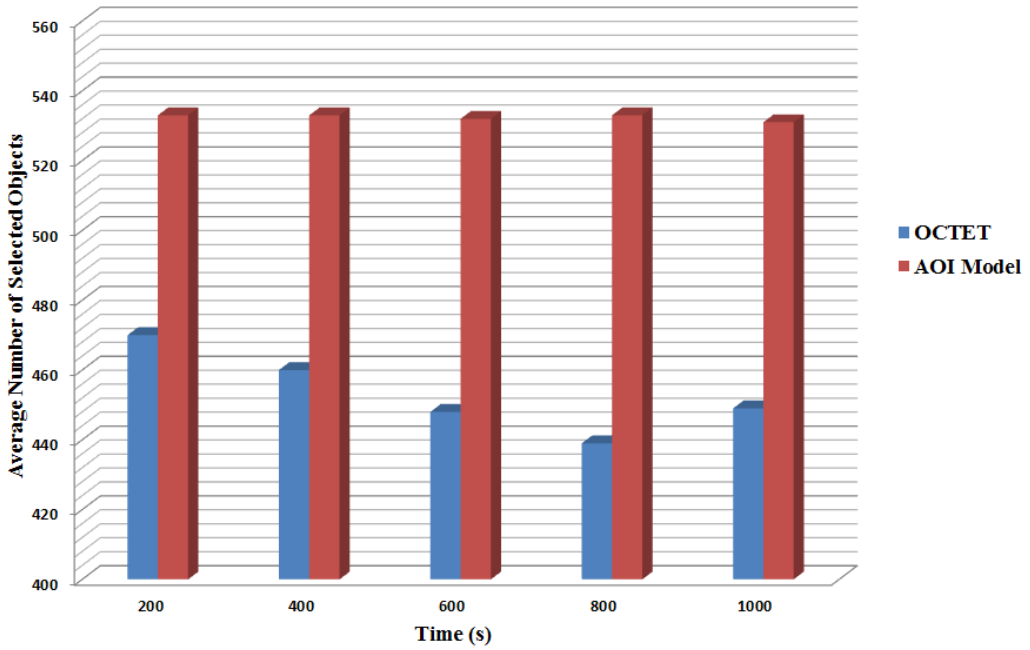

Figure 4.4: Average Number of Selected Objects VS Time

This difference is due to the fact that OCTET shrinks the set of relevant 3D objects according to

a) their contributions to the current activity being performed by the mobile user

b) their locations with respect to the intersection between the virtual headlight and the multi-level AOI.

Moreover, the radius of the multi-level AOI is set to be proportional to the mobile user's resources so that the value of the visibility range changes dynamically while there is mobile resource consumption. On the other hand, the AOI visibility model considers only one navigational action and needlessly also selects the objects behind the user.

4.4 Residual Energy

This metric is a good indicator about how both streaming strategies affect mobile resources. We assigned the initial energy of each peer to 100J, and as illustrated in Figure 4.5, we measured the residual energy over a period of time.

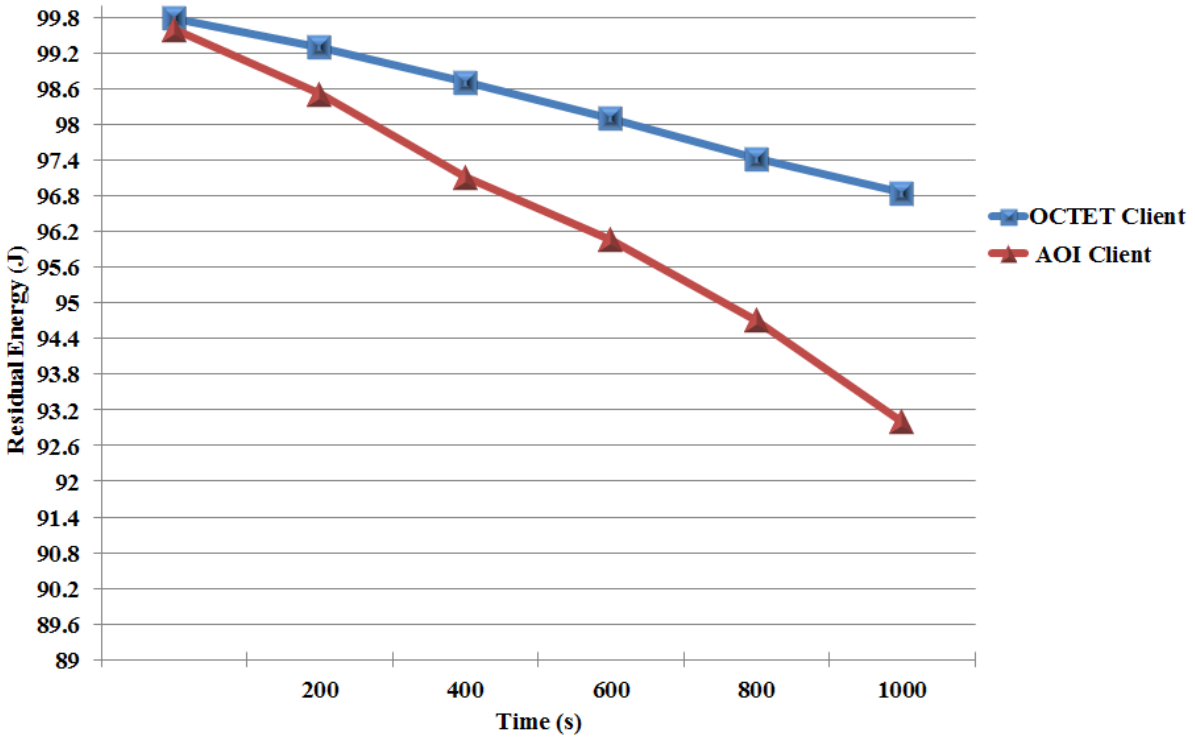

Figure 4.5: Residual Energy VS Time

For a simulation time of 50 seconds, the energy consumption is not very significant for either protocol, being around 99.4J. By increasing the simulation time, the variation between the two schemes becomes more obvious. As shown in Figure 4.5, after 600s, the mobile device's residual energy reaches 98.2J for the OCTET client and 95.8J for the AOI client. Eventually, after 1000s, the residual energy is 97J for the OCTET client and 92.9J for the AOI client.

A second experiment targeted the behaviour of the mobile's energy consumption when different numbers of nodes are applied. As shown in Figure 4.6, the user-AOI scheme behaves in an adverse manner when the number of mobile nodes is increased. Recalling that

the same source selection is applied for both schemes, the increase in the requester's energy consumption is caused by selecting undesirable objects. This in turn causes the provider's energy consumption to also increase due to the increasing number of incoming requests. In this way, the AOI model is seen as an unsuitable model to use in terms of system scalability.

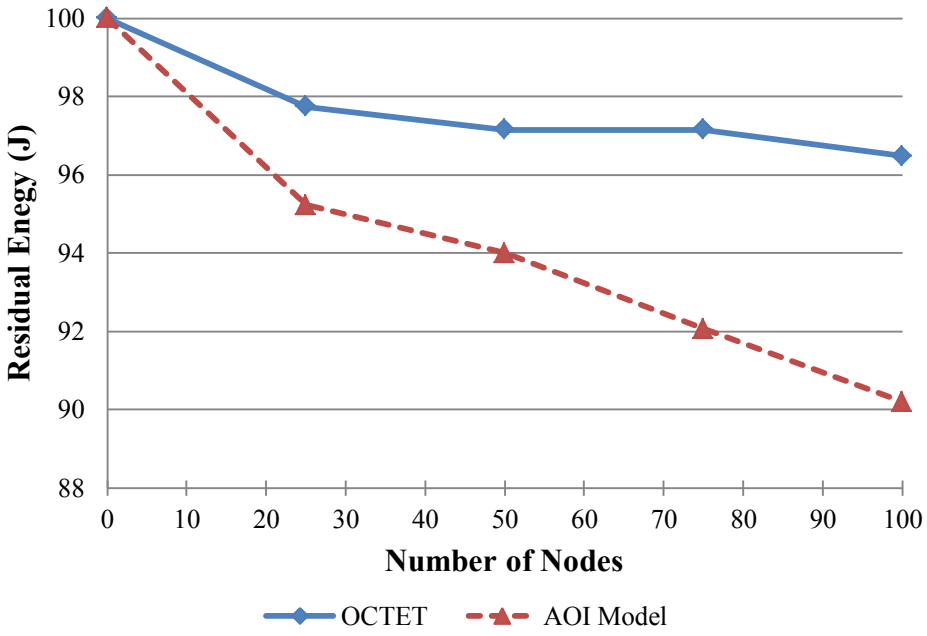

Figure 4.6: Residual Energy VS Number of Nodes

Conversely, the residual energy is one of OCTET's object selection criteria, so different sets of objects will be selected to be streamed, according to the mobile node energy level. Moreover, in OCTET, the residual energy will also affect the rendering resolution of each selected object so that any energy degradation will automatically affect the number of requested data messages, avoiding any energy dissipation for both the requestor and the provider peers. Overall, looking at the mobile hardware residual energy consumption, OCTET deals better with the restricted specifications of mobile devices.

4.5 Client's Download Rate

Figure 4.7 represents the bandwidth usage of both the OCTET and the AOI models. The significant reduction in bandwidth consumption is first clearly noticed around Time = 200s.

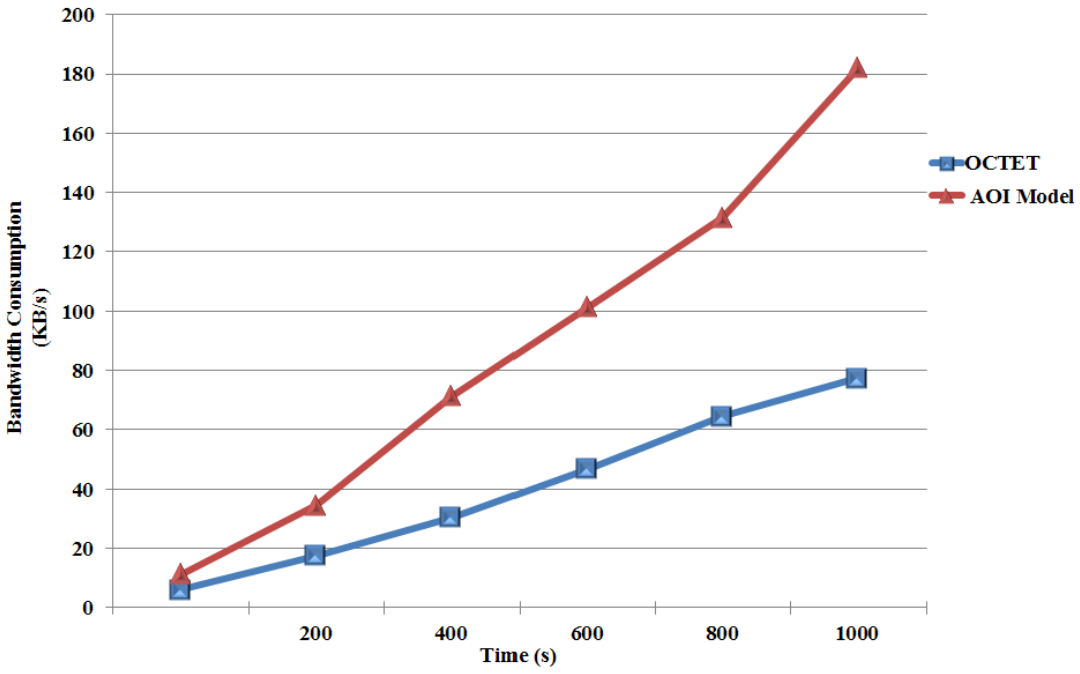

Figure 4.7: Bandwidth Consumption VS Time

By increasing the simulation time, we notice that OCTET behaves better than the AOI protocol in terms of bandwidth consumption. This can be explained by the following:

a) OCTET limits the number of downloaded 3D objects to those that contribute to the client's scene while taking into consideration the activity undertaken by the client.

b) OCTET applies a double prioritization technique for the 3D objects and its components (i.e., base and refinement pieces), which means that the 3D content is not downloaded in a random manner.

c) OCTET sets the number of refinement layers for a given 3D object to be proportional to the mobile device resources. If any degradation in the client bandwith occurs, the number of refinements layers will be lowered to save the mobile resources while keeping the 3D object to be rendered in its coarse view.

4.6 Average Response Time

The average response time is the average time between initiating a query to the RM and the time the selected 3D content has been downloaded in the client memory buffer. It is a good

indicator of the streaming quality as it quantifies how soon the 3D scene can be displayed on the mobile screen.

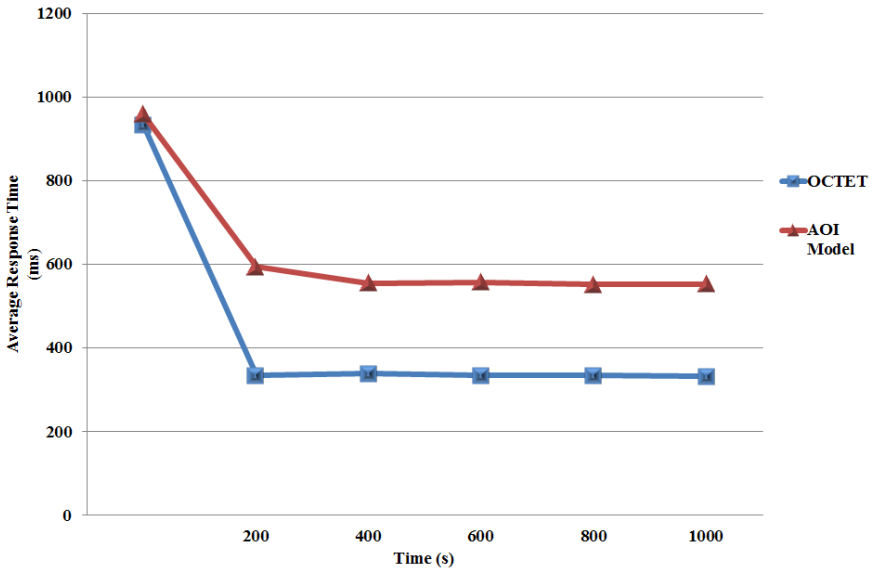

Figure 4.8: Average Response Time VS Simulation Time

As shown in Figure 4.8, regardless of the simulation time, the node mobility, and the node speed, the average response time is reduced by 45%. Hence, the system latency is reduced, allowing for a better streaming quality to be achieved.

4.7 OCTET VS Client-Server 3D Streaming

In this section, we quantitatively compare our proposed protocol to a client/server 3D streaming that adopts the AOI model as a visibility determination algorithm. To do so, different network performance metrics were collected through another set of experiments. These metrics are:

- **Server Outgoing Bandwidth**
- **Central Delivery Ratio**
- **End-to-End Delay**
- **Control Overhead**

4.7.1 *Server Outgoing Bandwidth*

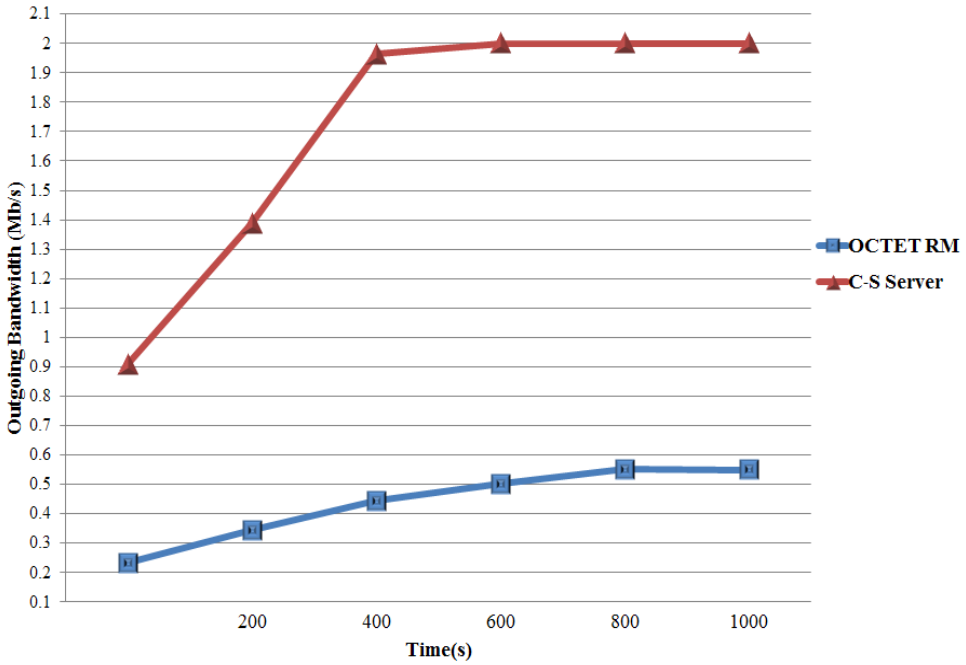

Figure 4.9: Outgoing Bandwidth VS Simulation Time

As shown in Figure 4.9, the C-S server exhausts its resources and reaches its maximum outgoing bandwidth at Time = 400s, which in reality indicates that the server is not scalable and could hang at any time. However, our region master's outgoing bandwidth remains relatively constant and stays just beyond 0.5 Mbps. Also, it is worth mentioning that this little increase in RM outgoing bandwidth is due to periodically sending both the list of surrounding objects (LAY_{Area}) and the list of content providers. It is important to keep in mind that the RM also acts as a gateway of last resort when the client fails to get the content from other peers.

4.7.2 Central Delivery Ratio

Figure 4.10 illustrates the Central Delivery Ratio (CDR) by number of nodes. We define the CDR as the ratio of the number of 3D reply packets that have been obtained in a centralized way from the server to the number of 3D reply packets that have been obtained from other providers' peers in a distributed way. The CDR for both approaches was measured as the number of mobile nodes varied from 25 to 100.

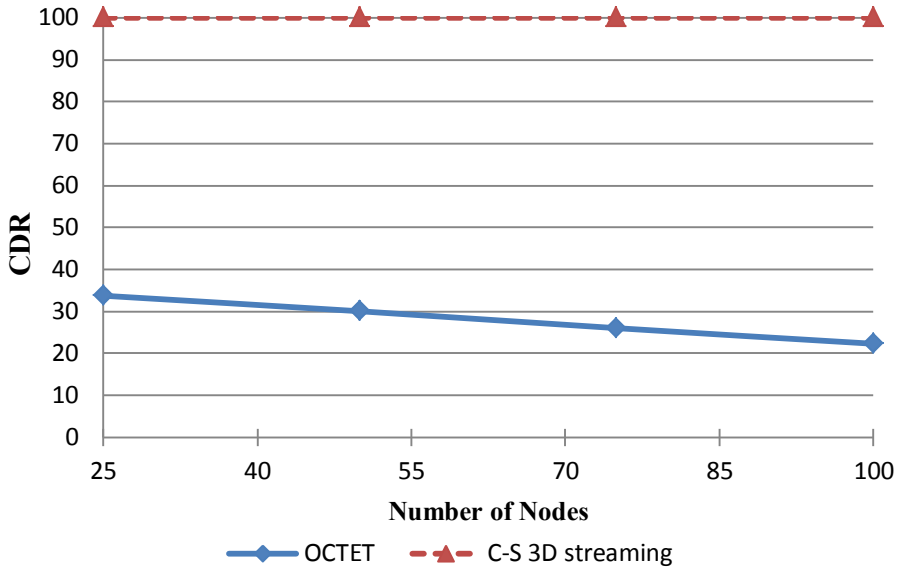

Figure 4.10: Central Delivery Ratio VS Number of Nodes

In Client-Server 3D streaming, the server acts as the sole content provider, so the central delivery ratio will always be at 100%. On the other hand, OCTET's client obtained around 70% of the required 3D content in a full P2P manner so that the CDR stayed relatively constant at around 30%. As a matter of fact, the CDR slightly decreased as the number of mobile nodes increased, since the probability to find content provider peers became very high.

4.7.3 *End-to-End Delay*

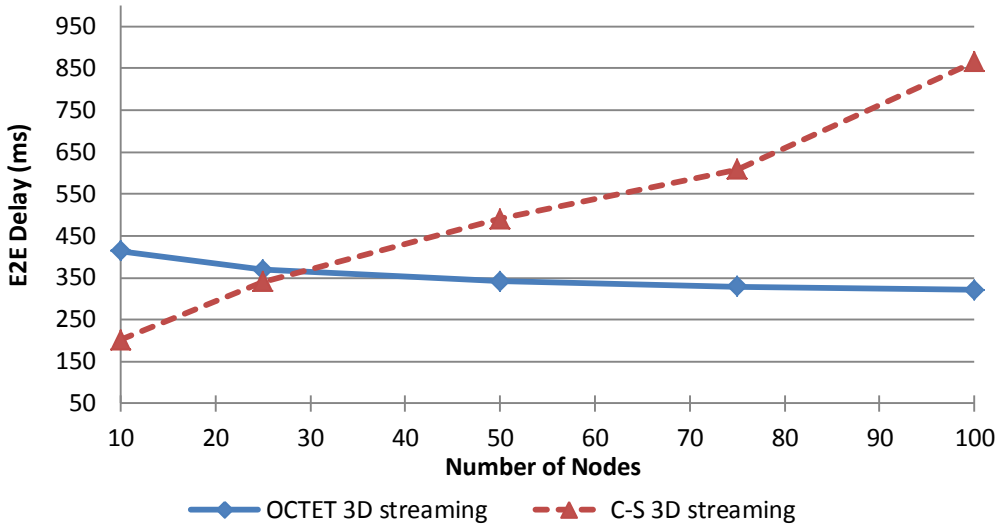

Figure 4.11: End-to-End delay VS Number of Nodes

Figure 4.11 illustrates the end-to-end delay generated by the two protocols. This metric is represented by the average amount of time measured from the instant a data message containing the 3D content is created until the packet is successfully received at the client node. Initially, from what can be seen in these results, the Client-Server approach behaved better than the OCTET 3D Streaming approach. This is because the number of mobile peers is much smaller and the clients' requests could be handled from a single place. However, the Client-Server end-to-end delay started to increase linearly with the increasing number of nodes. The server became the bottleneck and could not handle all the generated requests simultaneously. On the other hand, the end-to-end delay in OCTET remained relatively stable at around 350 ms with a slight decrease as the number of nodes increased.

In addition, the adverse behaviour of OCTET for the lesser amount of nodes can be explained by two main factors. First, low node density per area produces an increase in the distance between the source and the client, so packets need additional time to reach the client. Secondly, a small number of nodes will create an invalid request that will negatively affect

the time to get the data. Consequently, other evidence of better streaming quality and scalability of our protocol has been examined.

4.7.4 Control Overhead

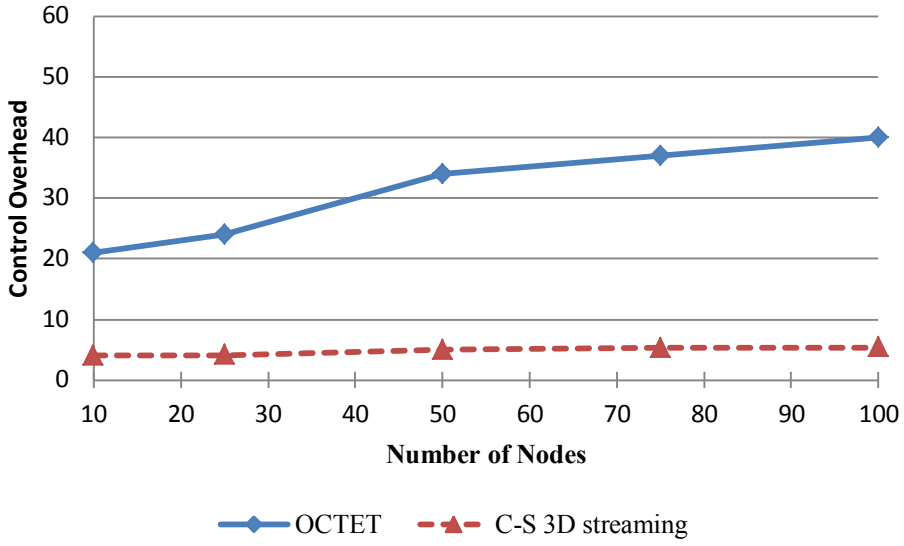

Figure 4.12: Control Overhead VS Number of Nodes

Finally, the two streaming schemes are compared in terms of the control overheads they generate. We define the control overhead as the ratio of the control packets generated by the protocol to the total number of packets in the scenario. As shown in Figure 4.12, the Client-Server has a very small number of control packets, so a very low control overhead is observed. In fact, the peer location messages are the only control packets in C-S 3D Streaming. On the other hand, although a dynamic assignment of the T_{phase} is used, OCTET's control hello messages still incur an additional control overhead. Nevertheless, this overhead is considered to be acceptable since the size of the hello messages is negligible compared to the size of 3D reply messages.

4.8 Summary

In this chapter, we evaluated the performance of our proposed protocol OCTET. The simulation implementation was done using ns-2 as a discrete event simulator. We compared our object selection strategy to the AOI model, the most widely used visibility algorithm in MMOGs and 3D streaming frameworks. Our results show that OCTET achieves better performance in terms of mobile resource utilization, the variety of the selected objects, throughput, and system latency. We also evaluated the efficiency of our source selection protocol when compared with the single content delivery used by Client-Server based 3D streaming. OCTET reduces about 75% of the server workload, which indicates better system scalability. Moreover, as the number of mobile peers increases, both the end-to-end delay and control overheads stay relatively constant, which indicates better streaming quality.

Chapter 5

Conclusion and Future Work

Since the moment internet applications appeared on handheld devices, users have dreamed about interactively accessing online virtual environments via these devices. The larger and more dynamic the 3D content, the more realistic the 3D virtual worlds become. Currently, the limited storage size of current end user's machines makes it impossible to pre-install the entire VE content in order to start interacting with these applications. Therefore, the domain of 3D streaming and the content of interest becomes fertile ground for novel approaches and new technologies.

In this thesis, we presented a comprehensive survey about 3D streaming techniques as well as the state-of-the-art visibility determination schemes in both local and remote-based rendering applications. Moreover, from the perspective of using different network architectures (client-server and Peer-to-Peer), overviews of the existing 3D streaming solutions for a thin mobile environment were introduced. This provided a broad view of existing 3D streaming protocols and frameworks. We then highlighted the limitations of these solutions when it applied to MANETs, where the hardware's limited capabilities and the narrow bandwidth of the wireless networks were the main issues.

Next, we formulated and implemented a hybrid-based 3D streaming protocol, which we refer to as OCTET, for enabling 3D scene streaming over thin mobile devices. OCTET is mainly composed of two sub-protocols: the object selection protocol and the source selection protocol. The object selection protocol combines both context aware and distance-based approaches in order to select the 3D objects of interest for the user's scene. This selection is based on a set of criteria that includes the user's action of interest, the available resources on

the mobile device, and the importance of the object in the user's field of view. Our method adapts the progressive mesh technique for model representation and transmission; hence, no redundant information is required to be sent across the network. Furthermore, double prioritization schemes are used to set the visual relevance of the 3D objects based on their contribution to the user's final scene. The source selection protocol aims to rapidly and simply select who is the suitable provider hosting the selected 3D objects from the previous phase. We adapt a centralized one-hop selection strategy where another peer in the network is selected as a provider based on its downloading time, available bandwidth, and mobility speed. We also increase the reliability of the source selection protocol by introducing a viewpoint similarity parameter, which gives a redundant solution to reach the providers when the centralized method fails.

To evaluate the performance of OCTET, implementation and an extensive set of simulations were carried out in the ns-2 environment, and our results show an improvement in communication overhead, latency, mobile resource utilization, throughput, and system scalability.

5.1 Future Work

Although this dissertation presents a comprehensive and evaluative study about 3D streaming, further research directions can be done towards the ultimate endeavour of deploying 3D streaming over thin mobile devices. A small description of these future directions is listed below.

First, enabling a pre-fetching mechanism by implementing prediction schemes can allow the users to cache the objects required to render the next possible future scene. This would reduce the number of client requests and control messages, thus enhancing the system latency and scalability. Secondly, applying hashing and encryption algorithms to provide

secure peer-to-peer 3D content delivery is a very important area to investigate. More specifically, there is a need to detect malicious peers trying to tamper with the shared 3D content. By doing so, the client can independently select the suitable content and its trusted provider without any involvement from the server side, and 3D streaming based on a full P2P overlay could be achieved. Finally, C++ OpenGL [47] 3D models could be integrated with any discrete event simulator [48], such as Omnet++, so that a real implementation of the VE could be achieved while reflecting more realistic results.

Appendix A

Glossary of Terms

AC:	Visual acuity
AOI:	Area of Interest
AR:	Augmented Reality
BIF:	Binary Format for Scenes
CIA:	Central Intelligence Agency
C-S:	Client Server
CPM:	Compressed Progressive Meshes
CPU:	Central Processing Unit
FLoD:	Flow Level of Detail
FOV:	Field Of View
GPS:	Global Positioning System
GPU:	Graphical Processing Unit
HMD:	Head Mounted Device
IBR:	Image Based Rendering
IM:	Interest Management
$\mathbf{LAY_{Area}}$**:**	Area Layout
$\mathbf{L_{AOIobj}}$**:**	List of objects inside multi-level AOI
$\mathbf{L_{neighbours}}$**:**	List of Neighbouring Peers
$\mathbf{L_{objInt}}$**:**	List Of Interested Objects Per Action
$\mathbf{L_{objVis}}$**:**	The set of visible objects

LOD:	Level of Detail
LODDT:	Level of Detail Description Tree
LSP:	List of Content Providers
LVS:	List of Viewpoint Similarity Providers
MANET:	Mobile Ad hoc Networks
MMOGs:	Massively Multiplayer Online Games
ns-2:	Network Simulator 2
NVE:	Networked Virtual Environment
PDA:	Personal Digital Assistant
PM:	Progressive Meshes
P2P:	Peer-to-Peer
R_{AOI}:	Radius of AOI
RM:	Region Master
TTL:	Time To Live
TR:	Time the RM Received an Area Layout Request
VE:	Virtual Environment
VRML	Virtual Reality Mark-up Language
VS:	Viewpoint similarity
WoW:	World of Warcraft
X3D	An Extensible Markup Language -Based 3D Graphics
2D:	Two Dimensional
3D:	Three Dimensional
3G:	Third Generation

Bibliography

[1] T. Akenine-Moller, T. Moller, E. Haines, *Real-Time Rendering*, A. K.Peters, Ltd., Natick, MA, USA, 2002.

[2] S.Y. Hu, et al., *FLOD: A framework for peer to peer 3D streaming*, Proceedings of the IEEE Conference on Computer Communications INFOCOM, pages: 1373-1381, 2008.

[3] Boukerche and R. W. Pazzi, *Remote Rendering and Streaming of Progressive Panoramas for Mobile Devices*, In Proceedings of the 14th Annual ACM international Conference on Multimedia, MULTIMEDIA 06, pages: 691-694, 2006.

[4] R. Cavagna, et al., *P2P network for very large virtual environment*, Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST), pages 269-276, 2006.

[5] Eliya Buyukkaya, Maha Abdallah, and Romain Cavagna. *Vorogame: A hybrid p2p architecture for massively multiplayer games.* In Proc. IEEE CCNC, 2009.

[6] J. Royan et al., *Networked based visualization of 3D landscapes and city models*, IEEE Computer Graphics and Applications (CGA), Volume 27, Issue 6 pages: 70-79, 2007.

[7] H. Hoppe. *Progressive Meshes*, Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH'96), pages: 99-108, 1996.

[8] R. Pajarola and J. Rossignac, *Compressed Progressive Meshes*, IEEE Transactions on Visualization and Computer Graphics, Volume 6, Issue 1, pages: 79-93, 2000.

[9] Jimmy Chim, Rynson W. H. Lau, Hong Va Leong, and Antonio Si. *Cyberwalk: A web based distributed virtual walkthrough environment*. IEEE Trans. on Multimedia, 2003.

[10] J. Calvin, A. Dickens, B. Gaines, P. Metzger, D. Miller, and D. Owen. *The SIMNET VirtualWorld Architecture*. In Annual International Symposium Virtual Reality IEEE, pages 450–455, Sept. 1993.

[11] Frey, Royan, Piegay, Kermarrec, Anceaume, Le Fessant, and Solipsis. *A Decentralized Architecture for Virtual Environments*, IEEE international Workshop on Massively Multiuser Virtual Environments (MMVE'08), pages: 29-33, 2008.

[12] NS2 -The Network Simulator, http://www.isi.edu/nsnam/ns

[13] B. Seo and R. Zimmermann, *Quantitative Analysis of Visibility Determinations for Networked Virtual Environments*, Journal of Visual Communication and Image Representation, 2012.

[14] B. Seo and R. Zimmermann, *Edge indexing in a grid for highly dynamic virtual environments,* Proceedings of the 14thannual ACM international conference on Multimedia (MULTIMEDIA 06), pages 402411, 2006.

[15] H. Rahimi et al., *Context-Aware Prioritized Game Streaming, in Proc. of Workshop on Interactive Ambient Intelligence Multimedia Environments*, in Proceedings of IEEE International Conference on Multimedia and Expo (ICME 2011), 2011.

[16] eMule. http://www.emule.com/ .

[17] World of Warcraft. http://www.worldofwarcraft.com/ .

[18] J. M. Pullen and D. C. Wood. Networking Technology and DIS. IEEE, 83(8):1156–1167, August 1995.

[19] Institute of Electrical and Electronic Engineers. 1516-2000, IEEE Standard for Modeling and Simulation High Level Architecture – Framework and Rules. IEE, September 2000. New York, NY, USA.

[20] M.Zhu et.al, *Towards peer-assisted rendering in networked virtual environments*, Proceedings of the 19th ACM international conference on Multimedia, 2011.

[21] bittorrent. http://www.bittorrent.com/ .

[22] Sony PlayStation 3. http://www.en.wikipedia.org/wiki/PlayStation_3

[23] Xbox360. http://xbox.com/ .

[24] Nintendo's Wii. http://wii.com.

[25] J. Botev et al., *TheHyperVerse Concepts for a Federated and Torrent Based 3D Web*, Intl J. Advanced Mediaand Communication, vol. 2, no. 4, 2008.

[26] Sterkin. *Interactive 3D streaming*. Research@Intel Blog, June 2008. Retrieved 1 April 2010.

[27] H. Liang, M. Motani, and W.T. Ooi, *Textures in Second Life: Measurement and Analysis,*Proc. 14th IEEE Int'l Conf. Parallel and Distributed Systems, Workshop P2P- NVE (ICPADS 08), IEEE Press, 2008, pp. 823–828.

[28] S.-Y. Hu et al., *Peer-To-Peer 3D Streaming*, IEEE Internet Computing, vol. 14, no. 2, 2010.

[29] Imesh. http://www.imesh.com

[30] C.H. Chien et al., *Bandwidth aware P2P 3D Streaming*, International Journal of Advanced Media and Communication(IJAMC), Volume 4 Issue 4, pages: 324-342,2010.

[31] Google Earth. http://www.google.com/earth/learn/

[32] H.Maamaret.al.,*3D Streaming supplying partner protocols for mobile collaborative exergaming for health*. IEEE Transactions on Information Technology in Biomedicine.2011.

[33] Second Life (SL). http://www.secondlife.com/.

[34] Shun-Yun Hu. *A case for 3d streaming on peer-to-peer networks*. In Proceedings of the eleventh international conference on 3D web technology (Web3D), pages 57{63, 2006.

[35] S. Kircher, and M. Garland. *Progressive multi resolution meshes for deforming surfaces*, In Proceedings of the 2005 ACM Siggraph/Eurographics Symposium on Computer Animation, pages: 191-200, 2005.

[36] WL Sung et al., *Selection strategies for peer to peer 3D streaming*, Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video, pages:15-20, 2008.

[37] T. Fang and Z. Tian, *efficient storage and progressive rendering of multi-resolution mesh,* Advances in Multimedia Information Processing (PCM), Volume 48, pages: 814-821, 2007.

[38] H. Li, B. Prabhakaran, S*mart Decision Module for Streaming 3D Meshes over Lossy Networks,* Proceedings of the International Conference on Distributed Multimedia Systems and International Workshop on Visual Languages and Computing (DMS/VLC'04), pages: 275-278, 2004.

[39] Andreas Gerndt, Bernd Hentschel, Marc Wolter, TorstenKuhlen, and Christian Bischof. Viracocha: *An efficient parallelization framework for large-scale cfd post processing in virtual environments.* In Proc. SC2004, 2004.

[40] Stephan Olbrich and Helmut Pralle. *Virtual reality movies - real-time streaming of 3d objects. Computer Networks*, 31(21):2215-2225, 1999.

[41] L. Mcmillan and Gary Bishop, Plenoptic modeling: *an image-based rendering sysem,* Proceedings of the 22nd annual conference on Computer graphics and interactive techniques (SIGGRAPH '95), pages: 39-46, 1995.

[42] W. Shi, Y. Lu, Z. Li, and J. Engelsma. *Scalable support for 3D graphics applications in cloud*. In Proceedings of IEEE CLOUD '10, pages 346–353, 2010.

[43] Steve Benford and Lennart E. Fahlen. *A spatial model of interaction in large virtual environments*. In ECSCW, pages 107–, 1993.

[44] W. Cheng, D. Liu, and W.T Ooi , *Peer assisted view dependent progressive mesh streaming*, Proceedings of the seventeen ACM international conference on Multimedia, pages: 441-450, 2009.

[45] Field of View FOV. http://en.wikipedia.org/wiki/Field_of_view

[46] H.Maamar. *Supplying Partners Suite of Protocols forP2P 3D Streaming Over Thin Mobile Devices.* PhD Thesis. 2012. University of Ottawa.

[47] OpenGL. http://www.opengl.org/

[48] Omnet++. http://www.omnetpp.org/

[49] Mojtaba Hosseini and Nicolas D. Georganas, *Mpeg-4 bifs streaming of large virtual environments and their animation on the web*, In Proc. ACM Web3D, pages 19{25, 2002.

[50] Rayner, *The perceptual span and peripheral cues in reading,* Cognitive Psychology, 7, 65-81, 1975.