Context Aware High-Fidelity Rendering over Peer-to-Peer Systems

Adrian De Barro

Supervisor: Dr Keith Bugeja
Co-supervisor: Dr Sandro Spina

Faculty of ICT

University of Malta

October 31, 2016

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Faculty of ICT

Declaration

I, the undersigned, declare that the dissertation entitled:

Context Aware High-Fidelity Rendering over Peer-to-Peer Systems

submitted is my work, except where acknowledged and referenced.

Adrian De Barro

October 2016

Dedication

To Jessie De Barro

Acknowledgements

I would like to thank my supervisors Dr Keith Bugeja, Dr Sandro Spina for their patience, commitment and support throughout this work. Special thanks goes to my family, especially mum and dad for their continuous support.

Finally, I would also like to thank my friends: Luke, Jordan, Josef, Kevin, Duncan and many others, with whom this year would have been more difficult.

Abstract

High-fidelity rendering has witnessed widespread adoption in a number of disciplines and areas such as engineering, archaeology, and the entertainment industry with video games and special effects amongst others. Its pervasiveness is such that any improvements to the fundamental techniques employed will see the benefits propagate to these areas of application. This work focuses on the collaborative aspect of high-fidelity rendering where we demonstrate that peer-to-peer systems are a viable alternative to traditional client-server approaches. More specifically, issues of scalability that adversely affected previous work have been shown to be related to the rendering technique rather than the networking paradigm used for communication. Results show that in addressing the shortcomings of the irradiance cache, a marked improvement in system scalability and rendering performance has been achieved, close to two-fold in speed-up. Furthermore, the epidemiological nature of information propagation fails to prioritise content, precluding peers from receiving events in order of importance. To address this limitation and improve update propagation, a context-based dissemination approach was explored, wherein peers use meta-information to direct content updates. Results demonstrate that context-aware rendering does not incur any realisation penalties; a higher update frequency between peers sharing the same contexts suggests that the approach may be capable of providing more focused updates that possess greater relevance for the receiving peer.

Contents

List of Figures

List of Tables

1. Introduction

In computer graphics, rendering is the process by which a mathematical representation of a scene or virtual environment is synthesised into an image. Rendering finds application in a number of fields, both industrial and academic; from engineering visualisations, to video games and special effects in movies. In both offline and real-time rendering (the interactive counterpart), an emerging field of study is high-fidelity rendering (HFR), which attempts to synthesise images that are indistinguishable from real-world photos. While traditional rendering was ad hoc in nature, high-fidelity rendering marries efficient computation with the fields of geometric optics, radiometry and photometry, to accurately simulate light transport during image synthesis. High-fidelity rendering is characterised by the Rendering Equation [34], an integral equation that evaluates the light intensity at any point in a scene. The rendering equation is generally intractable; no analytical formulation is available but for the simplest of scenes, and thus, solutions to it are approximated using numerical methods such as finite element and Monte Carlo simulations.

Greenberg et al. [29] developed a framework for HFR, stratifying the process into three stages:

Light reflection models are concerned with the development for arbitrary reflectance functions and their efficient representation.

Light transport simulation specifies the creation of illumination algorithms that

accurately simulate light transport within complex environment.

Perceptual issues deal with mapping the simulated radiance quantities to a display device, taking into account the physical characteristics of the device as well as the viewing conditions and the human visual system.

Greenberg's framework captures the complicated nature of lighting in HFR. In particular, the light transport simulation stage models light as it bounces across multiple surfaces before reaching the acquisition sensor (such as the observer or camera) where an image is formed. Global illumination denotes a class of algorithms that handle light transport such that light reaches the observer both directly (after a single reflection) or indirectly (after multiple reflections across multiple surfaces).

Traditionally, these algorithms have benefitted from the use of parallel and distributed computing in dedicated clusters. Interactive parallel and distributed HFR has been extensively researched [16], although seldom have low-end devices been factored in the equation, due to their lack of computational power. The advent of the cloud has brought about some changes to this status quo where service-oriented setups started making away with dedicated clusters, replacing them with a form of on-demand utility computing that offers rendering as a serivce (RaaS) [8, 4]. The computational complexity of HFR is absorbed by the cloud system, allowing complex visualisations beyond the capabilities of client devices such as tablets and smartphones. Notwithstanding this potential of cloud systems for HFR, these advances have mainly moved forward traditional offline rendering and have seldom been applied to interactive HFR. The entertainment industry has pioneered cloud game streaming [5, 9, 40], to perform interactive rendering in the cloud, and while the results may be visualised by any client device, including low-end ones, these systems do not try to advance the quality of the rendering beyond that which is possible on a powerful desktop machine.

Crassin et al. [19] introduced a system for HFR in the cloud, based on a number of highly-approximative global illumination algorithms for graphics accelerators

[20, 33, 42]. Their implementation strikes a balance between bandwidth requirements and client-side computational complexity by providing different strategies for various scenarios. Their approach distributes rendering between the client and a powerful cloud server.

Bugeja et al. [15] take a radically different approach and eliminate the centralised server component in favour of a peer-to-peer approach. They exploit collaboration between participating peers to amortise computation for speed-up gains, and implement a full high-fidelity solution using a wait-free version of the irradiance cache. The results demonstrate the viability of the approach on a heterogeneous network; however they claim that the oversaturation of the irradiance cache impact the scalability of the system, especially with large scale data sets. The propagation technique employed by Bugeja et al. is unbiased; they hypothesise that prioritising propagation on the basis of peer context may lead to lower response times for dynamic lighting.

1.1 Problem Description

The state of the art in distributed HFR is such that a number of unaddressed problems remain: streaming solutions are totally dependent on network performance and as such, fluctuations can cause disruptions in user experience. Distributed streaming pipelines such as CloudLight [19] either have high bandwidth requirements, or require manual annotation of data. Furthermore, some of the techniques employed in the client-side end of the pipeline require considerable computational power. Thus, they are not amenable for low-end devices.

Bugeja et al. [15] provide a scalable, fault-tolerant solution for collaborative environments, which also takes advantage of computation amortisation across peers. In the light of these advantageous properties, this approach would serve as a viable starting point for the work carried out in this dissertation. More specifically, this dissertation will focus on investigating novel techniques to advance the current

state of the art in peer-to-peer HFR.

1.2 Aims and Objectives

The aim of this dissertation is to advance the state of the art in peer-to-peer HFR. In particular this work is set to:

1. Investigate the state of the art in HFR to identify a rendering algorithm that overcomes the limitations of the wait-free irradiance cache.
2. Identify what and how peer context can be exploited to lower dynamic indirect lighting response times.

1.3 Methodology

The following steps enumerate the tasks comprising the approach taken for this dissertation:

- Identify a peer-to-peer (P2P) framework for P2P rendering
- Identify data sets to be used in evaluation with particular reference to:
  - size in terms of geometry complexity and dimensions
  - material properties of objects (diffuse, glossy, specular, transmissive, caustics)
  - a mixture of occluded and open spaces (for evaluation of global illumination)
- Implement irradiance cache on P2P framework [15]
- Identify an illumination algorithm that overcomes limitations of irradiance cache
- Identify peer context and design P2P overlay to take advantage of context

- Evaluate proposed solution

1.4 Thesis Outline

The rest of the dissertation is organised as follows:

- **Chapter 2:**

  This Chapter provides the key concepts related to high-fidelity rendering and the respective techniques used to approximate global illumination.

- **Chapter 3:**

  A number of techniques used for parallel and distributed rendering are examined in Chapter 3. Data dissemination techniques over peer-to-peer systems are compared.

- **Chapter 4:**

  The relationship between redundant computations and concurrent peers collaborating over an unstructured network is investigated in Chapter 4.

- **Chapter 5:**

  This Chapter provides an improved IC technique that is able to deal with over-saturation through the use of a two staging technique.

- **Chapter 6:**

  A novel data dissemination technique, through the use of an overlay is presented in Chapter 6, in which peers are presented with meta-information that allows them to prioritise message passing.

- **Chapter 7:**

  This Chapter concludes the dissertation by summarising the achievements of this work inline to the objectives set in the previous section. Future work to improve the techniques presented in this dissertation is also presented.

2. Background

In this chapter, the reader is provided with the necessary background required to better understand the chapters that follow. The chapter introduces the fundamental components of a scene, followed by defining materials and the introduction of the rendering equation in its three most common forms. Monte Carlo methods are introduced to the reader following an introduction to different rendering techniques.

2.1 Composite of a scene

A scene is made up of a number of objects and each object enclosed in the scene is made up of one or more surfaces. Each surface is defined by some equation which defines the set of 3D points enclosed by the surface. Some algorithms do not handle the original representation of the model but approximate the surfaces of the model through a process known as *tessellation*. Tessellation approximates a surface through the use of geometric primitives.

2.2 Geometric primitives

Geometric primitives are atomic shapes which are used as building blocks when constructing an approximate mesh of an object. A mesh is the agglomeration of geometric primitives for a particular object. Shirley et al. [53] explain that the

simplicity of the shape is what determines if the shape is recognised as a geometric primitive since simplicity allows hardware to exploit its simplistic construction and representation when processing these primitives. Through the use of the following three primitives, the most complex of geometries can be approximated.

1. *Points:* A point can be defined as a location on an arbitrary amount of dimensions. Moreover, it specifies a location in a specific coordinate system.

2. *Lines:* A line has nor width nor height. In fact, it is considered as a 1-dimensional construct which extends infinitely across a particular dimension.

3. *Polygons:* Polygons are used to represent surfaces; this is achieved by tessellating surfaces using triangles, quads and n-polygonal shapes.

2.3 Orthonormal Basis

Coordinate systems allow expressing the position and orientation of different objects in a scene on themselves and other objects. Through the use of orthonormal basis, vectors can be converted between various coordinate systems. Each component of the orthonormal basis is orthogonal to all the other components of the same bases. Each vector is of unit length, since each element defines only a direction.

An arbitrary point p found on a surface of a geometrical primitive has its orthonormal basis defined as provided in figure 2.1, where $\hat{\mathbf{n}}$ can be expressed as the surface normal, $\hat{\mathbf{b}}$ as the binormal and $\hat{\mathbf{t}}$ as the surface tangent.

The default orthonormal basis is known as the canonical basis and is commonly represented by the $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ directions. Any vector from a different coordinate system can be converted to the canonical basis through the application of equation 2.1. The resulting vector $\hat{\mathbf{r}}$, is $\mathbf{a}$ converted to the canonical basis.

$$\begin{aligned} \mathbf{a} &= (a_x, a_y, a_z) \\ \mathbf{r} &= a_x \cdot \hat{\mathbf{x}} + a_y \cdot \hat{\mathbf{y}} + a_z \cdot \hat{\mathbf{z}} \end{aligned} \tag{2.1}$$

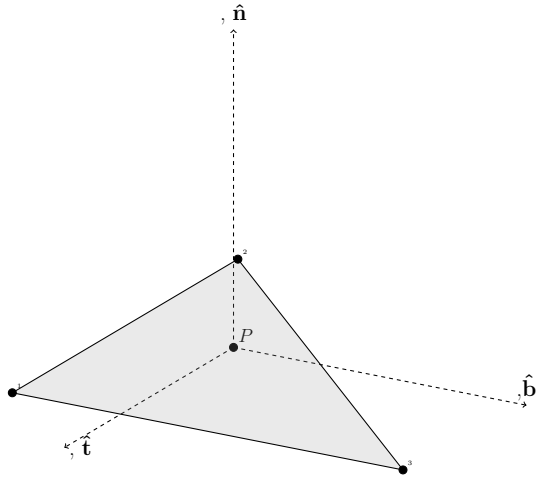


Figure 2.1: Geometric properties of a point P on a surface.

Similarly, by generalizing equation 2.1, any given direction can be converted with respect to any orthonormal system through the use of Equation 2.2.

$$\begin{aligned} \mathbf{a} &= (a_x, a_y, a_z) \\ \mathbf{r} &= a_x \cdot \hat{\mathbf{u}} + a_y \cdot \hat{\mathbf{v}} + a_z \cdot \hat{\mathbf{w}} \end{aligned} \tag{2.2}$$

Given two arbitrary vectors, $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$, it is possible to provide an orthogonal system composed from components; $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$. This can be achieved through the application of Equation 2.3.

$$\begin{aligned} \hat{\mathbf{u}} &= \frac{\hat{\mathbf{a}}}{\|\hat{\mathbf{a}}\|} \\ \hat{\mathbf{v}} &= \frac{\hat{\mathbf{a}} \times \hat{\mathbf{b}}}{\left\|\hat{\mathbf{a}} \times \hat{\mathbf{b}}\right\|} \\ \hat{\mathbf{w}} &= \hat{\mathbf{w}} \times \hat{\mathbf{u}} \end{aligned} \tag{2.3}$$

2.4 Solid Angles

An arbitrary point on a surface is enclosed in a hemisphere with radius r and since the hemisphere is of unit size, the radius r is equal to 1. All positions on the hemisphere can be represented through the use of two variables, ϕ and θ. ϕ known

as the *azimuth*, represents the angle produced with the surface tangent. On the other hand, θ, known as the *altitude*, accounts for the angle produced from the surface normal, this is represented in Figure 2.3.

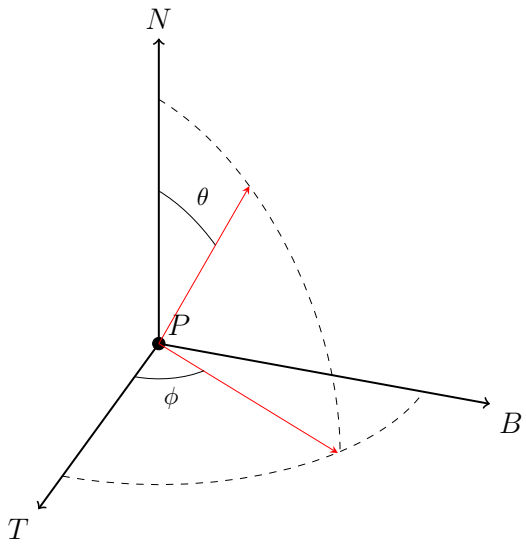


Figure 2.2: The ϕ and θ angles provide the components required for solid angles.

The hemisphere can be evaluated as a two-dimensional space representing all the possible directions through which rays of light can either be incoming towards the point or reflected from the point. Any vector can be converted into hemispherical coordinates through the use of the following equations:

$$\begin{aligned} x &= r\cos\phi\sin\theta \\ y &= r\sin\phi\sin\theta \\ z &= r\cos\theta \\ \hat{\mathbf{v}} &= (x, y, z) \end{aligned} \tag{2.4}$$

Physically based rendering (PBR) requires acquiring all the incoming light from all surrounding surfaces. This can be obtained by integrating over the hemisphere. To obtain such a function, a measure over the hemisphere is required, which is obtained through solid angles.

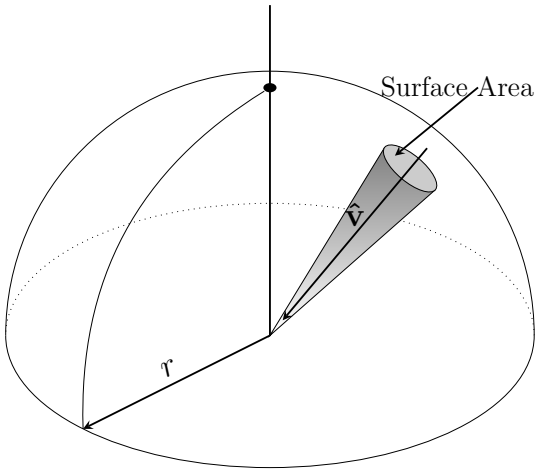


Figure 2.3: The solid angle is the projection of an object over the hemisphere

By definition, a solid angle can be defined as a cone having its size determined by the surface area subtended by the cone. Taking into consideration an infinitesimally small surface area over the hemisphere allows the representation of a solid angle to be made by a vector $\hat{\mathbf{v}}$. The solid angle Ω subtended by the area on the hemisphere is defined as provided in figure 2.3.

$$\Omega = \frac{A}{r^2} = \frac{2\pi \cdot r}{r^2} = 2\pi \tag{2.5}$$

By default, r, the radius of the hemisphere, is equal to 1 since the hemisphere is of unit length. Thus, the area subtended by the hemisphere is equal to 2π. Moreover, solid angles are measured in *steradians* (SR). Solid angles are not affected by the shape of the area subtended by the hemisphere, but are only dependent on the enclosing area. Thus, two shapes having different projection over the hemisphere can have the same solid angle. For the computation of solid angles, the surface is projected on the hemisphere, and the solid angle of the projection is computed [26]. For smaller surfaces another equation is needed such that an approximation of the enclosed solid angle is computed, presented by Equation 2.4.

$$\Omega = \frac{A \cdot \cos\alpha}{d^2} \tag{2.6}$$

2.5 Models of light

Different models exist to model light behaviour. Each model makes different assumptions or applies a different theory for the behaviour of light. Three most commonly known models of light are *quantum optics*, *wave optics* and *geometric optics*.

Quantum optics model light at the wave-particle level. Due to its vast amount of detail the model is not used for PBR. Similarly, wave optics is the simplification of the quantum optics model as described by *Maxwell's equations*. This method allows the capturing of particular light effects such as diffraction and polarisation, but as explained by Dutre et al. [26], for the purpose of image generation, this model is quite often ignored.

On the other hand, geometric optics is a more simplistic model of light that is commonly used in computer graphics. This model assumes that light has a constant wavelength that is much smaller than the objects with which the light rays are interacting. Further assumptions considered by geometric optics are listed as follows:

- Light travels in straight lines.
- Light travels instantaneously
- Light is not influenced by any external forces such as gravity and magnetism.

The goal of global illumination (GI) algorithms is the ability to capture the steady-state distribution of light in the scene. Steady-state distribution means that the light transferred between all surfaces does not change with time, thus, reaching an energy equilibrium. The balance of light energy in a scene can only be sustained on the assumption that the system's properties do not change either. For

the computation of steady-state distribution of light energy in a scene, the quantification of the respective light energy is required, an approach that is delivered by the field of *radiometry*.

2.6 Radiometry

Radiometry is the field of study which deals with the measurement of electromagnetic radiation. Results from this field are obtained in power (*Watts*), or can also be expressed in terms of photon flux (*photons per second*). Light in radiometry is modelled according to the field of geometric optics. Thus, the light model is simplified through the use of the assumptions stated in the section 2.5. The following section will be describing how light is quantified in the field of radiometry.

2.6.1 Radiometric quantities

Radiant energy is the light energy emitted, reflected and received by an arbitrary surface in a scene. The basic quantity in radiometry is the *radiant flux* or *radiant power*, denoted as Φ. Flux represents the amount of radiant energy being emitted, transferred or received by a surface at any point in time.

The amount of radiant power received by a point on a surface is referred to as *irradiance*, defined as the radiant power dQ per unit projected area dT and expressed in $Watts/m^2$, illustrated by Equation 2.6.1.

$$\Phi = \frac{dQ}{dT} \tag{2.7}$$

The convention requires a distinction between incoming radiant power and outgoing radiant power per unit area. *Radiant exitance* is the total amount of outgoing light from a point per unit projected area, and likewise, it is measured in $Watts/m^2$.

$$M = B = \frac{d\phi}{dA} \tag{2.8}$$

Radiance is the Radiant power per projected area per unit solid angle. As the definition states, *radiance* is the total amount of flux received by a certain point on a surface per unit solid angle and unit projected area, as illustrated in Equation 2.6.1. Thus, Radiance varies according to the position and the direction of incoming light.

$$L = \frac{d^2\phi}{d\omega \cdot dA^{\perp}} = \frac{d^2\phi}{d\omega \cdot dA\cos\theta} \tag{2.9}$$

Equation 2.6.1 applies a cosine along with the projected area, this stems from the fact that when the light comes at an angle, the surface area receiving light increases. Thus, the radiant flux would be distributed to more points, and, therefore, less radiant energy is given per unit surface area.

2.7 Bidirectional Scattering Surface Reflection Distribution Function (BSSRDF)

Different surfaces interact differently with light. The interaction that occurs between a surface and light depends on the properties of the surface. Upon collision with a surface, radiant energy can either be reflected back into the scene or absorbed by the surface and scattered as heat. Most materials combine both, i.e., a percentage of the incoming radiance is reflected back into the scene while the surface absorbs the remainder of the incoming flux. Thus, different surfaces can be modelled by differing the ratio of absorption to reflectance.

The function which governs the amount of energy which is absorbed and reflected by a surface is known as the BSSRDF. As stated by [33], the BSSRDF function is a function which takes into consideration both the incoming direction and

position of the light rays; moreover, it also takes into account the exitant position and direction of the same light. The exitant position of light cannot be assumed to be equal to the incoming position since there exist materials which allow scattering and transmission of light through the surface. Light scattering is a phenomenon that occurs when light energy is transmitted through the surface and, in turn, the light rays would then exit from another point in the same material, particularly noticeable in skin and marble. The only assumption assumed in the BSSRDF is that when transmission occurs, the transmitted light would instantaneously exit from a secondary location [33].

For global illumination purposes, scattering and transmission are ignored. Thus, the interaction of light with different surfaces is modelled by the bidirectional reflection distribution function (BRDF). The BSSRDF is an 8-dimensional function and very costly to evaluate [33], while the BRDF is a 6-dimensional simplified version of the BSSRDF function and therefore can be evaluated more efficiently.

2.8 Bidirectional reflectance distribution function (BRDF)

The BRDF, introduced by Nicodemus et al. [44], can be defined as the ratio of incoming flux at a direction Ψ to the amount of radiance which is reflected back in an arbitrary exitant direction Θ as shown in equation 2.10. The BRDF is defined as a ratio of the exitant radiance to the incoming irradiance instead of the ratio of incoming irradiance to exitant irradiance. This is because the reflected radiance is dependant and proportional to the solid angle Θ and $\cos\theta$.

$$f_r(x, \Psi \rightarrow \Theta) = \frac{dL(x \rightarrow \Theta)}{dE(x \leftarrow \Psi)} \tag{2.10}$$

$$\frac{dL(x \leftarrow \Theta)}{L(x \leftarrow \Psi)\cos(N_x, \Psi)d\omega_\Psi} \tag{2.11}$$

2.9 Materials

The interaction an arbitrary surface exerts with incoming rays of light is defined by the material. The material's behaviour with light is modelled through a defined BRDF. Some surfaces reflect light evenly over the unit hemisphere while other surfaces may concentrate the reflection in a single direction. Moreover, a surface may even favour a particular orientation for the direction of exitant radiation. Figure 2.4, illustrates the three different types of surfaces.

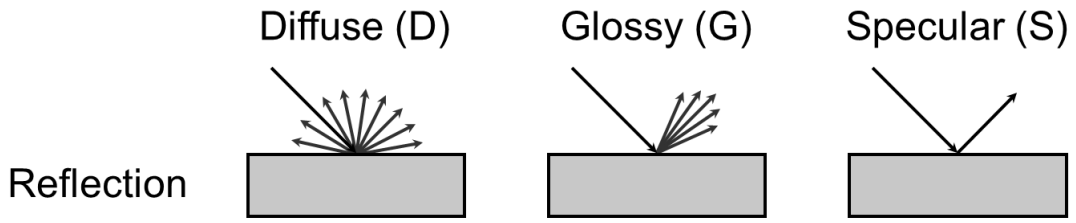


Figure 2.4: Distribution of reflected light energy by different surfaces, taken from [1]

Surfaces which distribute reflected light evenly over the hemisphere are known as *diffuse* surfaces. Diffuse reflections are produced by isotropic materials meaning that the BRDF function does not take into account the viewer's direction. Thus, radiant flux is distributed evenly in all directions. The BRDF function for diffuse surfaces can be modelled using equation 2.12, where the ρ_d represents the ratio of light energy that is reflected back into the scene.

$$f_r(x, \Psi \rightarrow \Theta) = \frac{\rho_d}{\Pi} \tag{2.12}$$

Specular surfaces are surfaces that have high reflective properties such as mirrors and glass. A perfectly specular surface reflects the incoming radiant energy into a single exitant direction. For an arbitrary incoming direction Ψ, the reflected ray can be computed using Equation 2.13.

$$\hat{\mathbf{r}} = 2(\hat{\mathbf{n}} \cdot \Psi)\hat{\mathbf{n}} - \Psi \tag{2.13}$$

Refractions must also be taken into account when dealing with specular transparent surfaces; the refraction phenomena can be defined as the bending of light towards the normal when the light is travelling in media with differing densities. The bending of light is dependent on the refractive indices of the two media, and this phenomenon can be modelled through the use of Snell's law , illustrated in Equation 2.14, where η represents the refractive index of the materials and θ represents the angle of incidence or the angle of exitance.

$$\eta_1 \sin\theta_1 = \eta_2 \sin\theta_2 \tag{2.14}$$

Most surfaces are neither purely diffuse nor ideal specular surfaces, but contain a ratio of both; such surfaces are called *glossy* surfaces. It is often difficult to model such surfaces in an analytical manner. As such, glossy surfaces need to be estimated through the use of approximate techniques.

2.10 Shading models for glossy surfaces

Glossy surfaces can be both challenging and complex to model and numerous models that capture glossy reflections exist. The *Phong* model is an example of such models. Although the Phong model captures the appearance of glossy reflections in a believable manner, however, it is still not physically correct. In fact, it is referred to as an ad-hoc model. As stated by Jensen [33], the Phong model reflects more flux than the total amount of light energy that is received at the point of interaction. Lewis in [39] addressed the issue of conservation of energy by applying a normalising factor through the use of the half vector.

On the other hand, the model proposed by Ward et al. [38] is based on empirical observations, i.e., the model is built through observations in the real world as opposed to the *Cook-Torrance* model [18], which is only mathematically correct and not empirically proven. Dutre et al. [26] state that both models are physically correct and offer an intuitive parametrisation of the BRDF.

The Cook-Torrance model assumes that the surface of the glossy material is made up of a constant random number of small smooth flat facets. During the rendering process, an arbitrary ray requiring interaction with a glossy surface will randomly hit one of the smooth facets. The model also takes into account *Fresnel* reflection and refraction. Fresnel reflections and refractions are effects which can be observed in homogeneous metals and dielectric materials such as glass and water. The Fresnel model is derived from *Maxwell's equations*[33].

The method proposed by Schlick in [52] is simple, intuitive and empirical. As stated by Jensen [33], the model is computationally efficient and can support importance sampling, a technique used in Monte Carlo ray tracing. The following three parameters govern the Schlick's approximation:

- F_0: The specular at the point of light interaction.
- σ: The roughness factor of material, $\sigma = 0$ results in a specular and smooth material, while $\sigma = 1$ results in a Lambertain material.
- ψ: The isotropy factor of the material, where $\psi = 0$ represents an anisotropic material and $\psi = 1$ results in an isotropic material.

2.11 Rendering Equation

Global illumination algorithms (GI) are a group of techniques used to solve PBR. Such techniques compute the steady state distribution of radiant flux in a scene. Propagation of light in the scene is assumed to happen instantaneously. Thus, steady-state distribution of radiant flux is also assumed to happen instantaneously too. The rendering equation applies the same concept for any arbitrary point, i.e., for any point in the scene, the rendering equation computes the amount of exitant radiance $L(x \to \Theta)$ for a particular direction Θ.

$$L(x \to \Theta) = L_e(x \to \Theta) + L_r(x \to \Theta) \tag{2.15}$$

2.11.1 Hemispherical formulation

The hemispherical formulation of the rendering equation is one of the mostly used formulations of the equation. The total amount of exitant radiance in an arbitrary direction Θ is equal to the addition of the summation of flux emitted by the point p and the amount of radiance reflected by p.

$$f_r(x, \Psi \rightarrow \Theta) = \frac{dL_r(x \rightarrow \Theta)}{dE(x \rightarrow \Psi)} \tag{2.16}$$

$$L_r(x \rightarrow \Theta) = \int_{\Omega_x} f_r(x, \Psi \rightarrow \Theta) L(x \leftarrow \Psi) \cos(N_x, \Psi) d\omega_\psi \tag{2.17}$$

The form of the equation 2.11.1 is known as a Fredholm's equation of the second kind [26] since an unknown is found on both sides of the equation.

$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{\Omega_x} f_r(x, \Psi \rightarrow \Theta) L(x \leftarrow \Psi) \cos(N_x, \Psi) d\omega_\psi \tag{2.18}$$

2.11.2 Area formulation

An alternative formulation of the rendering equation is the *area formulation*. The area formulation expresses the rendering equation as an integral over all the surfaces contained in the scene.

Since the formulation deals with all the areas in the scene, contributions of radiant flux to a point x can only be made by any point y which is not occluded from x. This visibility check requires an efficient function that tests the visibility of any two arbitrary points in the scene. Thus, any given surface point y needs to be checked if it is not occluded from point x before the contribution of point y can be computed. Visibility checks are tested through the use of *ray casting*. Ray casting samples a vector starting at point x in the direction of point y. In return, the function would deliver the closest point intersected for the given direction of

the vector. If the closest point is equal to point y it would mean that both points are visible to each others and thus, y would contribute to the radiance reflected by point x. Equation 2.19 determines if two points are occluded from each other.

$$\forall x, y \epsilon A : V(x,y) = \begin{cases} 1 \text{ if x and y are mutually visible,} \\ 0 \text{ if x and y are not mutually visible.} \end{cases} \tag{2.19}$$

Assuming no energy is lost between any two arbitrary mutually visible points x and y in a scene, the amount of radiance received from direction Ψ by x is equal to the radiance emitted by y in direction $-\Psi$ as provided in equation 2.20.

$$L(x \leftarrow \Psi) = L(y \rightarrow -\Psi) \tag{2.20}$$

The area formulation of the rendering equation requires integration over all the surfaces instead of integrating over the hemisphere. Thus, the hemispherical integral needs to be transformed into an area integral. [26] provide the relationship between a differential solid angle and a differential area which is listed in equation 2.21. In equation 2.21 the differential solid angle $d\omega_\Theta$ is being transformed to a differential surface dA_y at an arbitrary point y, thus, integration over the hemisphere can also be written as an integral for each and every visible differential area in a scene (see equation 2.22).

$$d\omega_\Theta = \frac{\cos\theta_y dA_y}{d_{xy}^2} \tag{2.21}$$

$$\int_\omega f(\Theta) d\omega_\Theta = \int_A \frac{\cos\theta_y}{d_{xy}^2} dA_y \tag{2.22}$$

Reformulating the hemispherical formulation provided in Equation 2.18, the area formulation can be constructed as listed in equation 2.23.

$$L(x \to \Theta) = L_e(x \to \Theta) + \int_A f_r(x, \Psi \leftarrow \Theta) L(y \leftarrow -\Psi) V(x, y) z \frac{\cos(N_x, \Psi) \cos(N_y, \Psi)}{r_{xy}^2} \tag{2.23}$$

2.11.3 Direct and indirect illumination formulation

Another different formulation of the rendering equation separates the direct and indirect lighting at a point. Direct illumination is the light energy that arrives directly from a light source in the scene, whereas, indirect illumination is the light which has already been reflected off another surface at least once. Thus, $L_r(x \to \Theta)$ can be split as follows in equation 2.25.

$$L(x \to \Theta) = L_e(x \to \Theta) + L_r(x \to \Theta) \tag{2.24}$$

$$L_r(x \to \Theta) = L_{direct} + L_{indirect} \tag{2.25}$$

Direct lighting can be defined as an integral over all the surface areas which emit light. Surface areas emitting light which are occluded from an arbitrary point would not contribute to the amount of radiance reflected by the point. On the other hand, the indirect lighting is an integral over the hemisphere such that the point can capture all the incoming indirect lighting from all the possible directions over the hemisphere. The combination of area integration and hemispherical integration stems from the fact that for the direct illumination, it is more efficient to integrate over all differential areas that are emitting light (§Equation 2.23). On the other hand, it is more efficient to use the hemispherical formulation for the indirect lighting (§Equation 2.18).

2.12 Monte Carlo methods

The analytical models for the rendering equation are highly complex to solve. As stated by Reif et al. [49] ray tracing is a *PSPACE-hard* problem, thus, numerical methods have to be employed to approximate the value returned by the analytical model. Due to the stochastic nature and high dimensionality of the rendering equation, Monte Carlo integration is often used to find a solution since quadrature rules cannot be used to evaluate the integral efficiently [33] [26]. Thus, Monte Carlo techniques have to be applied to find a solution efficiently. Furthermore, Monte Carlo techniques might even be the only viable solution.

$$\langle I \rangle = \frac{1}{N} \sum_{i=0}^{N} \frac{f(x_i)}{p(x_i)} \tag{2.26}$$

Monte Carlo integration makes use of *random variables* to compute and approximate a value. A random variable is a variable that can take a range of values which in turn are sampled for an N amount of times. The results would then be summed and divided by N as explained through Equation 2.26. The accuracy of the approximation depends on the number of samples used since Monte Carlo integration has a slow convergence rate of $\frac{1}{\sqrt{N}}$. As a consequence, to decrease the error of the approximation by half, the amount of samples used by the estimator needs to be quadrupled.

2.12.1 Variance reduction technique

Although increasing the number of samples would improve the results computed by the estimator, using as much knowledge as possible about the integrating function would yield better results. Two powerful variance reduction techniques are *importance sampling* and *stratified sampling*.

Importance sampling is the concentration of samples in critical areas of the hemisphere. For this purpose, a probability distribution function (PDF) is created to mimic the function that is being integrated. As such, ranges of positions

over the hemisphere are weighted according to their importance. Thus, particular orientations over the hemisphere have a better chance of being selected, allowing the preference of directions that might yield more information to the integrals approximation.

In its simplest form, stratified sampling divides the hemispherical surface area into smaller equally sized regions, where the number of regions is equal to the number of samples that is required to be cast in the scene. Each sample would be chosen randomly within the bounds of each grid; thus, the selected samples would be distributed evenly over the hemisphere. As stated by Jensen [33], if the number of samples to be used is unknown then stratified sampling becomes problematic.

2.13 Light transport notation

The path that is undertaken by a ray of light when bouncing from one surface onto another is known as a light path. It is often necessary to distinguish between the different types of surface reflections that can occur along a ray path. Heckbert introduced a notation in [30] to be able to describe the interaction of light paths with surfaces. Every vertex of a light path can be one of the items listed below:

- L represents the light source
- E represents the eye
- S represents a specular reflection
- D represents a diffuse reflection

As an example, *LDDSE*, represents a ray path which started from the **L**ight source, reflected on two **D**iffuse surfaces, reflected on a **S**pecular surface and was captured by the **E**ye. To be able to describe all the possible paths that can be reproduced by light rays Heckbert introduced the following notation:

- (k)+: one or more k events

- $(k)*$: zero or more k events.
- $(k)?$: zero or one k event.
- $(k|k')$: a k or k' event.

The notation provided above allows expressing all the paths that can be undertaken by a light ray in a specific algorithm. Combining the vertices and the notation provides expressions similar to $L(S|D)$+DE, that defines a light which will be reflected on a diffuse or specular surface for one or more times until a diffuse surface reflects it to the eye sensor.

2.14 Global illumination techniques

GI techniques provide the steady-state equilibrium of light energy in a scene. There exist many GI techniques which are able to provide an accurate representation of the steady-state distribution of light. As stated by Jensen et al. [33], most techniques can be divided into two major classes of algorithms:

- Point Sampling
- Finite element method

Both classes provides a different solution to the rendering equation by applying a specific formulation of the rendering equation (see §2.11). Both classes compliment each other in terms of weaknesses and strengths. In fact, there are techniques which combine the two approaches.

2.14.1 Finite element radiosity (FER)

The finite element radiosity (FER) introduced by [28], is a technique which provides a solution for GI through the area formulation of the rendering equation (see

§2.11.2). FER requires that the scene is subdivided into patches through which the steady-state distribution of light in the scene is computed.

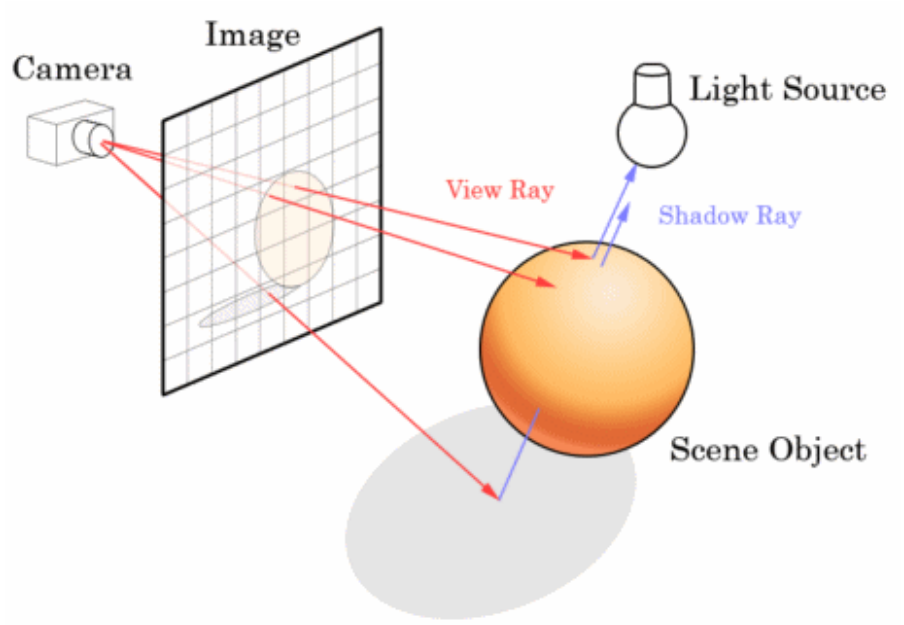
GI in FER is computed by solving a set of linear equations for the light that is exchanged between the patches in the scene. A naive implementation of the FER caters only for Lambertian surfaces. The method was then extended to cater for more complex surfaces. The solution provided by the FER is a view independent solution for the entire scene that is useful for walk-troughs yet, the solution is costly to compute. Radiosity's restriction to handling only diffuse surfaces limit the realism of the produced render. The FER was extended to handle different surfaces, such as, the use of view-dependent calculations for the handling of specular surfaces [31]. However, this technique proved to be more expensive than the general radiosity method.

2.14.2 Ray tracing

Ray tracing is an intuitive, elegant and simple. Providing an intuitive approach for the rendering of shadows and specular surfaces. In the real world, light sources would emit photons, photons travel in straight lines until they collide and bounce off a surfaces. A fraction of the colliding photons reach the eyes sensors. This process is what allows us to perceive the illumination around us.

Yet simulating this process is not practical for most scenes, energy transmitted is not affected by the medium in which it is travelling. Light energy travels in straight lines and the amount of radiance transmitted between two points is consistent. Given these two properties, light can be traced in backwards fashion. Ray tracing basis itself on the work of Appel [**?**]. Appel provided a solution to provide more solidity to computer renders such as, the rendering of mechanical parts and buildings. This was achieved by Appel through the introduction of the ray casting operator. The ray casting operator provides the capability to test if a point in the scene lies in a shadow or not, this is also known as the *visibility* function.

The goal of ray tracing is to compute the colour value of each and every pixel



[]

Figure 2.5: The basic ray tracing concepts, taken from [3].

in the image plane. This is achieved by tracing rays through the image plane. The plane is made from a grid, with each cell of the grid representing a pixel.

Rays are traced through each pixel done by tracing one or multiple rays through each pixel, which are known as primary rays. Testing for shadows is an intuitive process that basis itself on the ray casting operator. Upon an interaction between a surface and a primary ray, a secondary ray are traced in the direction of the light source from the collision point, as illustrated in Figure 2.5.

The geometry of the scene is treated as a black box, achieved by querying the geometry for collision. Unlike radiosity's tessellation process, the ray tracing technique is not restricted by the complexity of the scene geometry. Ray tracing is only capable of dealing with specular and diffuse surfaces, specifically *LD?S*DE*. Thus, this model omits diffuse-diffuse reflections and diffuse-specular reflections. The accuracy of the model is controlled at the image level. Unlike radiosity, the ray tracing model executes only in places where the view plane is directed, unlike the former method that provides a solution for the whole scene.

The ray tracing algorithm is not a full global illumination algorithm since it

cannot compute the indirect lighting in a scene. Moreover, the basic ray tracing algorithm is not able to compute phenomena such as motion blur and camera focus effects, most of which are approximated through the use of sampling. The limitations of this technique led researchers to expand ray tracing further through the use of stochastic techniques.

2.14.3 Stochastic ray tracing

Monte Carlo (MC) ray tracing techniques are the most general class of global illumination methods. Although ray tracing is simple and elegant, the method is restricted to solving the paths *LD?S*DE* (see §2.10).

The simplicity and elegance of the ray tracing method is only able to provide the solution of specular inter-reflections and direct lighting for diffuse surfaces. The Monte Carlo method manages to extend further the ray tracing method through the use of stochastic sampling in the scene. Thus, MC techniques are able to replicate further phenomena such as indirect lighting, motion blur, depth of field and penumbras.

Distributed ray tracing & path tracing

Cook et al. [17] extended the ray tracing technique through the use of stochastic sampling. For example, aliasing can be eliminated by integrating over the pixel area. Cook et al. state that this phenomena can still be filtered out as with any other analytical methods. Instead of solving the analytical model, the pixel area is sampled multiple times, such that a better approximation of the pixel colour can be worked out. Distributed ray tracing used the concept of over sampling over other domains such that other phenomena can be reproduced. In fact distributed ray tracing does not use more sampling than over sampling techniques, it merely applies them to other domains not necessarily to the spatial domain, such as:

- Out of focus:

Sampling over the lens surface area to produce the out of focus effect

- Motion Blur: When the subject of a photo is captured whilst moving, the end result would produce what is called as motion blur, the phenomena can be reproduced in ray tracing by sampling the time domain.

- Penumbras:
  Shadows in real life are not always sharp shadows but follow a transitional gradient. This effect occurs due to area lights. This would require an integral over the area light's surface. This phenomena is replicated by sampling the surface area.

Similarly, the rendering equation is solved by further sampling over the scene. Upon intersection of the primary ray , multiple secondary rays are emitted such that the lighting contribution of points intersected by the secondary rays is computed. This technique introduces the issue of explosion of samples. If this procedure is left unchecked and repeated, it would result in an exponential growth of the number rays that have been cast in the scene.

Unlike Cook et al. [17], instead of oversampling the point's hemisphere, Kajiya et al. [34] solves the rendering equation by integrating over the pixel area. Furthermore, this technique is known as path tracing.Upon collision, the path tracer casts another single ray per bounce to approximate the indirect lighting. The ray that is cast is chosen according to a *Probability Distribution Function* (PDF). Path tracing does not sample a specific number of rays over the hemisphere, but builds multiple random paths all of which are cast through the pixel. Once a ray interacts with a surface a random direction is chosen and a new secondary ray is cast into the scene. The process is repeated until the path reaches a depth limit or until the amount of contribution provided by further bounces will have negligible contributions to the final result. Kajiya et al. noticed that it is better to focus computation on events that have undergone an amount of reflections. Thus, multiple paths are emitted for each pixel and the respective contributions returned are averaged. Path tracing is

very prone to variance, resulting as noise in the final outputted render. To reduce the amount of noise, a large amount of paths must be sampled per pixel, such that, an accurate estimate of the integral is produced. Specifically to reduce the variance by half, the amount of paths cast for each pixel must squared. As stated by Jensen [33], on average 10,000 path are required to be cast through each pixel to remove the variance completely. Both Cook et al. [17] and Kajiya et al. [34], managed to extend ray tracing further to a global illumination solution while still preserving the basic properties of ray tracing, that is:

- Separate Geometry
- Lighting models
- View dependence for specular effects
- Pixel independence for parallelisation

2.14.4 Irradiance Cache

The Irradiance cache (IC) is a global illumination algorithm that originally provided a solution for diffuse inter-reflections. IC technique adapts and improves the distributed ray tracing (§2.14.3) technique. Accurate diffuse indirect lighting requires the sampling of many rays. However, indirect illumination is a slow changing function, thus, instead of sampling the scene for each request, Ward et al. [63] make use of previously computed samples when possible to compute the current indirect lighting query. Since diffuse reflections are view independent (see §2.5), it is possible to store computed diffuse indirect lighting samples. In fact, it uses a data structure for efficient insertions and retrievals of stored samples. The data structure is then queried for points that reside in a specific part of the scene. If the data structure retrieves points, the current queries indirect lighting value is computed by interpolating over the retrieved IC samples. Thus, the IC provides an on demand solution to solve the rendering equation. Achieving computational speed-

up when possible by interpolating on previously computed samples. Moreover, the number of IC samples produced does not depend on the image frame resolution, thus, the solution for high resolution images can still be delivered efficiently. A summary of the IC algorithm is summarised in Algorithm 1:

Algorithm 1 Get Indirect Lighting

```
1: procedure InsertInNode
2:     pointsFound = FindInterpolationPoints()
3:     if pointsFound.length > 0 then
4:         Interpolate through samples
5:     else
6:         Compute a sample and store the generateed sample
7:         CacheSample
```

Indirect lighting at a point is computed by sampling over the unit hemisphere. Once the values for the rays cast is retrieved, the resulting values are averaged and assigned to the current IC sample. The hemisphere is sampled with a constant amount of rays. The larger the amount of rays used, the more accurate the final render; resulting a render with less variance.

Irradiance gradients

Hemisphere samples give an estimate of the irradiance value at a point. As stated by Krivanek et al. [37], Ward et al. [63] designed an estimator, by which stored samples can be interpolated to calculate the irradiance of neighbouring points without requiring any sampling in the scene. Interpolations are achieved by a technique deviced by the authors called *irradiance gradients*. Irradiance gradients vary both with the rotation across the normals of the sample's surface and the distance of separation between the two points.

Taking into consideration two surfaces x and y, where x is a bright surface., surface x contributes light towards the latter surface. As the bright surface is rotated towards the normal of the other surface, more flux energy is received by surface y. Rotation gradients provide the first order approximation of the irradiance

change due to rotation. Furthermore, the direction of rotation signifies the axis of rotation, while the magnitude specifies how quickly irradiance is changing.

On the other hand, transitional gradients signifies how irradiance is changing as the distance between the point being queried and the sample increases. Therefore, transitional gradient is dependant on the distance to the surface that contributes the indirect lighting. Occlusion and dis-occlusion play an important role for translational gradients. Occlusion and dis-occlusion information is provided by the shortest collision distance of all the hemispherical samples. Given a position over a surface, transitional gradients allow us to approximate the actual change happening in irradiance over the scene geometry.

Interpolations

Efficient interpolation over cached samples depend on efficient storage and retrieval of such points. For caching purposes the octree data structure is used. The octree is data structure that stores object-space samples. It splits the 3D space into 4 smaller voxels recursively on demand. An octree's voxel is re split into a finer-grained octree nodes, if the current IC sample has a radius that is smaller than half the width of the current voxel, then the voxel that intersected with the respective sample is re-split. This recursive process would would terminate when half the voxel's longest side is greater or equal to the current sample's radius of effect or the octree's depth limit is reached. Once the caching data-structure is queried, the data structure would be traversed to find points that are fit for interpolation with the current position. The fitness function that dictates if an IC sample is fit for interpolation purposes is provided in equation 2.28. If the fitness function provides a weight that is larger than 0, then the sample retreived is fit for interpolation purposes. The fitness function depends on the transitional and rotational gradients. Furthermore, the said function also depends on the alpha variable. The alpha variable dictates the amount of error that is allowed. If the alpha value is set to 0, no sample found in the cache would be fit for interpolation purposes. On the other hand, as the alpha

value is increased, more distant samples are acceptable for interpolation purposes. However, this results in a solution that is less accurate and more prone to variance.

$$S(p)\{i; w_i(p) > 0\} \tag{2.27}$$

$$w(p_i) = \frac{1}{\frac{|p-p_i|}{R_i} + \sqrt{1 - (n_i \dot{m}_i)}} - \frac{1}{\alpha} \tag{2.28}$$

$$E(p) = \frac{\sum_{i=0}^{n}(p) w_i(p)}{\sum_{i=0}^{n} w_i(p)} \tag{2.29}$$

2.15 Wait-free irradiance cache

Traditionally, concurrent access to a shared data structures is controlled through the use of critical sections. Critical sections guarantee that only a single thread is updating the shared data at any instance. Atomicity of the critical section is built through the use of locks that block access to other concurrent threads while another thread resides in the critical section. Blocking of threads incur exceptional penalties when the said actions occur frequently.

The lock based IC uses critical sections for the traversal and insertions of new samples in the octree data structure. Moreover, insertions and updates of the octree itself, such as, updates of the tree structure, are done through the use of more blocking mechanisms. Debattista et al. [25] designed an efficient wait-free IC that does not make use of blocking mechanism and still provide guarantees of atomicity. Furthermore, unlike the lock-free counterpart, the wait-free IC allows concurrent threads to interact and update the IC.

2.15.1 Atomic operations

The wait-free IC relies on the use of atomic operations to provide atomicity to for critical operations. Debattista et al. make use of the *fetch and add* (XADD) to

increment a counter atomicly, while, *compare and swap* (CAS) is used to check the status of a variable. If the state of the variable is equal to the value inputted to the function then the instruction would swap the variable's value with the new value, all of which is done in one single instruction. Both instruction's pseudo code are provided in Algorithms 2 and 3.

Algorithm 2 Fetch and add

```
1: procedure XADD(address location)
2:     int value = &location;
3:     *location = value + 1;
4: return value;
```

Algorithm 3 Get Indirect Lighting

```
1: procedure InsertInNode(address locatio, valuer cmpVal, value newVal)
2:     if *location == cmpVal then
3:         *location = newVal;
4:         return true;
5:     else
6:         return false;
```

2.15.2 Wait-free list

The wait-free IC makes use of a scalable wait-free linked list, for concurrent reads and writes purposes. The linked list object is composed of a linked list node pointer names as the *head* and an integer counter. A linked list node is made from an array of fixed size and a pointer that is referred to as the *tail*. Upon insertion the linked list counter is incremented upon each and every insertion that occurs. Since the counter is incremented through the use of the XADD, no threads are assigned the same index. Thus, insertions in the arrays can occur simultaneously, since they concern different array locations.

If the current insertion exceeds the current node's array limit, the linked list requires to be extended further. Thus, the respective thread would be required to create a new linked-list node. Through the use of the CAS, the tail of the last

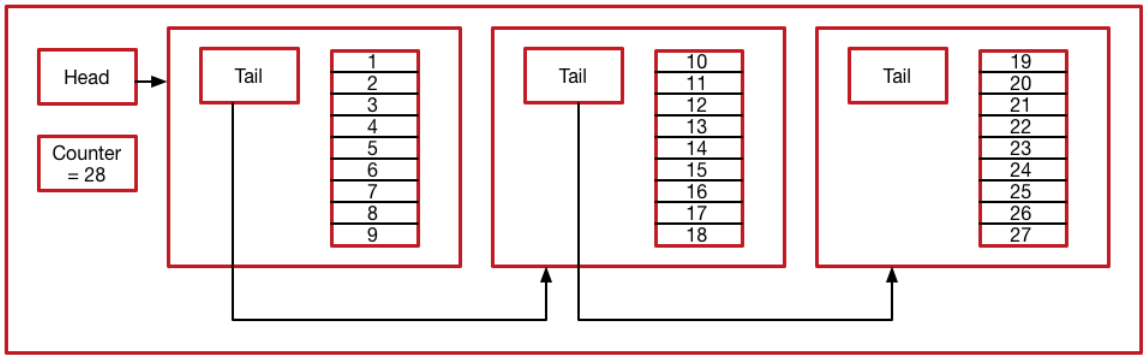


Figure 2.6: The structure of the wait-free list, the head points to the root node of the list and an atomic counter represents the next index.

linked list node is checked if it still equal to null. If null, then the CAS would assign the pointer to the new linked list node to the pointer and the new sample insertion occurs. On the other hand, if the CAS does not succeed, then it would mean that another thread has already created and bound a new instance to the last linked list node. Thus, the new instance is deleted and the process is continued on the already allocated linked list node instance. The index provided by the global counter, provides a unique global index for each sample. Thus, the respective global index must be changed to local index and local block number, before the sample can be inserted in the wait-free linked list. The block number specifies the linked list node number, while the local index represents the array index within a specific node. The local index is computed through the use of equation 2.30 while the block number is computed through the use of equation 2.31.

$$localIndex = globalIndex \,\%\, ARRAYLENGTH \tag{2.30}$$

$$blockNumber = globalIndex \,/\, ARRAYLENGTH \tag{2.31}$$

2.15.3 Wait-free octree

The wait-free linked list provides us with the tool through which points and indices can be stored, without requiring any locks on critical sections. The wait-free IC stores samples in a centralised list, while the respective insertion index is stored in the octree. Similar to the linked-list the wait-free octree makes use of the CAS to alter the structure of the tree. The octree nodes are altered and split into further smaller voxels, per on demand basis. A point is only inserted in a voxel if the minimum ray length is equal to half the longest axis of the voxel or the maximum depth limit was reached. If the children of a voxel have not yet been initialized, the thread would create a temporary voxels for the areas with which the sample has intersected. Once created, using the CAS operation, each voxel is compared to check that it is still equal to null and swapped if it is still unchanged. However, if the pointer is not equal to null, the temporary voxel is deleted and recursively visit the sub-child.

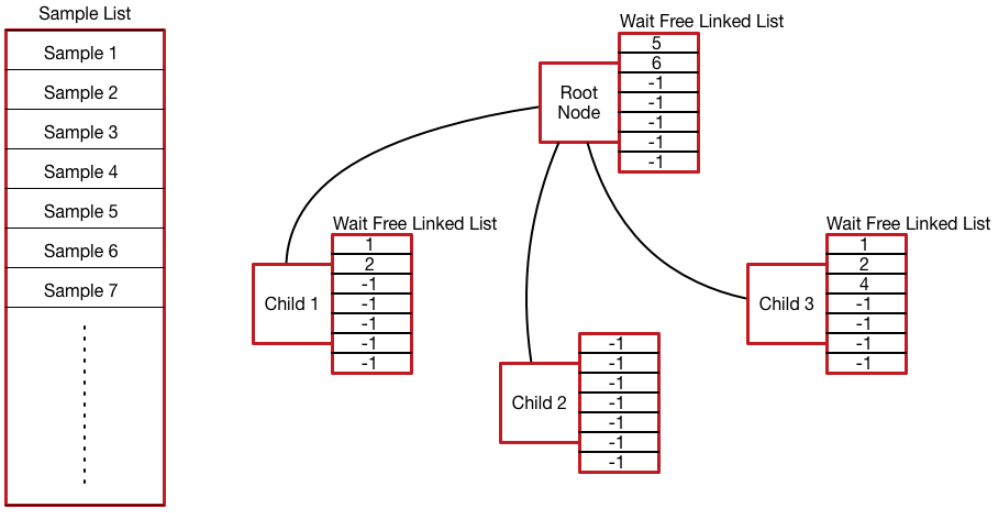


Figure 2.7: Samples are only stored in a central list, while the nodes store the respective indices.

2.16 Conclusion

This chapter has introduced and explained the mathematical background for rendering. All existing types of materials were introduced along with their respective mathematical model for the reflection of light. The rendering equation was explained, and its three most common forms were detailed. Monte Carlo integration was summarised, and Heckbert's notation was described. An overview of ray-tracing techniques was given. The chapter was then concluded by detailing the irradiance cache, a technique used for diffuse-diffuse inter-reflections and the wait-free Irradiance cache was summarised.

3. Literature Review

In this chapter, work related to distributed rendering and data dissemination over peer to peer systems is compared and contrasted. Distributed rendering techniques will be examined from the fields of parallel rendering, distributed rendering, and large scale distributed rendering. Furthermore, data dissemination methods will be compared from the fields of one layer peer to peer systems to multi-tier peer to peer systems.

The chapter is organised as follows: Section 1 introduces distributed rendering and peer to peer systems, while Section 2 compares distributed rendering techniques and different peer to peer systems. Section 3 lists a number of areas that require further investigation and Section 4 concludes the chapter.

3.1 Overview

Rendering is a computationally expensive technique. However, it is also an embarrassingly parallel problem. To speed up the performance of ray tracing, parallelisation may be applied.This can be achieved through the use of specialised hardware, such as the GPU or through the use of multiple CPUs residing on the same computing unit. This paradigm is referred to as parallel rendering. Furthermore, parallelisation can be extended further to make use of multiple machines; a process that is known as distributed rendering. Moreover, there exist systems that are classified as a hybrid of both, where the system makes use of distributed rendering while containing more than one processing element (PE) on each physical machine. Since each PE is required to work on a subset of the problem, at some point during the rendering process, the PEs are required to synchronise. The subdivision of the problem is carried out by a special PE called the master, while the solution is worked out by a PE known as the worker. After all the subtasks have been processed, the master would then gather all the results and compose the final solution. Peer-to-peer (P2P) is a paradigm in which there is no master PE; moreover, all PEs are equal and autonomous. Several propagation and connectivity strucured over P2P networks are used to disseminate information which may be required or that might be beneficial in the future for the respective peers. If computations computed by a PE can be shared with other PEs, the respective PEs might use the received computations without the need to compute them themselves and thus achieving a speed-up.

Parallel and distributed rendering require a layer through which PEs can communicate. If the PEs reside on the same computing unit, then the PEs would communicate through the use of shared memory. On the other hand, distributed PEs must use some other form of communication channel since each computing element has dedicated memory and backing storage. In these cases, the PEs require a common communication backbone through which messages and data can be shared, such as, a network or the internet.

Rendering can be subdivided into two different fields namely offline non-interactive rendering and its counterpart interactive rendering. In Offline rendering there is no user interactivity and every action happening in the scene is pre-defined, such as animated movies. On the other hand, the latter rendering paradigm allows the user to interact with the scene and change the parameters that govern the scene. The user then can get real-time feedback of the changes done to the scene. Interactive rendering introduces further timing constraints to the rendering process since frames are required to be rendered sequentially; Furthermore, each frame is required to be rendered within a specific time limit such that the system adheres to a specific frame rate.

Distribution of tasks is key for load balancing of the work between the corresponding PEs. The most obvious task subdivision is to subdivide the problem into a set of disjoint tasks that can be assigned to specific PEs [50]. Coarser sub-division will keep each PE busier, however, once a PE finishes a task and the job queue residing on the master process is empty, the PE will end up in an idle state. A PE can also transition into the idle state where the PE waits for some form of synchronisation to get data from another PE or the master process. On the other hand, smaller tasks would result in less work being processed on a PE and more time is wasted for communication ans synchronisation.

3.1.1 Parallel rendering

Many techniques have been proposed that enhance the performance of the ray tracing technique through the use of parallelisation; this work was then extended further to improve the performance of high-fidelity rendering [16]. Shared memory parallel processing has the advantage of providing the PEs with a common shared memory [46]. Such architectures provide the means to share geometry and data, such as the scene data. On the other hand, it can also slow down the performance of the PEs, since the shared memory could become the bottleneck of the system. Koholka et al. [36] could not rely on the use of shared memory architecture since

the system was implemented to sustain distributed memory architectures. Thus, each PE is required to load the scene data into its local memory. Moreover, they proposed a system for an offline rendering that made use of the IC to deliver offline GI. The system used a demand driven on per frame basis approach, where a frame is subdivided into a number of tasks and new jobs are not submitted to the job queue until all the workers have finished computing the tasks for the current frame. Irradiance is independent of the viewing direction, thus peers update other workers with computed IC samples by broadcasting them. To prevent excessive amount of network traffic, samples are batched samples in groups of 50 samples and broadcasted them to other PEs.

Unlike Koholka et al., Muus et al. [43] make use of a central file server to store complex scene geometry data that sums up to 1 million polygons. This approach introduces a global dedicated storage space that workers can use on demand. Although efficient, this approach can also impose severe inefficiencies as the system scales, since traffic towards the file server would also increase. Muus et al. provide a system for interactive rendering purposes, in fact, this imposes further constraints since the system was made to adhere to a frame rate of 5 frames per second (fps). In a similar fashion to Koholka et al.'s approach, the system still distributed jobs on a per frame basis. Muus states that updates to the frame buffer are required to be tightly synchronised with the rendering process as not to deliver temporal artefacts. If a worker is not able to deliver the solution for a particular tile in time, the frame buffer would not update the related pixels, resulting in pixels containing data from previous frame. For this reason, Muus et al. introduce an incremental rendering approach, where the final resolution of the frame is reached through an incremental and recursive method. If a worker fails to deliver the render task in time, the resulting render would contain a part of the rendered image rendered at a lesser resolution. Through the use of 8 nodes containing 12 processors each, the system was capable of delivering interactive rendering over geometries spanning 1 million polygons. Unlike Koholka et al., Muus et al. did not implement GI but

only direct illumination.

This approach was extended further by Parker et al. [46], who used frameless rendering to deliver a modified Whitted-style ray tracing [64]. As opposed to Muus the frame buffer updates are carried out in an asynchronous manner to the rendering process. Thus, the frame buffer is updated periodically. Moreover, the method makes use of Hilbert's 1D curve to split the image-space into a set of tasks, that are distributed in a round-robin fashion to the workers. The load balancing procedure used is similar to Koholka et al's approach, where the job's size decreases linearly as it approached the end of the frame. Moreover, the system makes use of shared memory that is used for the loading of the scene data.The ray/scene interactions are optimised through the use of grid-based subdivision, coupled with bounding volumes. Parker et al. achieved interactive Whitted-style ray tracing through the use of 48 processors. Additional PEs introduce a slight drop off in performance.

Parallel interactive rendering was extended further by Wald et al. [60] which improves on the work done by Wald et al. [62] through the application of out of core rendering for very highly complex models, while still using commodity hardware. This was achieved through a custom-built memory management scheme and by a pre-processing method to build a system of proxies and a binary space partitioning (BSP) tree representation of a mesh. Proxies are a coarse grained approximate representation of subtree. When a required subtree is not currently loaded in memory, instead of halting the progress of the current ray until the polygons that are part of the BSP subtree are loaded, ray intersection is carried out against the respective proxy.

Custom memory management allows Wald et al. to detect and avoid page faults; this is done through the use of a *tile table* that marks what pages are available in memory. Since the hardware that the system is running on is a 64-bit system, keeping track of all the pages would require 2^{52} entries. Thus, pages are joined into *tiles* and stored in a hash table of tile addresses. Tiles that are marked as

not available are required to be loaded from memory; this job is assigned to *fetch threads*, which take missing requested tiles and load them into memory. The fetch threads only fetch required data, on the other hand, pages that are not being used need to be removed. For this purpose, an eviction procedure is used so that unused pages are removed from memory. Every tile has a usability bit assigned, which is set to 1 when used by a thread. The *evictor* cycles continuously through loaded tiles and resets their used bit to 0; if rays require a specific tile they would switch R-bit back to 1. Moreover, the eviction process removes encountered tiles that have their corresponding R-bit already set to 0. Using a single dual-Opteron PC, Wald et al. were capable of delivering frame rates of 3 - 7 fps on a resolution of 640 by 640 pixels. Scaling the resolution to 1024 by 1024 degraded the performance to 1 - 2 fps.

3.2 Distributed rendering

If no specialised hardware is available, clusters of commodity hardware are used for improved rendering performance [50]. As opposed to parallel systems, distributed systems do not make use of shared memory, and thus, any required data must be loaded by the respective element into its local memory. Since distributed systems do not have shared memory; scene data must either be loaded on each PE or distributed evenly between the PEs. Moreover, distributed systems require a robust communication backbone through which synchronisation messages and data can be exchanged. Aggarwal et al. [10] introduced the concept of fault-tolerance through the use of quasi-random sampling over interactive high-fidelity rendering. As stated by Muus et al. [43], a coarse-grained subdivision of the frame space is not fault-tolerant. For this reason, Aggarwal et al. [10] subdivide the frame through the use of quasi-random sampling over the frame's pixels. If a worker fails to deliver a result in time, the frame would still be of adequate quality. Furthermore, missing pixels are then reconstructed in the image reconstruction process. This process

would not be possible to reproduce on a frame that has been coarsely divided into tiles. The image reconstruction process is executed on the GPU and fed back to the master process for display. Both systems implement a modified on-demand master/worker structure. Aggarwal et al.'s system makes use of time constraints for frame renders, that are issued by the master process. If a worker fails to deliver a result in time, the same task is issued back on the job queue if enough time is left. Furthermore, the worker is removed from the worker's list. The deadline constraint is a variable constraint basing itself on the ratio of tasks that have been solved to the number of tasks issued on the job queue. This idea is extended further by the same authors for time-constrained animations [11]. Instead of sampling over a single frame, jobs are a collection of samples from different frames. Moreover, the post-image reconstruction process does not feed to the master process, since the resulting frames would require a lot of local memory. Instead, the frames are saved to an out-of-core storage procedure through the use of memory mapped files. Unlike Koholka et al. [36], the workers never communicate with each other. The path tracing technique is an image space technique, thus, view-dependant [34] and computations computed are not sharable.

Unlike Aggarwal et al., Wald et al. [61], deliver interactive high fidelity rendering through the use of the *instant radiosity* [35] and rendering of caustics through the use of the photon mapping [32] techniques. The instant radiosity basis itself on virtual point lights (VPL), which are produced by tracing rays from the light source across the scene, upon collision. The direct light contribution for each intersection point is computed and saved. Once enough VPLs have been traced in the scene, the rendering process is commenced, the direct and indirect components are computed in a similar manner but for the latter instead of using the light source for the scene; the generated VPLs are used. Caustics are visualised through the use of a simplified version of the photon mapping technique, since the original technique is too slow for storing and retrieving at interactive rates. Through the use of the generated VPLs as photons, caustic phenomena can be identified when the VPLs

density is high.

Using the same VPLs for all the pixels results in aliasing, however, using new VPLs for each and every pixel will provide an inefficient technique that is not adequate for interactive rendering. Thus, Wald et al. generalised the interleaved sampling procedure that was proposed by Molnar et al. [**?**] and through the use of a number of different VPL sets, they provide a render with adequate quality. Moreover, since interleaved sampling achieves a better visual quality, the number of VPLs used can be lowered than norm. Interleaved sampling can still produce artefacts on the final render, thus interleaved sampling is combined with the discontinuity buffer to provide local smoothness to the render.

3.3 Hybrid rendering systems

Hybrid systems make use of both distributed and parallel processing to achieve better performance. Debattista et al. [22] applied the method of component-based rendering through an on-demand master/worker system. Workers are divided into two subgroups that are referred to as the PA and PR groups. Every PE of both subgroups makes use of an IC. Yet, the former group only computes the lighting component until there is no computation of indirect lighting required, on the other hand, the latter group only computes indirect lighting requested by the former group. Koholka et al.'s system would theoretically benefit if broadcast speeds are increased, yet, it would not scale so well since communication traffic is increased and thus, communication would become more costly. On the other hand, Debattista et al.'s system contains only a subset of the system broadcasting, specifically, the PA group. Thus, the communication frequency can be increased without an issue of network traffic. Samples computed by the PA group are both shared with the master process and fellow PA group workers alike. When elements of PR synchronise with the master process for new jobs they also sync the IC cache. The PAs communicate through the use of shared memory. Moreover, PAs never send

solutions back to the PRs, but the composition of the final solution is left to the master process. Once all the tasks of the job queue have been exhausted, the master-process enters a load balancing phase. As the PA members are communicating with the master-process they also synchronise their status and the respective amount of requests pending to compute. Once, a member of the PR group syncs for new jobs the master process would send a message to a PA to offload some of its computational load to the respective PR member.

Similar to the work of Koholka et al, Debatista et al's system is intended for offline purposes. On the other hand, De Marle et al. [23] and Wald et al. [62] exploit hybrid rendering systems to deliver interactive rendering of complex geometries. Both systems make use of a master/worker paradigm. However, in De Marle et al's system the geometry is split evenly between all the running PEs. In the case that a voxel is required and not residing in the local memory, the respective PE would request the voxel from the PE containing the voxel and load it in memory. The procedure was found to be more efficient than loading elements from the backing store since it may be more time consuming. In contrast to De Marle's approach, Wald et al. build a pre-processed BSP; if specific voxels are not loaded in memory, rays intersecting the voxels are paused and an asynchronous process loads the polygons from the network file server, an approach that was also followed by Muus et al. [43]. Once loaded, all rays requiring the respective voxel can be resumed again. Both systems make use of multiple threads and for added performance Wald et al. makes use of shared memory. Thus, voxel data loaded by one thread is stored in the shared memory and visible to the other threads and vice-versa. Wald et al. relies on a last recently used method to remove unused voxels from memory. Similarly, De Marle et al. applies a semaphore to each component that was loaded from network to make sure no received data is removed while a thread is using the respective datum. De Marle et al. managed a maximum of 23 fps while using 30 PEs. On the other hand, Wald et al. using 7 PEs achieved 3-5 fps when the PEs were not SIMD optimised. Performance increases to 6-12 fps when SIMD

optimisations are applied.

3.4 Large scale distributed rendering

As stated by Bugeja et al. [14], distributed rendering extends to the use of the grid, through the use of algorithms aimed at the sharing of resources and data. The Big Ugly Rendering Project (BURP-BOINC), is a global project for the use of domestic hardware as free rendering computational power. The BURP project requires a good communication infrastructure through which available machines can be detected and exploited. The BURP-BOINC [7] project relies on the BOINC [12] for these services, furthermore, BOINC would then use CPU cycles on the commodity for rendering projects. A similar project is presented by Patoli et al. [47], allowing Blender [6] projects to be rendered on a powerful grid of computers. Similar to the BURP-BOINC project, Patoli et al.'s system relies on the HTConodor [56] as a hardware management system for the detection and message passing of tasks and required data. Unlike BOINC, HTCondor only makes use of commodity hardware that is in an idle state.

Both BOINC and HTCondor make use of the master/worker paradigm. Thus, both communication layers are dependent on a singular entity that serves as the master process. In the case that the master process stops working the whole infrastructure would malfunction. On the other hand, Matteo et al.'s system uis a decentralised infrastructure for voluntary computing specifically built for rendering purposes. Unlike BOINC and HTCondor, Yafrid-NG was designed specifically for distributed large scale rendering to be able to dynamically scale, in contrast to the two counterparts that offer a general solution for distributed systems, which have been applied for rendering purposes. Furthermore, the BURP-BOINC project does not support fine-grained tasks [48]. The system is built on two types of users, consisting of the *providers* and the *clients*. Once a client submits a job, the respective peer would manage the whole rendering process. Yafrid-NG is not completely

decentralised, in fact, it requires two services to be run using the master/worker paradigm. The two services are the *SessionManager* and the *NodeManager*. The former service keeps track of established sessions over the peer to peer system while the latter keeps track of all the available nodes for rendering.

The systems that have been mentioned so far, rely on the master/worker paradigm to deliver non-interactive and interactive rendering. Most systems rely on the master to manage the rendering process. On the other hand Crassin et al. [19] provide a system that makes use of the cloud to deliver rendering capabilities to connected devices. The system called *CloudLight* follows a client/server paradigm for clients that are interacting together in the same scene. Through the use of a very powerful server-side hardware that mimics the cloud, the system is capable of delivering GI to tablets, low-power PCs and powerful PCs alike. The server/client approach allows decomposing the rendering process into a two part technique. Computationally intensive operations are computed remotely on the cloud/server, while lightweight tasks are computed on the client side. The client side then syncs the computed with the results received from the cloud. The server/client approach is highly dependent on the internet connection and bandwidth available to the respective devices. Different tiers of devices are limited by the internet connection they can employ. Crassin et al. make use of three different techniques to deliver GI to the different classes of devices. Voxel cone tracing [20] is used for low-end devices such as tablets and mobiles, Irradiance maps [42] are used for mid-range devices while photon mapping [32] is used by high-end devices. Wald et al. [61] had stated that for interactive purposes amortisation is not possible since feedback must be given in few milliseconds. Yet the centralised cloud-based approach allows Crassin et al. to deliver dynamic interactive GI by exploiting previously computed data when possible. The server component's hardware system is tailor made to sustain the two stage GPU technique. The first GPU is in charge of the rendering process while the latter compresses the data that will be sent to the client device. Both Irradiance maps and Photon mapping techniques are amortizable since both techniques use or

require object-space data. On the other hand, voxel cone tracing is not amortizable since each frame is view dependant. Moreover, the voxel cone tracing technique requires that the rendering process is all done on the server side since it is not feasible to send voxel data to the client devices. Thus, this technique is processed on the server side; the rendered frames are encoded by the secondary GPU and the client device just decodes the frame and streams it. This technique requires the client side device able to decode H.264 and requires a bandwidth similar to voice over IP (VoIP). Although it appears as if updates are being rendered synchronously, updates are occurring in asynchronous fashion on the server side. The irradiance mapping technique divides the indirect lighting calculations to the server while the client is required to compute the direct lighting and compose it with the indirect lighting received in the form of mesh textures. Thus, this technique requires a device of mid-range power, to be able to render direct lighting and decode H.264 textures. Moreover, the technique is quite efficient since it only requires enough bandwidth to recieve scene textures. However, this technique is restricted since UV parameterization is often times difficult to compute [13]. Photon mapping is heavily dependent on the client side's machine computational power.The photon mapping technique sends the clients photon batches that have been emitted from the scene, further updates that would be received from the client are batches of photos that have been refined or invalidates. On the client side, the device must be capable of reconstructing the photons received and compose the indirect lighting with the direct lighting component of the scene.

Peer to peer systems provide a system that is both scalable and fault tolerant. Although Mateos et al. borrow P2P concepts to provide resilience and fault tolerance, their system still contains singular centralised components and thus their system falls under the hybrid peer to peer techniques. Similarly, Crassin et al's approach is rather centralised, providing remote computational power through which devices with less computational power are able to compute GI. Bugeja et al. [15] provide an interactive and dynamic solution for large scale distributed rendering.

Similar to Mateos et al., Bugeja et al. make use of a peer to peer system to deliver interactive HFR. Following on Koholka et al.'s and Debattista et al.'s approach, Bugeja et al. make use of the IC [25]. Taking into consideration the union of all the local IC caches of the peers collaborating in the same scene would lead to a global cache containing all the updates, allowing a peer to use the network received samples and only interpolate instead of sampling the scene. Assuming multiple peers are interacting in the same scene, computations computed by one peer can be used by another peer in the future. Through the use of epidemiological techniques, updates created by a peer traverse the network until every peer has cached them. However, epidemiological techniques [27][24] can only guarantee that at some point all peers will receive the data. In a similar manner to Koholka et al. [36] and Debatista et al. [22], Bugeja et al. batch updates in in an object called observable events. Bugeja et al. batch the samples in batches of 100 samples. Moreover, each observable event contains a unique identifier and a vector clock [57] stamp of the producer's current vector clock state. The vector clock stamp is used to produce a deterministic global order of all the observable events that have been exchanged. If the order of two observable events cannot be ordered because the two vector clock instances result as equal, then a tie breaker function is used that depends on the *UUID* of the two observable events. Global order of the events allows the system to deal with dynamic state changes that can take place in the scene, such as the triggering of a point light or the movement of the sun. Thus, updates are required to be ordered in a globally deterministic manner. In order to determine a sequential time line of event occurrences. This allows the system to deal with other events that do not necessarily provide an update to the IC cache, such as, invalidations of the cache or updates to the scene geometry. Unlike Crassin et al.'s system, the said system cannot handle critical events since updates are sent in such a way that may not retain relevance until they reach all the peers that might require the update.

3.5 Data Dissemination Techniques over P2P Systems

Peer to peer systems are used in scenarios where multiple servers or workstations are needed to keep each other in a consistent state. Peer to peer systems can be divided into two main classes; the unstructured and the structure topology networks [55]. Structured peer to peer systems tend to make use of distributed hash table(DHT) techniques to map object identifiers to distributed nodes. DHT techniques are a class of decentralised algorithms that offer efficient lookup services. On the other hand, unstructured peer to peer systems don't make use of a structured approach for communication purposes, providing an ad-hoc network of peers [45]. As explained by Tarkoma [55], unstructured peer to peer systems make use of random walks and similar opportunistic techniques to disseminate information throughout the network.

The most simple message passing technique for resource location and resource discovery is the use of the flooding technique [55]. On the other hand, peers may use a broadcast technique to send updates to all known peers. However, this technique is only viable when all the peers are already discovered. Demers et al. [24] exploit epidemiological techniques through the use of the anti-entropy technique, to solve a long-standing problem of high traffic and database inconsistency. Epidemic techniques are a class of random techniques that simulate the process by which diseases spread. Moreover, the said techniques make use of the small world paradigm, where each peer never caches the details of all the peers found on the network, but a selected number of neighbours. This phenomenon allows the network to stabilise and allows updates to be propagated in a balanced manner.

Similar to Demers et al., Bugeja et al., apply the anti-entropy technique to disseminate IC information between multiple peers that are interacting together in the same scene. Anti-entropy is a useful technique since it can guarantee that at some point all the peers will receive an update that a peer wishes to spread. An

issue that epidemic techniques contain is the fact that updates are only guaranteed to be disseminated across the network at some point, thus updates spread are never directed to a specific peer. Moreover, both systems are considered to be unstructured peer to peer systems.

Napster [55] extended the peer to peer paradigm to create a hybrid technique in which file exchange took place over the peer to peer network, while it also made use of a centralised directory that contains the file lists of all the peers that are contributing to the network. The motivation behind this decision is that it does not require much bandwidth to transfer file lists [55]. Thus, peers would then query the file server for details about a specific file. If the file server finds peers for the said query, the server would reply to the client with the details. Once the peer has acquired the details, the querying peer would then be able to communicate for the file directly with the peers. On the other hand, Gnutella [55] takes a completely unstructured approach towards peer to peer networks. The system makes use of flooding to find peers with matching data items to the query, once found the peer uses direct file exchanges. The protocol was improved further through the introduction of hubs that are referred to as *ultra nodes*, while peers that are only able to service and request file exchanges are referred to as leaf nodes. The ultra nodes introduced a two-tier level peer to peer network. Moreover, their purpose is to serve as connection hubs, where each ultra node is connected to at least 32 other nodes. Furthermore, the ultra node network is an unstructured and flat network. Each leaf node is connected to at least 3 ultra nodes and as stated by Tarkoma et al. [55], the change in structure was done to implement the small world phenomena and thus evenly balance the whole network.

The approach of Gnutella was extended further in [45], where Papdaski et al. reduce the amount of flooding that occurs between the super peers. An ultra node on Gnutella might receive multiple copies of the same requests through the use of flooding. This results in more traffic on the communication network. Thus, Papadakis et al. improve the performance of Gnutella through structured partitioning

of the super peers in different layers. Each ultrapeer in the system is assigned a single category randomly, thus, ultrapeers that have the category form a subnetwork and a leaf peer connects to multiple ultrapeers that are found in the same cluster.

Gnutella introduces the concept of multiple types of nodes in the network structure as to improve the search and retrieval of data while still providing a scalable, fault tolerant and decentralised solution. Unlike unstructured networks, structured networks constrain the peers to operate through a topology; similar systems are Chord, Tapestry and Pastry. All of which rely on a distributed hash table to route messages through the peer to peer network [54][65][51]. However, systems mentioned above are only used for routeing purposes. Thus such systems try to optimise the amount of hops required to reach each and every peer found on the network.

Data being exchanged between peers over the network can interest multiple peers or none at all. Papadaski et al, improved Gnutella by subdividing the ultrapeers into different subgroups as to reduce redundant message passing. The publisher/subscriber paradigm is used to group peers that share temporarily or permanently similar interests. Thus, messages in between peers found in the same group are found at a 1-hop distance from the respective publisher. There exist two types of publisher/subscriber paradigms; *topic-based* and *context-based*. The former paradigm's groups are constructed on a set of pre-specified topics while the former paradigm's groups are formed by arbitrary predicates on attributes [59]. Although this approach provides an efficient dissemination of data, since all peers get data that is related to them in 1-hop, it still restricts its scalability. Sub-2-Sub(S2S) [59] overcame this issues by utilising a publisher/subscriber method. Moreover, the system is autonomous and self-organizing. Autonomous implies that dissemination of events to all interested nodes is accomplished by the nodes involved in the respective network, whilst the self-organizing. nature enables to organise themselves to enable cooperation for event dissemination. Each new group creates their re-

spective logical layer, where each layer is organised in an unstructured manner. In each layer the updates are disseminated through the use of epidemic techniques. Each subscriber can be part of multiple layers and the base layer for the S2S system is an unstructured network of peers. The peers periodically exchange information to discover similar peers to form clusters with. Peers that are temporarily making part of a particular layer self-organize into a bidirectional ring. Thus new updates are disseminated across the ring topology in linear time. New updates are not only disseminated over the current cluster but through two other protocols. Voulgaris et al. make use of three protocols to propagate updates in an efficient manner while still maintaining a balanced P2P system. Cyclon [58] is deployed on the base layer, this is in charge of periodically exchange neighbour views with another random peer. Moreover, S2S makes use of the Vicinity protocol which keeps track of peers that have intersecting attributes. Therefore, they might be interested in updates disseminated or passed on by the respective peer. The last protocol used for the exchange of information is the bidirectional ring topology through which updates are sent to the neighbours of the said peer.

Publishers and subscribers never tend to change roles, yet, Datta et al. [21] introduce the concept of *non-sink nodes* and *sink nodes*. Sink nodes are peers that are allowed to diffuse information over the network whilst non-sink nodes are the subscribers listening for information. Unlike the approach of Voulgaris et al., all nodes are given a chance to serve as a publisher. Moreover, peers forming a cluster are arranged as a weakly connected graph that is also a directly acyclic graph (DAG). The DAG graph allows the cluster to determine which peers will be the next sink nodes.

3.6 Conclusion

This chapter presented a literature review on distributed and parallel rendering. Furthermore, data dissemination techniques over peer-to-peer systems where anal-

ysed in terms of their respective topology and the technique governing the data dissemination.

4. Peer-to-Peer High-Fidelity Rendering

Amortised rendering over P2P systems has been investigated by Bugeja et al. [15] through the use of the epidemiological techniques to disseminate updates in an unstructured P2P system. IC samples computed by peers are shared with other collaborating peers. Bugeja et al. argued that as the number of peers participating over the P2P system increases, so would the number of samples that are exchanged over the network. Due to the number of samples that would be exchanged over the P2P system the IC cache of each peer would fall under a state of over-saturation due to the sheer number of samples received. Over saturation is caused by redundant samples that slow down the interpolation process. Interpolation is the process of using already computed samples for the approximation of a new samples, hence providing a speed up since it is more efficient than sampling the scene. However, if too many samples are eligible for interpolations, then the interpolation process becomes inefficient and the speed up is lost. This chapter examine the relationship between the scaling amortised rendering systems and the amount of samples that are shared over the network.

This chapter is organised as follows: The introduction provides a background to amortised rendering, while Section 4.2 presents a methodology for the experiments together with an overview of the rendering system built. Section 4.3 tabulates and

explains the results of the carried out experiments. this is followed by Section 4.4 that discusses the outcomes of the experiments. Finally the chapter is summarised in Section 4.5.

4.1 Introduction

Distributed and parallel rendering systems traditionally make use of the master/worker paradigm to deliver non-interactive ray tracing (see §3.1.1).This paradigm was later applied to support high-fidelity rendering at interactive levels (see §4). Crassin et al. [19] made use of the client/server paradigm to provide GI capabilities to a wide range of devices. This is achieved by dividing the computational concern between the client and the server. Thus, computationally expensive operations are executed on a remote server and sent back to the respective client. The client would compose the computed and received components and produces the final render. Crassin et al.'s approach provided 3 techniques aimed at three different tiers of devices: mobiles and tablets using the using the voxel cone tracing technique [20], medium power devices using irradiance maps [42] and high-end computing machines making use of image space photon mapping [41].

Computationally weak devices streamed fully rendered frames from the server, while mid-level devices received indirect-lighting in the form of textures. On the other hand, powerful machines where provided with batches of photons (see §3.4). This technique did not scale well once the number of clients increased. As the number of clients was increased to 30 the frame rate dropped from 30 fps (for 5 clients) to 12 fps. Moreover, the technique intended for mid-level devices could not be applied to complex scenes given the inherit difficulty to UV parametrize [13]. High-end devices capable of rendering GI through the use of Photon Mapping, still required a very large bandwidth (up to 43Mbps).

Bugeja et al.'s approach lies on the other end of the spectrum, since it is fully decentralised. This approach provides fault-tolerance, since all peers operate au-

tonomously. The speed-up gained by each peer is achieved through amortisation of computation, where previous computations done by other peers can be exploited to achieve a speedup. The authors apply ammortised rendering through the use of the IC. The peers involved, share computed samples. Thus, interpolating over samples received from the network costs a fraction when compared to the time it takes to sample the scene and compute a sample. Since the samples stored in the IC are irradiance samples and unlike radiance samples they are not affected by the viewing angle, they can be shared freely. Albeit decentralised, Bugeja et al.'s system faces an issue of scalability. Two peers working in parallel might produce redundant samples that would still be disseminated across the network. As the number of peers interacting over the network increases, so would the number of redundant samples computed. This issue leads to an over-saturation of the IC residing on each peer and degrades the performance of the interpolation process.

This issue will be investigated by extending further the tests carried out by the authors: running eight peers collaborating over the network, the use of two online gaming standard scenes and the use of 3 different collaboration scenarios. The first test scenario consisted of having all the peers starting to collaborate at once, while the others introduce a delay between the peers starting time, 60 and 120 seconds accordingly.

4.2 Methodology

The collaborative peer-to-peer rendering process used in this work is composed from two main components: the *rendering layer* and the *communication layer*, as provided in Figure 4.1. The rendering layer is responsible for the high fidelity synthesis of virtual environments while the communication layer serializes computations computed by the rendering layer and shares it with other peers for collaborative rendering. Speed-up of computation is gained from the usage of points that are retrieved from over the network, allowing the rendering layer to leverage

retrieved computations without requiring to be computed by the rendering layer.

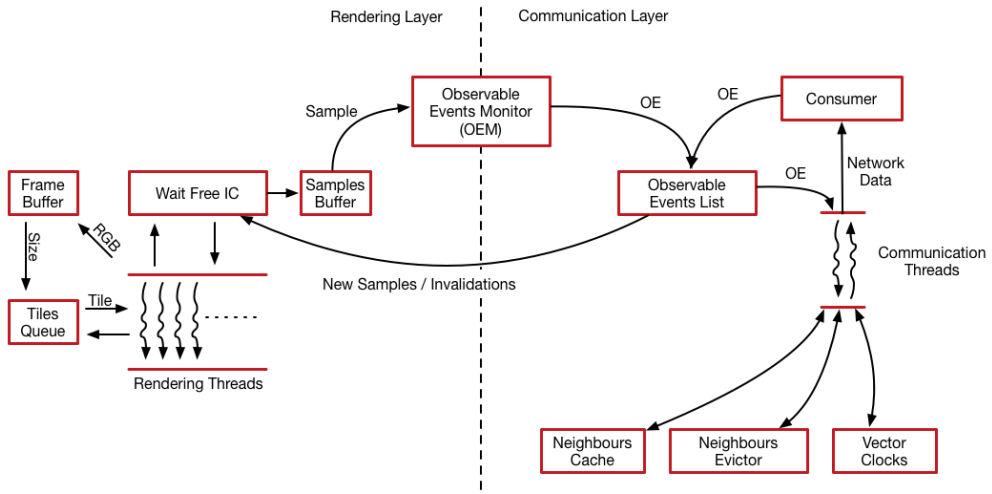


Figure 4.1: High-level overview of the rendering system.

4.2.1 Rendering layer

The rendering layer provides high-fidelity rendering with a diffuse indirect lighting solution that makes use of the wait-free IC, for efficient multi-threading. The wait-free IC guarantees atomicity on write operations while not requiring any form of synchronisation. Parallelism is achieved through the bag of tasks algorithm, using image-space subdivision into tiles of equal dimensions.

In the nomenclature of the original work, computed samples are considered as internal events. When some property is satisfied, these internal events are batched together to create an observable event that is able to modify the global state. The entity responsible for examining internal events and batching them into observable events is the *observable events monitor*. The observable events monitor, periodically monitors the sample queue for new samples. Once found, the samples are retrieved from the queue, parsed into OEs and pushed on the *observable events list*. The observable events list is the structure that contains all the OEs produced by

the system, along with all the OEs that were retrieved from the network.

4.2.2 Communication layer

The communication layer does not commit received updates to the rendering process, instead updates are pushed on a temporary queue and processed in a deferred fashion by the *observable events consumer*. This method avoids blocking of communication with other peers. The communication layer never modifies the state of the observable events list, but only copies its current state such that it is shared with other peers. Modifications to the list are only executed by the observable events consumer and the observable events monitor.

The communication layer does not commit received updates to the rendering process. Instead a producer/consumer paradigm is used. Received updates are pushed onto a queue which is in turn ocnsumed by the observable events consumer. This approach prevents the communication layer from blocking to commit received updates to the IC.

The peer's neighbours are stored in a list that is ordered in descending order according to the number of times a successful exchange has occurred with them. Changes to the neighbours list are never executed instantaneously but a checkpoint system is used to delete neighbours at a safe points during the executing of the anti-entropy protocol, as shown in Figure 4.2.

4.2.3 Experimental Setup

The experimental setup is comprised of 17 virtual machines running on the Microsoft Azure infrastructure. In order to provide a controlled testbed for the experiment, all virtual machines are homogeneous, created within the standard A3 azure tier (see Table 4.1). In particular, each machine guarantees 4 CPU cores with simultaneous multithreading (SMT) running at a clock speed of 2.4 GHz and 7 Gb of RAM. In terms of function, 16 VMs were used to simulate clients in the P2P

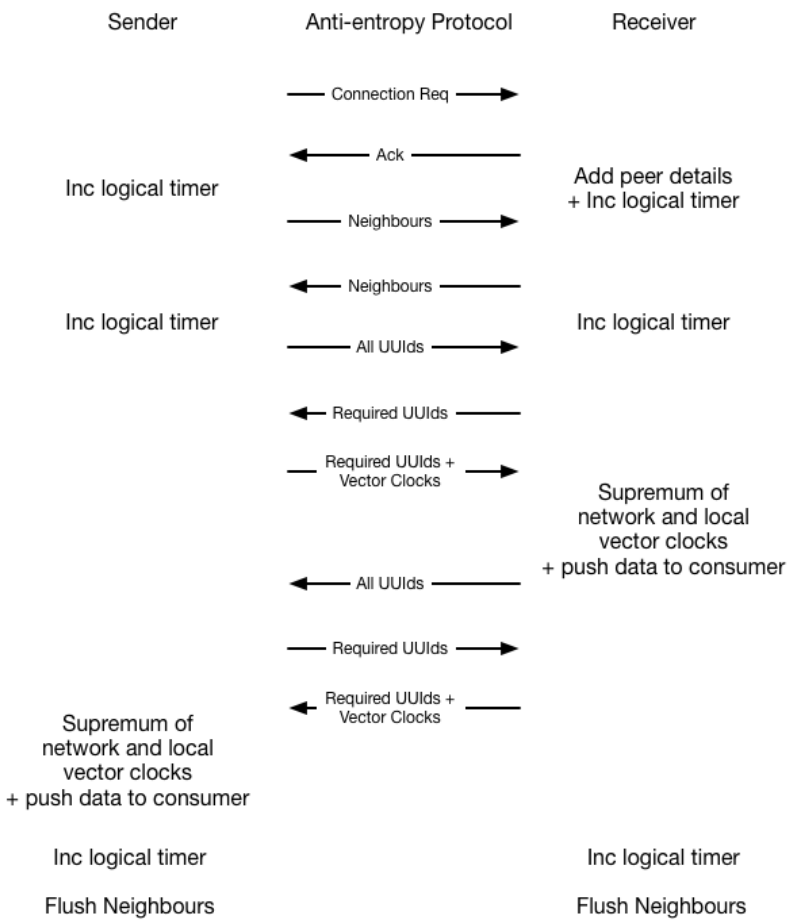


Figure 4.2: Summary of the anti-entropy protocol.

network, while the 17^{th} hosted the management tools for the testing infrastructure and was used for result gathering amongst other things.

Infrastructure management

A number of applications help to automate system auditing and logging, benchmarking and result gathering. The following three services listed below are hosted on the controller:

- Peer Management System
- GateKeeper

Size	Cores	Clock(GHz)	Memory(GB)
StandardA3	4	2.40	7
HDD(GiB)	Max data disks	Data disk throughput(IOPS)	Network Bandwidth
285	8	8x500	2/high

Table 4.1: Full specification details of the virtual hardware corresponding to the standard A3.

- RESTful logging server

The bootstrapping process in the original work was to cold start the neighbour cache of the peer with another peer's details introducing the peer to the network through the Anti-entropy protocol. Azure makes use of a dynamic IP, thus, the original bootstrapping process cannot be used. A gatekeeper service was created such that the bootstrapping procedure can be automated. The connecting peer would communicate with the Gatekeeper, which would reply with the details of the last known peer to have interacted with the gatekeeper.

A Peer Management System (PMS) was built to automate the start-up of the rendering process of each individual peer. Tests are automated through the use of a domain specific language which controls the behaviour of each collaborating peer. A RESTful logging server persists all the observable event exchanges among peers to a MySQL database, for later analysis. The machine hosting the Gatekeeper was given a static IP address (10.0.0.99), which is also used to seed the initial communication of the individual peers. At the start of a new test run, the PMS launches the Gatekeeper and interacts with the collaborating peers as illustrated in Figure 4.3.

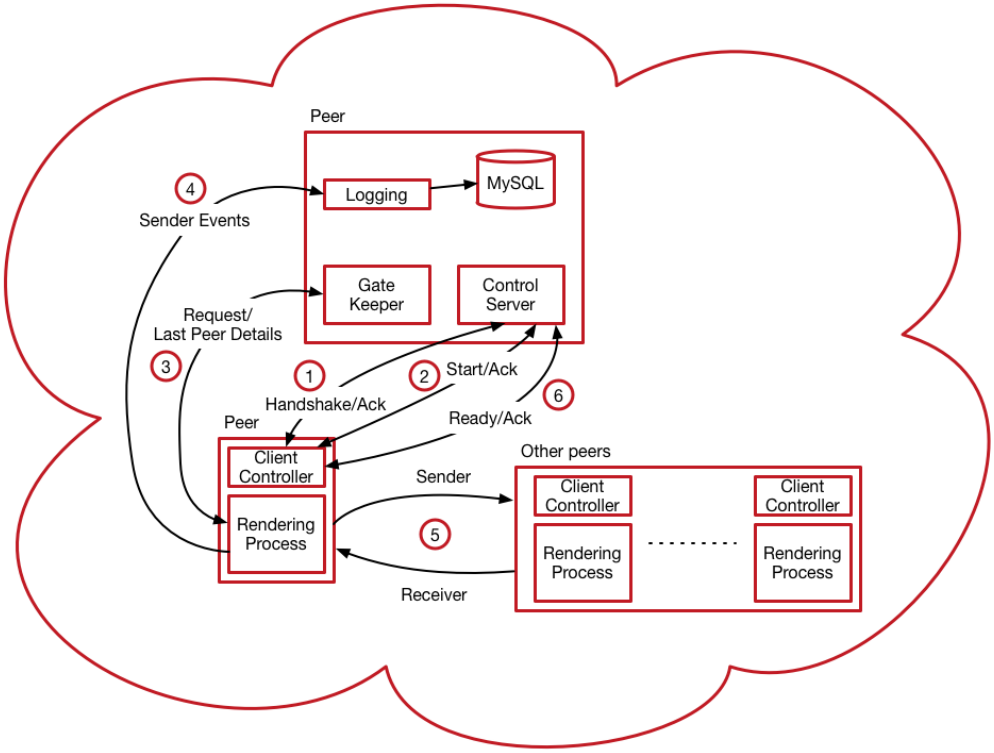


Figure 4.3: Summary of communication exchanges happening through out each experiment..

The enumerated exchanges in Figure 4.3 are described below: (i) a peer performs handshaking with the PMS (see Algorithm 4.2.3); (ii) following a successful handshake, a client awaits PMS start-up or termination commands; (iii) client receives command and the rendering process is booted or terminated accordingly; (iv) an active client enters the bootstrapping phase and communicated with the Gatekeeper. In this phase, the neighbour list is populated and primed for event dissemination; (v) client logs event exchanges through calls to the RESTful logging server, allowing further analysis of the network behaviour and dissemination to be carried out offline; (vi) on completing the rendering process, the client informs the PMS and awaits further commands.

Algorithm 4 Handshake request client to PMS server

```
1: previousDetails = NULL;
2: procedure ConnectToServer(server)
3:     var msg = REQ_HANDSHAKE_CONST
4:     var ack = ConnectToServer(socket, server, SERVER_PORT)
5:     if ack then
6:         ListenForCommands(socket)
```

Algorithm 5 PMS handshake

```
1: varPMS_SERVER_PORT = 11000
2: procedure ListForCommands
3:     var socket = nullptr
4:     InitSocket(PMS_SERVER_PORT, socket)
5:     while (true) do
6:         vardetails;
7:         var commandType = ListenForRequests(details, socket)
8:         if commandType == HANDSHAKE_REQUEST_CONST then
9:             var result = AddClientDetails(details)
10:            var success = SendMessage(socket, result)
11:        CloseConnection(socket)
```

Algorithm 6 PMS server start execution protocol

```
1: varPMS_CLIENT_PORT = 13000
2: procedure StartPeers
3:     for Client currentClient in clients do
4:         varresult = Connect(socket, currentClient.Ip, PMS_CLIENT_PORT)
5:         if result then
6:             Send(socket, START_PROCESS_CONST)
7:             result = RecevieAck(socket)
8:             if result == ACK_MESSAGE_CONST then
9:                 clientsList.SetStarted(client)
10:        CloseConnection(socket)
```

Algorithm 7 Gatekeeper peer-side protocol

```
1: GATE_KEEPER_PORT = 20000
2: procedure ConnectToGateKeeper(gateKeeperIp)
3:     varresult = Connect(socket, gateKeeperIp, GATE_KEEPER_PORT)
4:     AddNeighbours(result)
5:     CloseConnection(socket)
```

Map paths

Experiments in this chapter employ the same methodology present in the work of the original authors, diverging only in the maximum number of peers used. As a consequence, the tests were extended through the addition of eight new walk-through paths, supplementing the original eight and raising the total number of paths to sixteen.

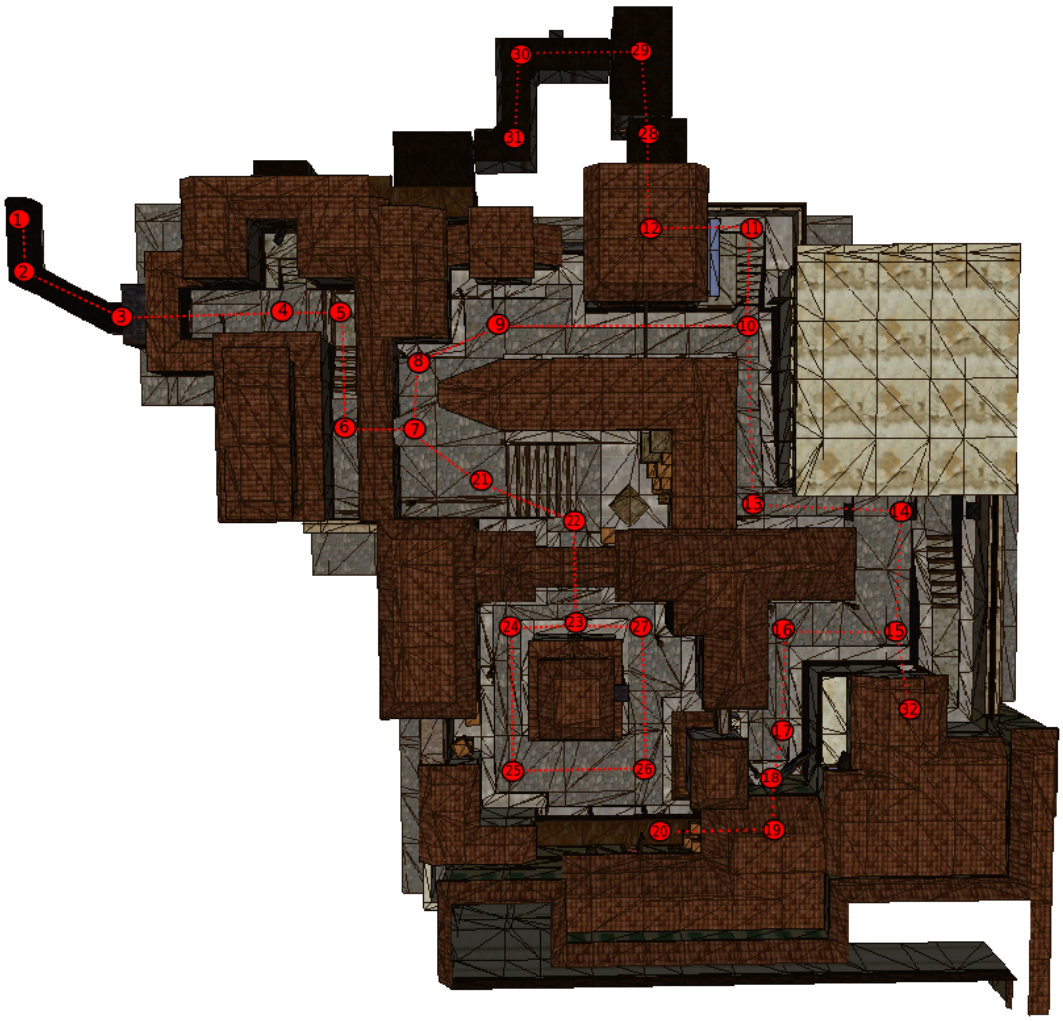


Figure 4.4: The configuration of paths created on the Town scene.

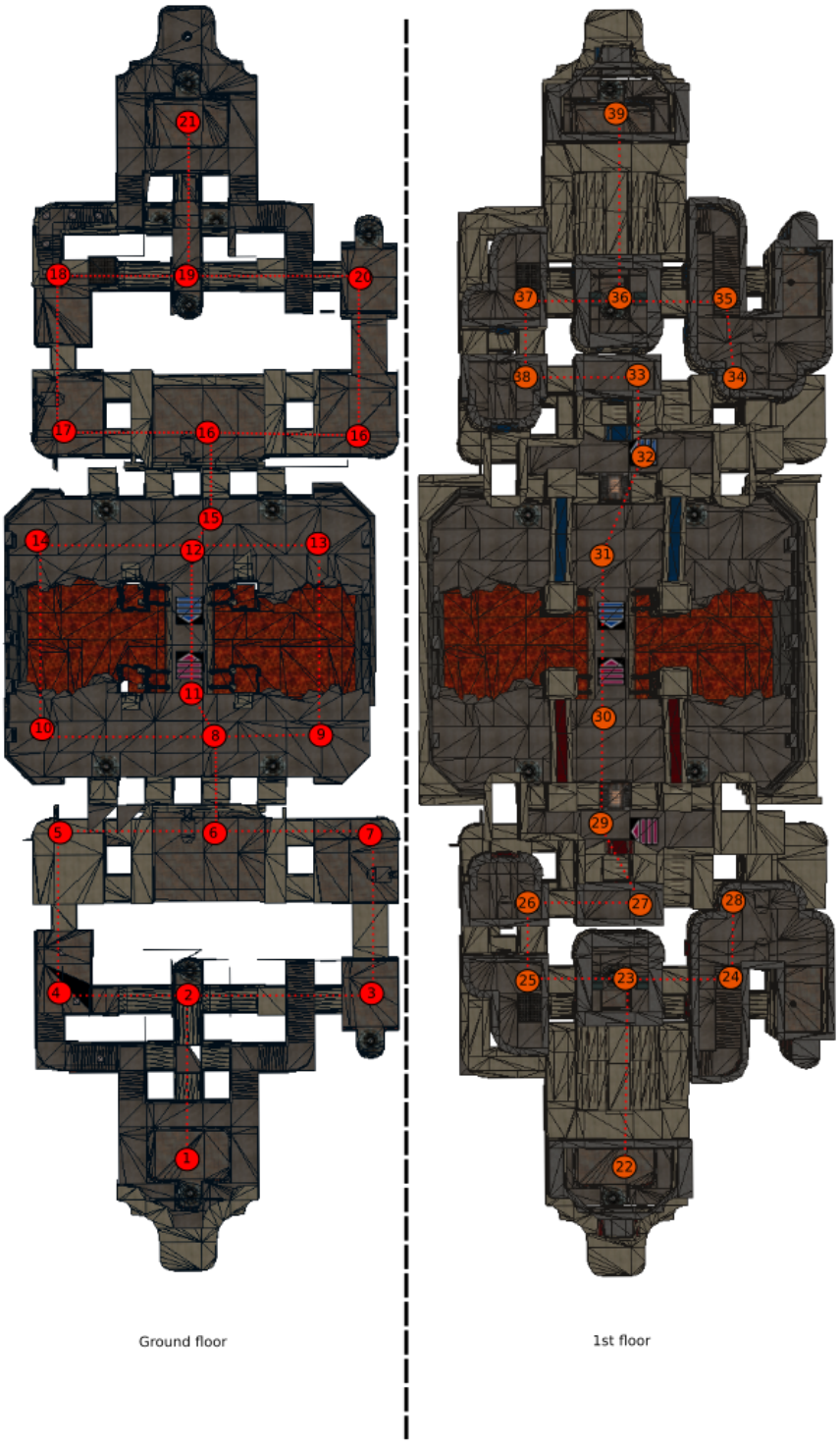
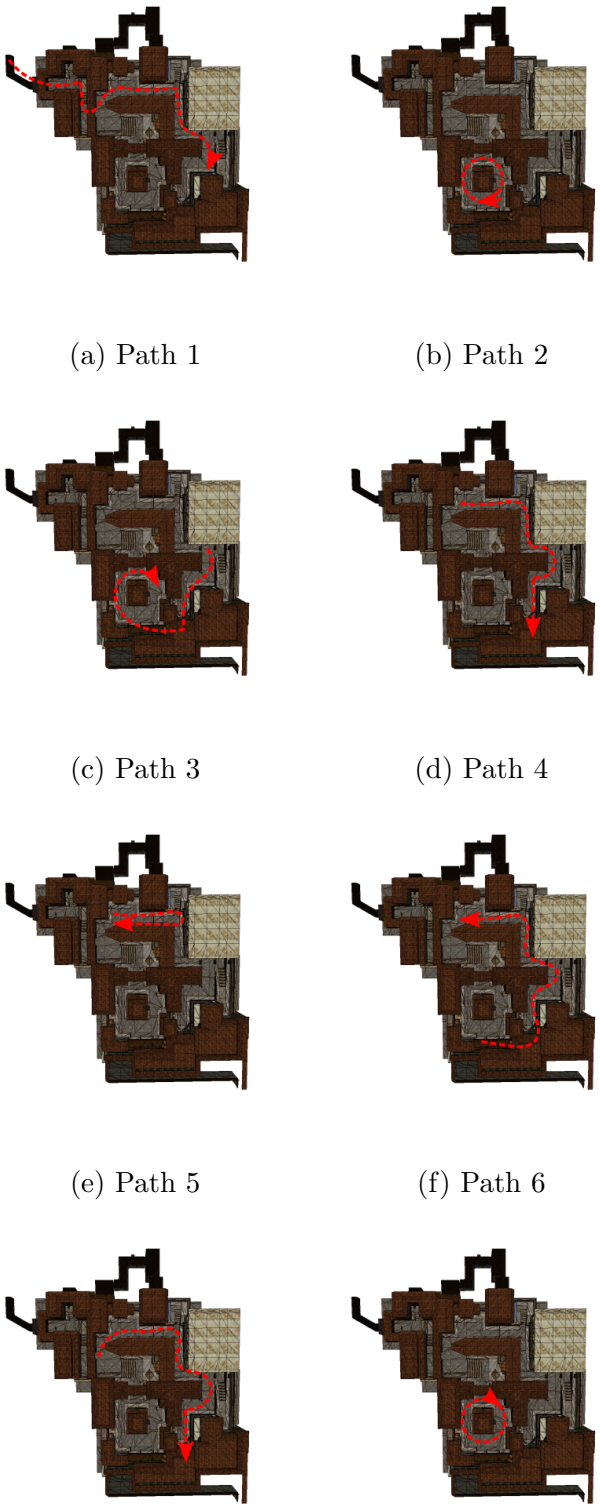


Figure 4.5: The configuration of paths created on the Inner Sanctum scene.



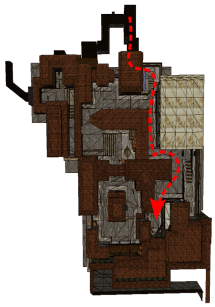
(a) Path 1 (b) Path 2

(c) Path 3 (d) Path 4

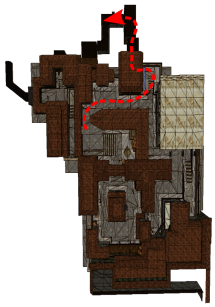
(e) Path 5 (f) Path 6

(g) Path 7 (h) Path 8

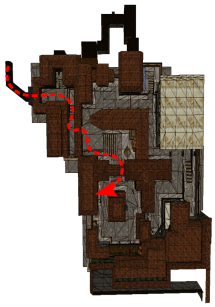
Figure 4.6: Paths created for the town scene.



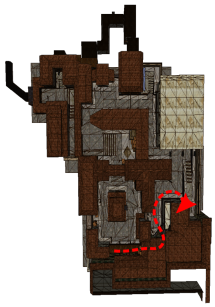
(a) Path 9



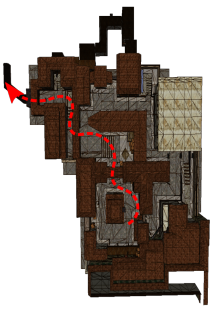
(b) Path 10



(c) Path 11

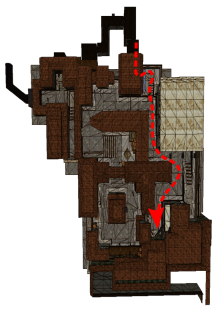


(d) Path 12

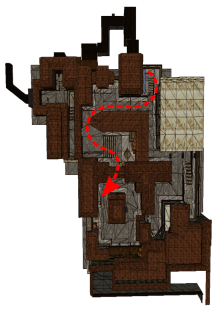


(e) Path 13

(f) Path 14



(g) Path 15



(h) Path 16

Figure 4.7: Sample of the paths created for the town scene.

Similarly, three different scenarios were tested: simultaneous, staggered with two different interval time deltas, which are highlighted below:

1. Simultaneous: all peers start collaborating together at once.
2. Staggered 400: A delay of 400 seconds is introduced between each peer's entry to the network.
3. Staggered 800: A delay of 800 seconds is introduced between each peer's entry to the network.

Scenarios 2 and 3 were introduced to replace 60 seconds and 120 seconds delays used in the original experiments, which were empirically determined, based on the performance of their rendering system. Taking into consideration the timings of the longest runs on the town scene for both the old system and the new implementation, using simple proportion the new delays were calculated to be 400 and 800 seconds alike.

4.3 Results

The experiments were run on the maps used by the original authors while more varied walkthroughs were created to accommodate 16 peers. Each experimental result was run twice, and results were averaged to eliminate outliers. System specific variables such as the resolution of the renderer and the error accepted by the IC were kept equal to to the values chosen by the original authors [15]. Thus, the IC error value (α) was set at 0.15 and resolution set at 512×512. The sending thread was originally constrained to 250 ms updates, but since the new implementation is less efficient, the delay between attempts by the sender for anti-entropy data exchanges is at least 2.5 seconds. The cache size of the neighbours of each peer was still maintained at a constant value of 4 for an eight peer collaboration while increased proportionally to 8 when the number of peers was increased to 16.

4.3.1 Speed-up in Quiescent networks

The speed-up gained by a peer does not depend on the classical distribution of work that occurs by through the distribution of ones workload to other computing elements. On the other hand, it depends on the IC samples that are received by a peer. If samples required by the peer are received from the network, the peer would achieve a speedup since the interpolation process is less expensive than computing new samples. In a quiescent network, updates by all peers have propagated throughout the whole network, making the global IC consistent across all peers [15]. The discrepancy between the first peer and the second peer timings for the walk-through is the range from the best time possible that can be achieved while the worst case is signified by the first peer. In a quiescent network, a performance increase was recorded ranging between $\times 1.7$ to $\times 2.5$ for the Inner-Sanctum scene while for the Town scene improvement recorded ranged between $\times 1.62$ and $\times 3.41$.

4.3.2 Simultaneous Start

In this test scenario, the time taken for the peers to render the walk-throughs will be observed along with the amount of samples that are exchanged. The rendering performance between 8 peers collaborating over the network will be compared with 16 peers. All peers start with an empty IC cache and thus, are required to compute samples on demand. Moreover, since all the may be computing all the samples in parallel, there is a greater risk for redundantly computed samples.

In the Town scene through the use of 8 peers, the speed-up gained was of $\times 0.99$. Thus there is a decrease in performance. On the other hand, the inner-sanctum scene was capable of exploiting more the peer to peer network and produced an overall speed-up of 1.28. If a respective peer's walk-through is never intersected by any other peer, then that peer would not gain assistance from the points received. Moreover, the points received would slow the rendering process, since the interpolations would have to be performed on more samples. The performance discrepancy

Scene	Peer	Single	Quiescent	Speed-up(×)
Town	1	7122	2103	3.38
	2	3782	1568	2.41
	3	2638	1287	2.05
	4	2930	1232	2.37
	5	1571	825	2.37
	6	3738	1485	2.52
	7	4601	1534	2.27
	8	3832	1527	2.51
	9	2614	1375	1.90
	10	1398	812	1.72
	11	4127	2365	1.66
	12	1214	617	1.97
	13	3923	2046	1.92
	14	2812	1477	1.91
	15	3129	1259	2.49
	16	2426	1439	1.67
Sanctum	1	3374	1424	2.37
	2	3923	1520	2.58
	3	2370	935	2.54
	4	2451	1068	2.32
	5	3437	1673	2.05
	6	3479	1574	2.21
	7	3023	1489	2.03
	8	2360	1377	1.71
	9	2436	980	2.49
	10	4001	1619	2.47
	11	5100	2240	2.28
	12	2296	1210	1.90
	13	4261	1646	2.59
	14	2310	882	2.62
	15	3887	2410	2.61
	16	3248	1436	2.27

Table 4.2: Timing and performance improvement between the leader and the follower.

between the two scenes, is dependant on the size of the scene. The Inner Sanctum is a smaller scene when compared to the Town scene, thus, peers have a larger probability of computing samples that will be used by other peers once shared.

As the authors argue, as the number of peers collaborating over the network increases, so does the number of samples that is shared between the peers. The number of peers collaborating simultaneously over the P2P was increased to 16, the results recorded a mean performance drop-off to 0.77 for the town, while the Inner Sanctum recorded a mean performance drop-off to 0.91. The number of samples exchanged over the network increases drastically. Figures **??** highlight the difference in the number of points that have been exchanged. In the town scene, a 50% increase was recorded and a 12% increase in the Inner Sanctum scene. The Inner Sanctum is a smaller scene when compared to the town scene, thus less points produced. Although the individual peers in both scenes received more points, many of the received points were redundant since they provided negligible change to the final renders.

4.3.3 Staggered Start

These tests were carried out so as to simulate multiple peers interacting together at different times. Delaying the peers by 400 for an 8 peer collaboration results in a $\times 1.02$ performance improvement for the town scene while the inner-sanctum recorded a decrease in performance to $\times 0.92$ Figure 4.3.3. Scaling the collaboration to 16 peers, the Town scene recorded speedup of $\times 0.91$, which is an improvement over the simultaneous 16 peers. The Inner Sanctum performance drop-down was marked with a mean speed up of $\times 0.82$.

Interarrival times of 800 for 8 peers recorded a speedup of $\times 1.06$ was recorded for the town whilst a speed up of $\times 0.95$ was recorded for the Inner Sanctum. That is better than the staggered 400 test runs yet still fall short from providing a considerable improvement. Scaling of the experiments to 16 peers resulted in a speedup of $\times 0.90$ for the Town, while the inner-sanctum decreased to a performance

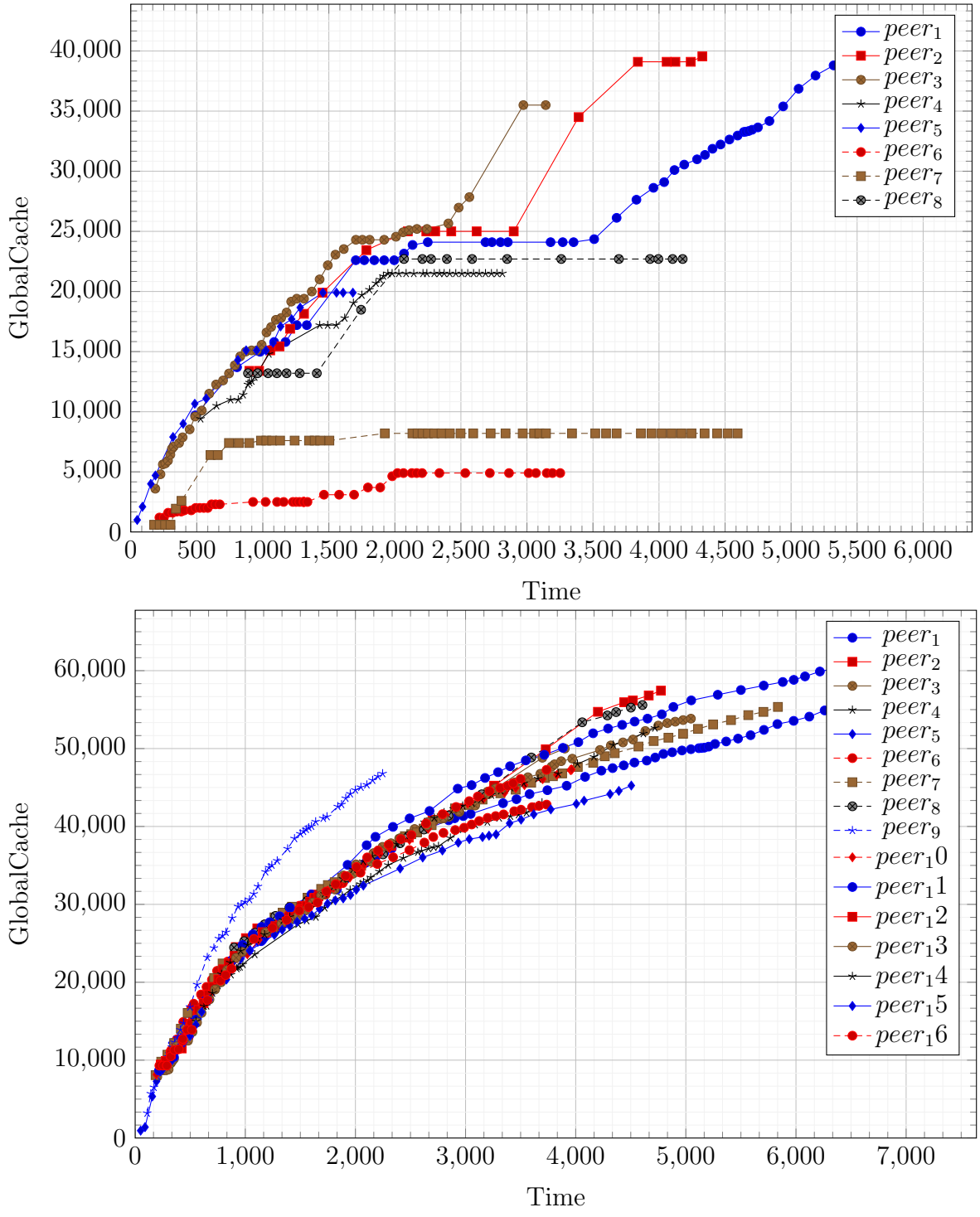


Figure 4.8: Total amount of shared points 16 peers, collaborating over the town scene.

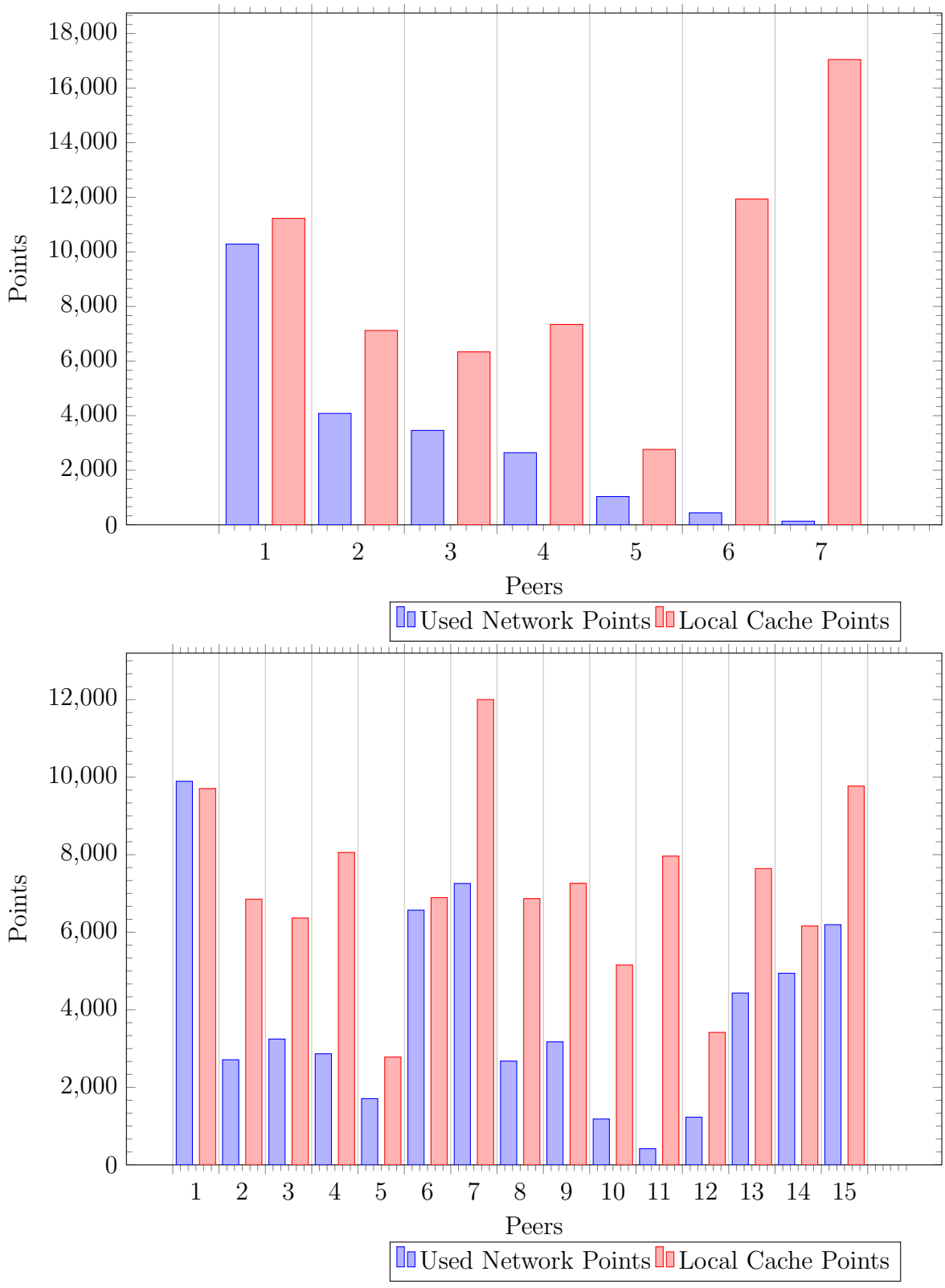


Figure 4.9: Ratio of created IC samples vs used network samples, 8(Top) to 16(Bottom) peers

Scene	Type	Speed-up 8	Speed-up 16	Global 8 peers	Global 16 peers
Town	simult	0.99	0.75	40,000	60,000
Sanctum	simult	1.28	0.91	32000	37,000
Town	stagg 400	1.02	0.91	49,000	53,000
Sanctum	stagg 400	0.92	0.82	32,000	58,000
Town	stagg 800	1.06	0.90	42,000	53,000
Sanctum	stagg 800	0.95	0.80	28,000	58,000

Table 4.3: Timing and performance between 8 peers and 16 peers for all scenarios.

of $\times 0.80$. The 800 delay proved to be less efficient than the 400 second delay,. In fact, in both scenes there is a drop in performance.

4.4 Discussion

The results show that there exists a correlation between the number of peers that have joined the network in parallel and the number of samples that are exchanged on the network. Although the staggered results should have performed better on the inner-sanctum as indicated by Bugeja et al.'s results, a drop-down in performance was still recorded. As illustrated in Figure 4.3.3, as the number of samples received over-saturates the cache.

As the number of peers increases, the chance of computing redundant samples also increases. These provide negligible change in the computed approximation, while creating a larger search space for the interpolation process. Although more samples were found in the cache, it still performed less efficiently than the 8 peer simultaneous run. The Inner Sanctum is a smaller map, thus, samples computed by two peers have a larger probability to be found in the same area. Since the town is a larger map, redundantly computed samples have a better chance of being distributed evenly over the map. To reduce IC sample density, Poisson filtering is applied, which discards points within a given radius of others already stored in the cache. This technique is only effective when samples are tightly packed due to

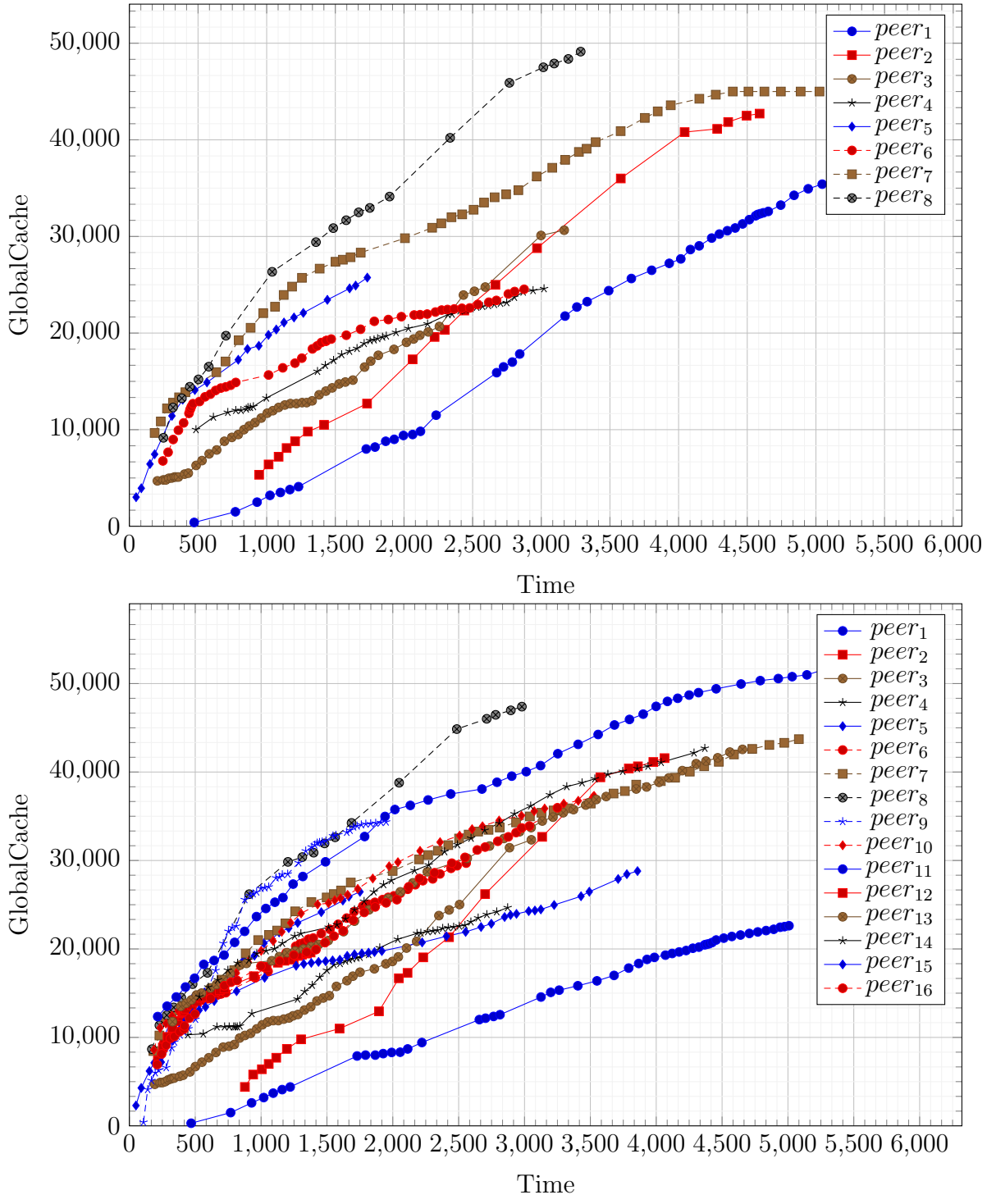


Figure 4.10: Time vs sample count received from network for a stagger of 400 in a collaboration of 8(Top) and 16(Bottom) peers.

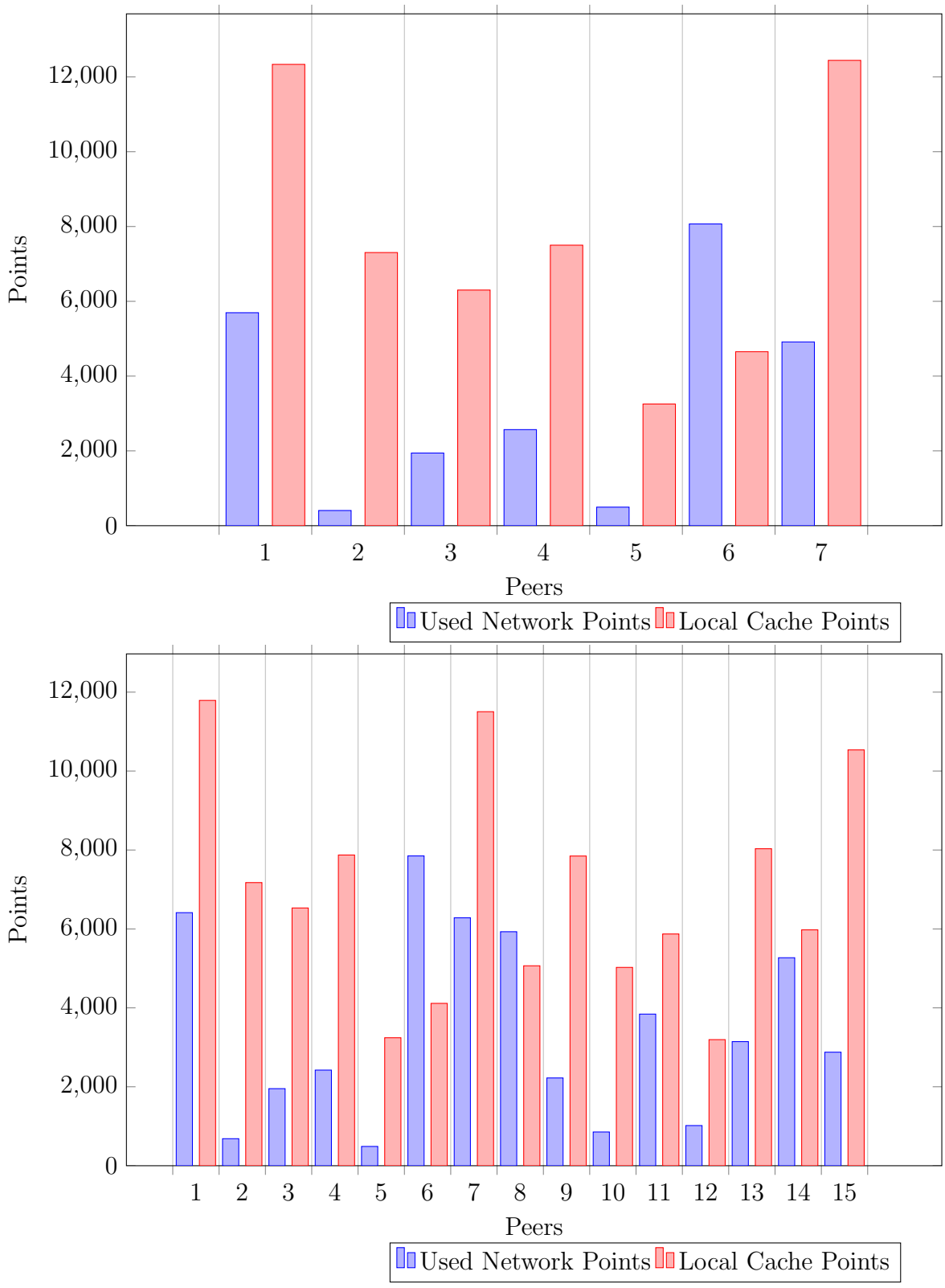


Figure 4.11: Ratio of locally generated samples to samples gained from the network

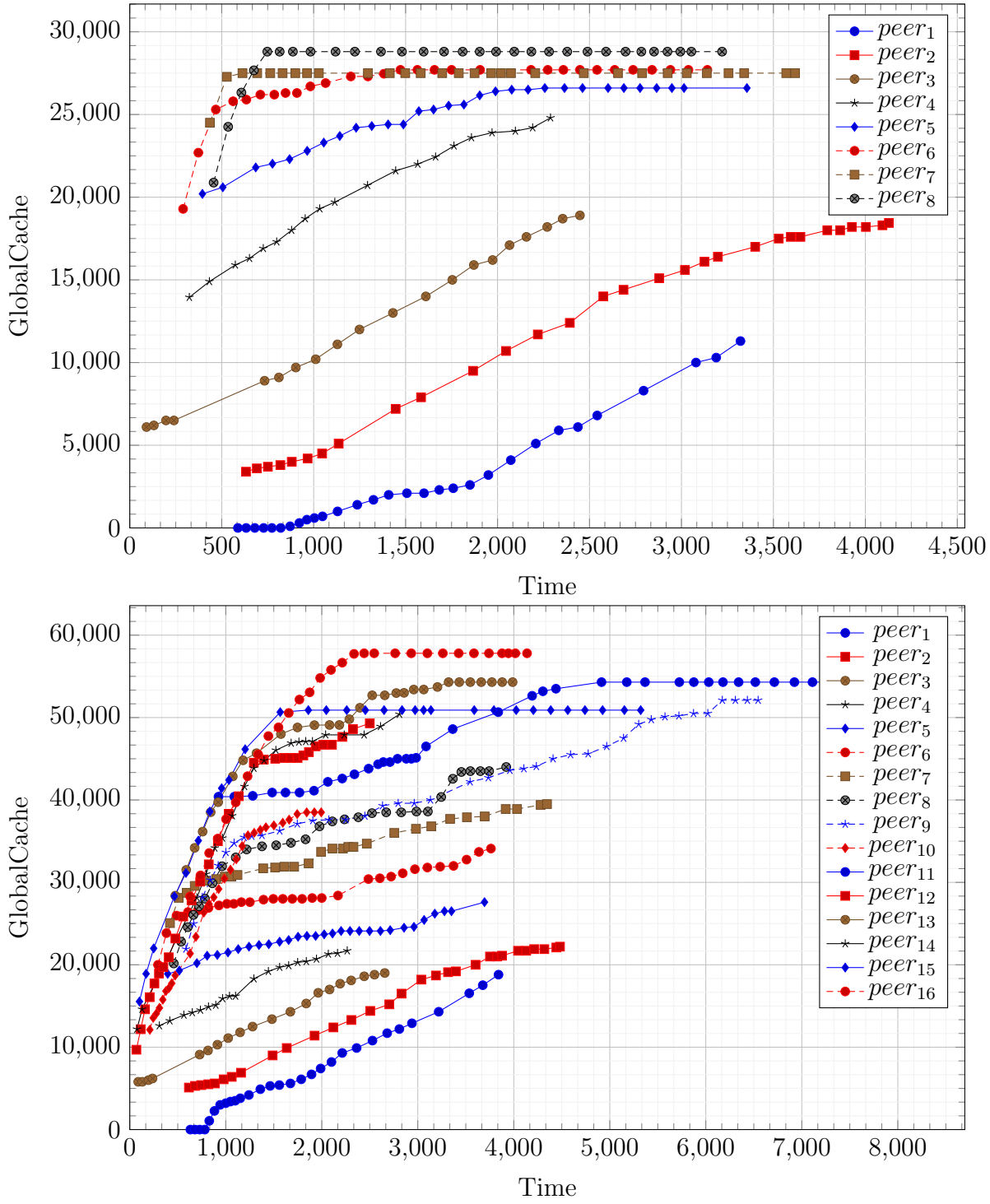


Figure 4.12: Received samples from network in a 8 and 16 peer staggered by 800 seconds collaboration over the town scene.

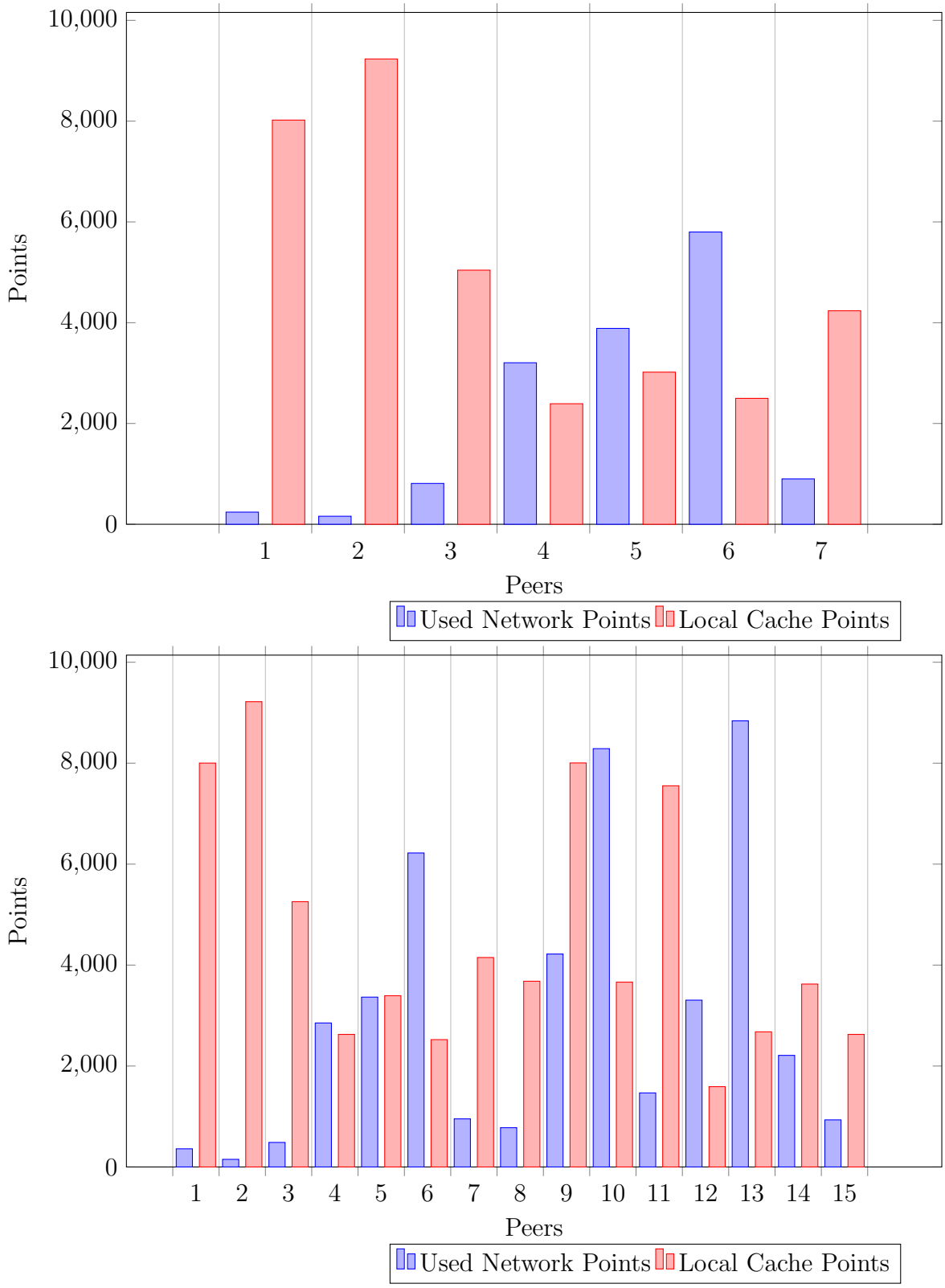


Figure 4.13: Computed samples vs used network samples in a 8 and 16 peer staggered by 800 seconds collaboration over the town scene.

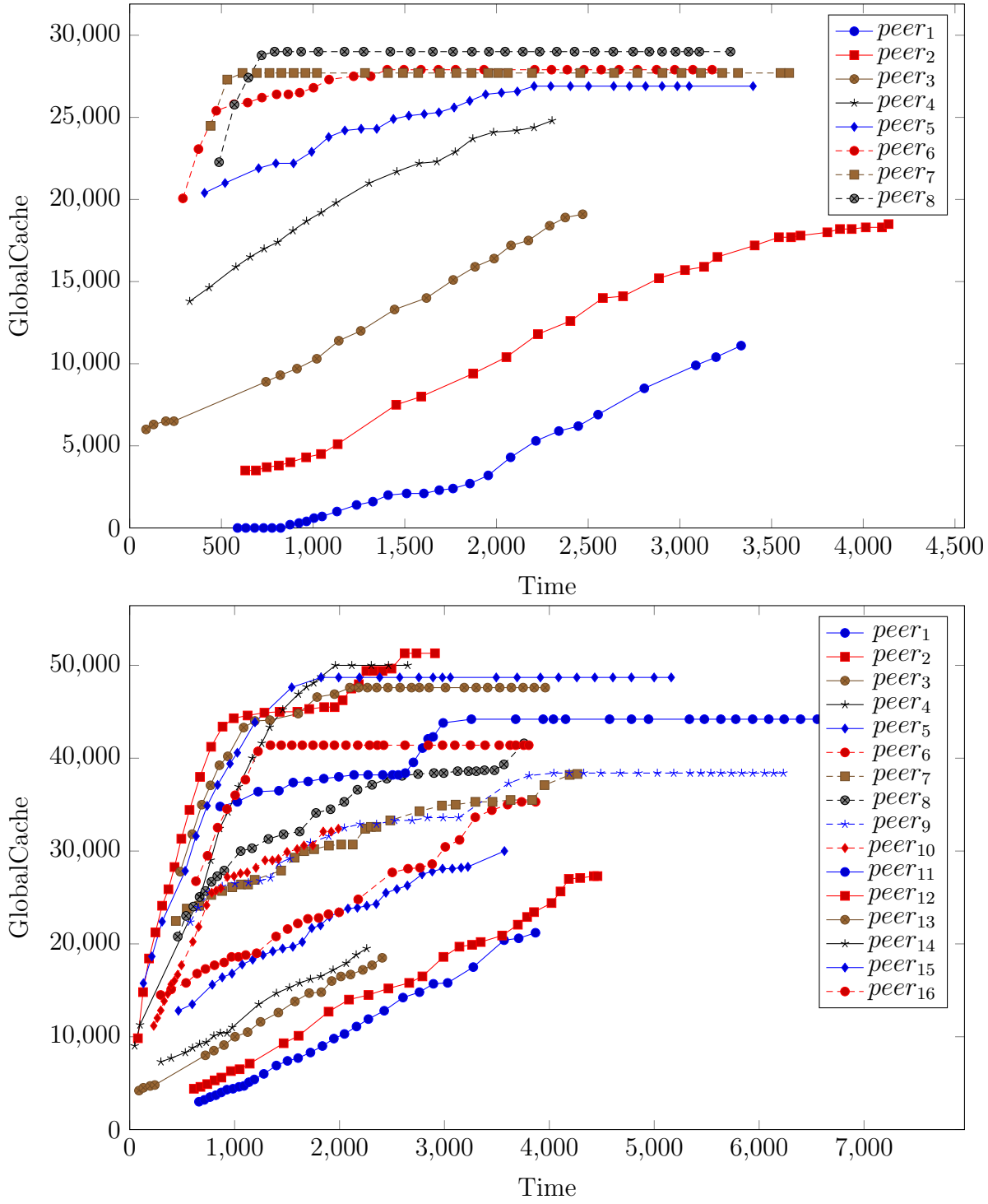


Figure 4.14: Time vs network sample count for an interarival stagger of 400 seconds 8 (Top) and 16 (Bottom) peers.

the small minimum ray length (see §refbackground). As the minimum ray length of samples lengthens, samples would be more sparsely packed, thus the minimum distance metric wouldn't be able to filter points in these areas. Moreover, as the number of samples stored in the cache increases, queries with the poisson method take longer.

Bugeja [14]highlights a potential solution to the over-saturation problem: the addition of a staging octree to which network updates would be committed. This would provide the rendering process insertions and interpolations that are more efficient. If an interpolation is not possible, using the samples stored in the local cache than the network cache would have queried to check for possible interpolations from points that were received from the network.

4.5 Summary

This chapter, extended the work of Bugeja et al. [15], to look at amortised rendering as the number of peers collaborating at an instance increases. The extended test runs provide a clearer view on the relationship between the number of samples redundantly shared as the number of peers running in parallel increases. Redundant samples lead to an over-saturation, degrading performance and the benefits of amortised computation.

5. Two phase Irradiance cache

This chapter introduces a novel extension of the network-based irradiance cache aimed at reducing over-saturation caused by redundant computation. In Chapter 4 we demonstrate how increasing the number of collaborating peers over the peer-to-peer network results in an increase in the number of exchanged irradiance samples, causing the local caches to store more potentially redundant samples which slow down the irradiance interpolation process. The two-phase irradiance cache (2PIC) presented in this chapter addresses the problem of redundant samples through the partitioning of the sample cache into local and global caches.

This chapter is structured as follows: in Section 5.1 the performance degradation incurred by the IC due to redundant samples is discussed, followed by an in-depth exposition of the 2PIC in Section 5.2. The method evaluation, including the experimental methodology, is presented in Section 5.3. A summary of the chapter and conclusions are given in Section 5.4.

5.1 Irradiance caching problem

The IC, introduced by Ward et al. [63], accelerates the computation of diffuse indirect lighting through the caching and interpolation of irradiance values. Diffuse indirect lighting is a slowly varying, low-frequency function that is computationally expensive to evaluate. Using a component-based approach, it can be separated

from the higher-frequency direct lighting, sampled at a lesser frequency and reconstructed via interpolation techniques with little to no loss in quality. The sampled quantity, irradiance, which is generated on demand during the distributed ray tracing process 2.14.3, is view-independent (diffuse reflectors reflect light equally in all directions).

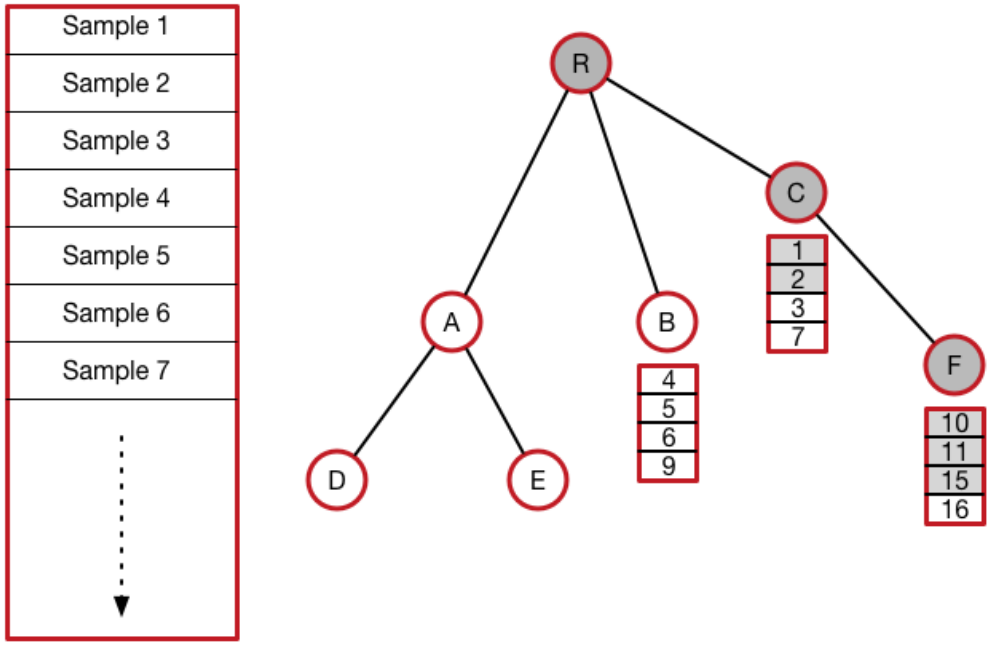


Figure 5.1: Retrieval of information from an octree data structure for the path R, C, F.

Interpolation of irradiance entails looking up irradiance samples neighbouring the affected point; in order to speed up searches, irradiance samples are stored in an octree. The octree is a tree-based data structure wherein 3D-space is partitioned hierarchically into octants; each octant maps to a node in the tree. Bound to each node is also a list of irradiance samples that is queried during traversal, when irradiance is being interpolated. An example of octree traversal for a given branch A is shown in Figure 5.1, where the shaded elements are the irradiance samples that contribute to the interpolation of the reconstructed indirect lighting.

During traversal, examining the individual irradiance samples at each node takes linear time in the number of samples ($\mathcal{O}(n)$, where n is the number of samples). Thus, a large number of samples stored at each individual node would increase the runtime of the algorithm without necessarily yielding an improved or more accurate solution. By the Nyquist theorem, sampling twice the frequency of the original waveform is enough to reconstruct the function - a higher sampling rate would would yield no measurable gains. On the contrary, a larger number of samples than is required leads to over-saturation of the irradiance cache, where any gains acquired through interpolation are lost due to the additional time spent processing redundant samples.

In Figure **??**, the earlier example is revisited; this time, however, an over-saturated octree incurs both algorithmic and realisation penalties: (i) the larger number of samples takes more time to process and (ii) cache capacity misses are more frequent given that coherence is lost due to the larger memory requirements of the new tree.

Over-saturation is a problem that arises out of collaboration: a single peer generates samples on demand, without over-saturating its own cache; it is only when samples generated from similar viewpoints are merged that over-saturation sets in. In Figure 5.2, two peers, A and B, generate samples for the overlapping regions in the scene due to sharing a similar view at Δt. In a future data exchange, the two peers exchange irradiance samples which are largely redundant to both (see Figure 5.3). The irradiance samples received over the network (blue) provide little improvement to the solution while significantly degrading interpolation performance. An accept-reject approach can be employed to avoid sample clumping, whereby samples that fail a minimum distance test are discarded. In particular, a sample x received over the network will be inserted in the local irradiance cache if and only if there is no other sample within a given distance d of x (see Algorithm 8).

This straightforward approach is only effective for areas with high sample densities (B in Figure 5.3); it is a heuristic that does nothing to alleviate or mitigate the

Algorithm 8 Min Distance Sample Insertion

```
1:  procedure InsertInNode
2:      newSampleState ← true
3:      d ← GetMinDist()
4:      newSample
5:      samplesStored ← SamplesStoredInNode()
6:      for each storedSample ∈ samplesStored do
7:          if dist(newSample, storedSample) > d then
8:              newSampleState = false
9:              break;
10:     if newSampeState then
11:         AddPointToNode(newSample)
```

number of redundant samples (blue) that contribute nothing to the reconstruction of indirect lighting.

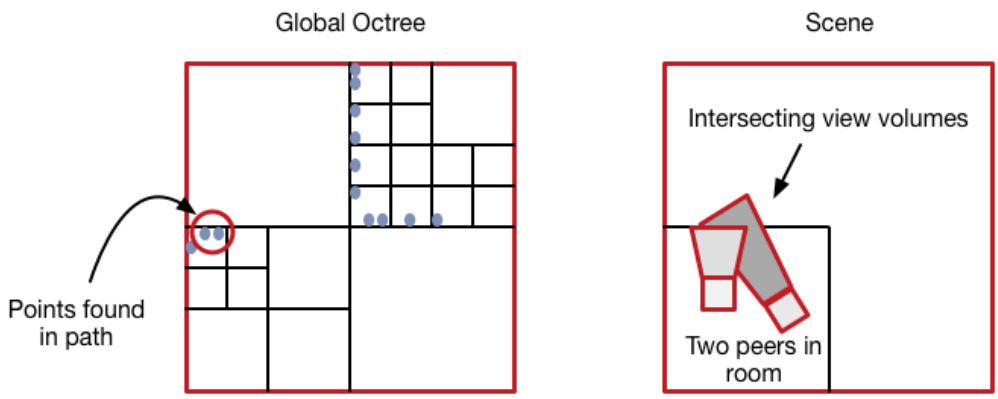


Figure 5.2: Intersection between views provides also redundant samples.

5.2 Two-phase Irradiance Cache

Irradiance samples generated locally share the same octree data structure as those received over the network. Although the latter samples might benefit other peers during future exchanges, they may also cause over-saturation and impede efficient rendering on the local peer. The two-phase irradiance cache (2PIC) has been intro-

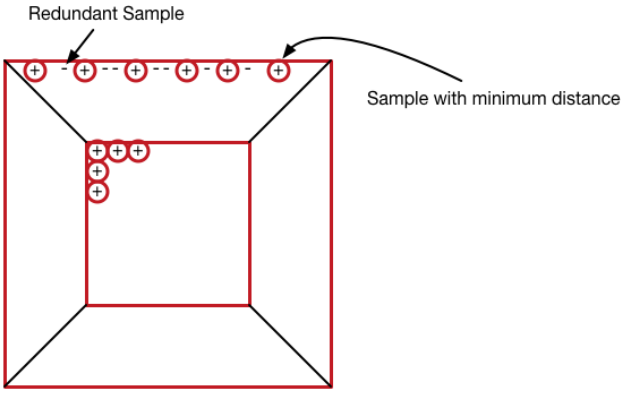


Figure 5.3: Samples not densely populated cannot be filtered using the minimum sample distance metric.

duce to mitigate the disadvantages incurred through the use of a single data structure to store irradiance samples indiscriminately. In particular, 2PIC utilises two octrees, termed the *local* and *global cache* respectively. The global cache receives all samples that are exchanged over the network, while the local cache contains only the samples required by the local rendering process to reconstruct indirect lighting. This partitioning strategy allows the rendering process to perform efficient traversals of the local cache, without being subject to over-saturation, while still retaining the ability to cooperate and disseminate irradiance samples to other peers over the network.

The interplay between the global and local caches of the 2PIC is such that the global cache is queried as seldom as possible, to avoid the penalties associated with an over-saturated octree (see Figure 5.4). However, through the introduction of a secondary cache, the 2PIC also introduces a further level of indirection in sample generation. Whenever diffuse indirect lighting needs to be estimated for a particular point x, the local cache is queried. If irradiance at x can be interpolated, then no further indirection penalties are incurred (see Algorithm 9). Conversely, if irradiance at x cannot be interpolated, the global cache is queried; the irradiance returned by the global cache is stored as a new sample in the local cache, to avoid

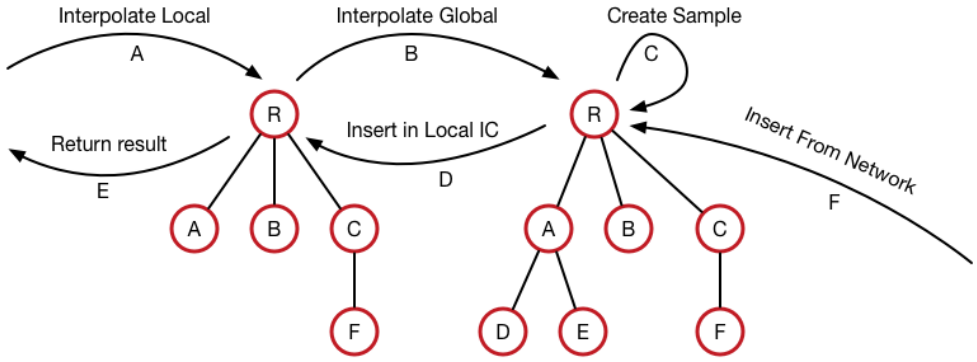


Figure 5.4: The 2 phase IC comprised from the local and global cache.

further indirections for the given region in the future (see Algorithm 10). When the global cache is queried, if irradiance cannot be interpolated, a new value is generated using distributed ray tracing and stored both in the global and local caches (see Algorithm 11).

Algorithm 9 Irradiance Interpolation in Local Cache

```
1: procedure InterpolateLocalCache
2:     localCache ← GetLocalCache()
3:     interpSamples ← localCache.GetInterpolationSamples()
4:     interpValue ← Interpolate(interpSamples)
5: return interpValue
```

It is important to highlight the fact that not all irradiance samples inserted in the local cache are directly sampled from the diffuse indirect lighting function - only samples generated locally in the global cache (when global cache interpolation fails) are directly sampled; other values are the result of global cache interpolation. The approximative nature of the 2PIC compels one to evaluate not only the possible speed-up brought about by the method, but also any divergence in image quality and fidelity where the method is employed in place of the original IC.

Algorithm 10 Irradiance Interpolation in Global Cache

```
1: procedure InterpolateGlobalCache
2:     localCache ← GetLocalCache()
3:     globalCache ← GetGlobalCache()
4:     interpSamples ← localCache.GetInterpolationSamples()
5:     if interpSamples.size == 0 then
6:         interpSamples ← globalCache.GetInterpolationSamples()
7:         interpValue ← Interpolate(interpSamples)
8:         localCache.Insert(interpValue)
9: return interpValue
```

Algorithm 11 Creation of New Irradiance Sample

```
1: procedure CreateNewSample
2:     localCache ← GetLocalCache()
3:     globalCache ← GetGlobalCache()
4:     interpSamples ← localCache.GetInterpolationSamples()
5:     if interpSamples.size <= 0 then
6:         interpSamples ← globalCache.GetInterpolationSamples()
7:         if ( theninterpSamples.size() == 0)
8:             newSample ← globalCache.NewSample()
9:             globalCache.Insert(newSample)
10:            localCache.Insert(newSample)
11: return newSample
```

5.3 Results

The 2PIC is evaluated on three distinct criteria: (i) image fidelity with respect to the original IC; (ii) indirection overhead with respect to the original IC; and (iii) network speed-up in the context of collaborative peer-to-peer rendering. An experimental methodology precedes these evaluations; having determined the best parameters for the 2PIC in terms of quality and performance with respect to the IC, the network speed-up for collaborative rendering is then tested and evaluated.

5.3.1 Quality and Performance Methodology

In this section, the 2PIC is compared and contrasted to the original IC in terms of image fidelity and performance. The performance aspect gauges the overhead incurred by the additional indirection due to the second cache in the 2PIC when compared to the IC.

Image Quality Metrics

In order to evaluate image quality and fidelity, established metrics for image differences are employed. Aggarwal et al. [10] use the visual difference predictor (VDP) to measure the perceptual difference between reconstructed images from their distributed rendering system and a path traced [34] ground truth image. Peak Signal to Noise Ratio (PSNR) is also commonly employed as a measure of degradation between compressed and uncompressed images. The PSNR metric is based on the mean squared error (MSE) between two images and is measured in decibels; it is given by Equation 5.3.1:

$$MSE = \frac{1}{MN} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} ||f(i,j) - g(i,j)||^2, \tag{5.1}$$

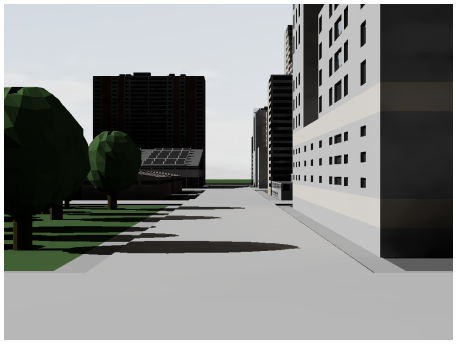
where f and g are the signals representing the two images. PSNR gives the peak square error rather than the mean and is given by Equation 5.2:

$$PSNR = 20 \log(\frac{255}{\sqrt{MSE}}). \tag{5.2}$$

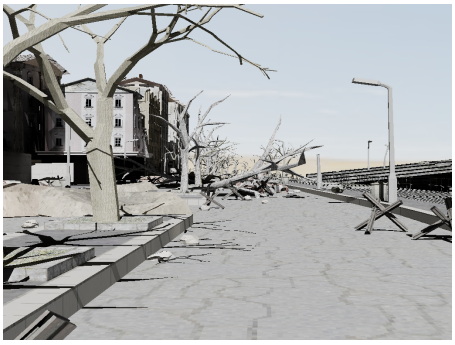
PSNR does not factor perceptual responses in the differences between the two images.

Evaluation Data Set

The data set used in image quality and fidelity tests satisfies a wide set of criteria, from complex scenes with many intricate geometric designs, highly textured scenes, high-polygon scenes, low-polygon scenes and the original scenes used by Bugeja et al. [15]. More specifically, Inner Sanctum (Figure 5.5c) and Town (Figure 5.5d) are taken from the video games Quake and Halflife respectively, Damaged Town (Figure 5.5b) and Big City (Figure 5.5a) are two large and complex scenes, and the Sponza Atrium (Crytek) (Figure 5.5e), Sponza Atrium (Dabrovic) (Figure 5.5e) and Sibenik Cathedral (Figure 5.5g) are data sets commonly used in the evaluation of global illumination algorithms.



(a) Big City



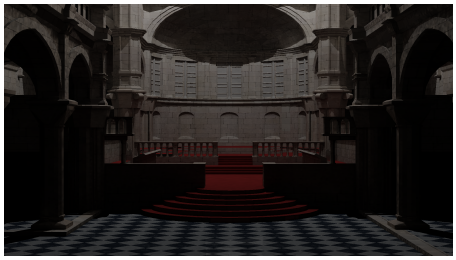
(b) Damaged downtown

(c) Inner Sanctum

(d) Town

(e) Sponza

(f) Sponza Dabrovic



(g) Sibenik

Figure 5.5: Sample shots from the used scenes.

Quality and Performance Overhead

In order to measure the performance overhead of the 2PIC with respect to the original IC, the two methods have been compared under the same error threshold over a range of image syntheses. To assess the performance overhead of 2PIC, its global cache error was set to zero; this forces all searches in the global cache to generate an irradiance sample using distributed ray tracing, fundamentally making the 2PIC behave in the same way as a traditional IC, only with the additional indirection of failed searches in the global cache. Three different camera positions were chosen for each of the data sets; for each camera position, three error thresholds were employed ($\alpha = 0.1, \alpha = 0.2, \alpha = 0.5$). Typical α values range between $(0.1, 0.2)$ and the PSNR extracted; Bugeja et al. [15] use $\alpha = 0.15$, for instance. The inclusion of $\alpha = 0.5$ extends the analysis further than the typical values.

The image quality and fidelity of 2PIC output when compared to the original IC algorithm was quantified though the use of a number of combinations of α thresholds for both the local and global cache, contrasted with equivalent IC syntheses with an $\alpha = 0.1$. Specifically, the Cartesian product of $\alpha_{local} = \{0.1, 0.2, 0.5\}$ and $\alpha_{global} = \{0.1, 0.2, 0.5\}$ contrasted to ground truth images from the traditional IC show how varying the respective thresholds for each cache affects image quality and performance, while serving to highlight any correlation between these two properties.

5.3.2 Quality and Performance Overhead Results

Table 5.1 shows the mean PSNR values for the Sibenik scene; the choice of $\alpha = 0.1$ was motivated by the necessity to compare the 2PIC output images against high quality traditional IC results.

The overhead incurred by the 2PIC over the traditional IC is shown in tables 5.2, 5.3, 5.4, 5.5, 5.6 and 5.7. Note that overhead, in seconds, is computed as the difference between the total frame rendering time under 2PIC and and the

α	$\alpha_{\mathbf{local}}$	$\alpha_{\mathbf{global}}$	**PSNR**
0.1	0.1	0.1	46.49
0.1	0.1	0.2	52.81
0.1	0.1	0.5	43.19
0.1	0.2	0.1	45.54
0.1	0.2	0.2	45.16
0.1	0.2	0.5	40.22
0.1	0.5	0.1	42.175
0.1	0.5	0.2	42.189
0.1	0.5	0.5	41.97

Table 5.1: PSNR results for the respective pairs of alpha when compared to a render produced by an IC with an acceptance error of 0.1.

traditional IC for the same image; a (-) signifies no 2PIC overhead.

α	$\alpha_{\mathbf{local}}, \alpha_{\mathbf{global}}$	**Image**	**IC(s)**	**2PIC(s)**	**Overhead(s)**
0.1	0.1, 0.0	1	1195	1045	-
0.2	0.2, 0.0	1	794	766	-
0.5	0.5, 0.0	1	283	284	1
0.1	0.1, 0.0	2	1808	1223	-
0.2	0.2, 0.0	2	1181	1054	-
0.5	0.5, 0.0	2	476	399	-
0.1	0.1, 0.0	3	346	219	-
0.2	0.2, 0.0	3	247	237	-
0.5	0.5, 0.0	3	130	124	-

Table 5.2: Comparison between the IC and 2PIC in Damaged Down-town.

Scenes with moderately complex geometry, such as the Sponza Atrium (Crytek) (see Table 5.6) records a substantial overhead in all of the frames rendered. In other scenes, the 2PIC performed on par with the traditional IC, suggesting that the two-level indirection of the 2PIC may be creating a performance bottleneck during irradiance sample insertions and interpolation. Clearly, while showing the performance overhead of the 2PIC, having used an α_{global} value of zero nullifies all the possible benefits of using the technique.

α	$\alpha_{\mathbf{local}}, \alpha_{\mathbf{global}}$	**Image**	**IC(s)**	**2PIC(s)**	**Overhead(s)**
0.1	0.1, 0.0	1	302	311	9
0.2	0.2, 0.0	1	126	130	4
0.5	0.5, 0.0	1	57	57	-
0.1	0.1, 0.0	2	337	297	40
0.2	0.2, 0.0	2	131	112	-
0.5	0.5, 0.0	2	62	54	-
0.1	0.1, 0.0	3	246	229	-
0.2	0.2, 0.0	3	121	106	-
0.5	0.5, 0.0	3	64	63	-

Table 5.3: Comparison between the IC and 2PIC in Inner Sanctum.

α	$\alpha_{\mathbf{local}}, \alpha_{\mathbf{global}}$	**Image**	**IC(s)**	**2PIC(s)**	**Overhead(s)**
0.1	0.1, 0.0	1	510	500	-
0.2	0.2, 0.0	1	243	243	-
0.5	0.5, 0.0	1	108	109	1
0.1	0.2, 0.0	2	734	721	-
0.2	0.5, 0.0	2	364	356	-
0.5	0.5, 0.0	3	201	186	-
0.1	0.1, 0.0	3	1263	1124	-
0.2	0.2, 0.0	3	645	656	11
0.5	0.5, 0.0	3	265	230	-

Table 5.4: Comparison between the IC and 2PIC in Big City.

α	$\alpha_{\mathbf{local}}, \alpha_{\mathbf{global}}$	**Image**	**IC(s)**	**2PIC(s)**	**Overhead(s)**
0.1	0.1, 0.0	1	2142	2698	556
0.2	0.2, 0.0	1	780	1418	638
0.5	0.5, 0.0	1	251	894	643
0.1	0.2, 0.0	2	1225	1920	695
0.2	0.5, 0.0	2	470	1177	707
0.5	0.5, 0.0	3	155	885	730
0.1	0.1, 0.0	3	916	1613	697
0.2	0.2, 0.0	3	311	1136	825
0.5	0.5, 0.0	3	92	873	781

Table 5.5: Comparison between the IC and 2PIC in Sponza Dabrovic.

5.3.3 2PIC Quality-Performance Function

The 2PIC results vis-a-vis quality and performance are highly dependent on its configuration parameters: α_{local} and α_{global}. In the case of the traditional IC,

α	$\alpha_{\mathbf{local}}, \alpha_{\mathbf{global}}$	**Image**	**IC(s)**	**2PIC(s)**	**Overhead(s)**
0.1	0.1, 0.0	1	3094	4441	1347
0.2	0.2, 0.0	1	1338	2371	1033
0.5	0.5, 0.0	1	391	1331	933
0.1	0.2, 0.0	2	1669	3204	1535
0.2	0.5, 0.0	2	682	2047	1365
0.5	0.5, 0.0	3	292	1483	1191
0.1	0.1, 0.0	3	5811	6478	667
0.2	0.2, 0.0	3	2482	3133	651
0.5	0.5, 0.0	3	703	1243	540

Table 5.6: Comparison between the IC and 2PIC in Crytek Sponza.

α	$\alpha_{\mathbf{local}}, \alpha_{\mathbf{global}}$	**Image**	**IC(s)**	**2PIC(s)**	**Overhead(s)**
0.1	0.1, 0.0	1	101	152	51
0.2	0.2, 0.0	1	52	76	24
0.5	0.5, 0.0	1	47	49	2
0.1	0.2, 0.0	2	93	142	49
0.2	0.5, 0.0	2	56	60	4
0.5	0.5, 0.0	3	48	46	-
0.1	0.1, 0.0	3	108	119	11
0.2	0.2, 0.0	3	61	67	6
0.5	0.5, 0.0	3	45	49	4

Table 5.7: Comparison between the IC and 2PIC in Town.

the user-defined constant α denoted the permissible interpolation error; with the 2PIC, the mapping between local and global α values and image output quality is less straightforward and intuitive (except for border cases). To address this shortcoming, a scoring function is introduced that gives the best 2PIC α values for the desired quality-performance trade-off. More specifically, for each triple $(\alpha, \alpha_{local}, \alpha_{global})$ a mapping to a score s is created, determined by: (i) the quality of the reference image, (ii) the PSNR between the reference and the 2PIC output, and (iii) the magnitude of the performance improvement given by 2PIC, determined through a number of preliminary network runs of the 2PIC algorithm with various α values, in contrast with the same runs using the traditional IC, again at various α values.

Particularly, for a given reference image the quality is given by $10(0.5 - I_\alpha)$, where where I_α is the error threshold used to generate image. The quality scoring function Q returns a value in the range $[0, 1]$ based on the PSNR between reference and 2PIC sequences:

$$Q(I_{psnr}) = \left\{ \begin{array}{l} 0\ ;\ -\infty < I_{psnr} < 30 \\ \frac{(I_{psnr}-30)}{20}\ ;\ 30 \leq I_{psnr} < 50 \\ 1\ ;\ 50 \leq I_{psnr} < \infty \end{array} \right\}, \tag{5.3}$$

where I_{psnr} is the PSNR between the 2PIC output and a reference; the performance function P scores the 2PIC performance for a given set of local and global α values:

$$P(I_t) = 0.6(I_t) - 0.675, \tag{5.4}$$

where I_t is the network performance difference between 2PIC and the traditional IC. The final scoring function s is subsequently given by:

$$s(I) = 10(0.5 - I_\alpha)\left[Q(I_{psnr}) + P(I_t)\right] \tag{5.5}$$

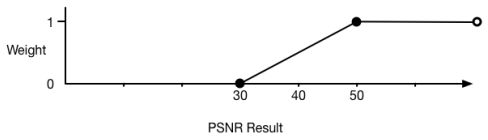


Figure 5.6: Quality weighting graph for the 2 phase IC

The weighted 2PIC scores for $\alpha, \alpha_{local}, \alpha_{global} \in 0.1, 0.2, 0.5$ are shown in Table **??**, together with the individual component scores for quality and performance. The table is sorted in descending order by score; the preferred quality-tradeoff overall score is shaded in grey.

α	α_{local}	α_{global}	**PSNR**	**Perf.**	**Wt. PSNR**	**Wt. Per.**	**Overall Rating**
0.1	0.5	0.5	41.97	4.6	0.5985	2.085	5.1985
0.1	0.5	0.2	42.189	4.56	0.60945	2.061	5.16945
0.1	0.2	0.5	40.22	4.56	0.511	2.061	5.071
0.1	0.5	0.1	42.175	4.38	0.60875	1.953	4.98875
0.1	0.1	0.5	43.19	3.45	0.6595	1.395	4.1095
0.1	0.1	0.2	52.81	2.49	1	0.819	3.49
0.1	0.2	0.2	45.16	2.82	0.758	1.017	3.578
0.1	0.2	0.1	45.54	2.72	0.777	0.957	3.497
0.2	0.5	0.2	42.189	1.66	0.60945	0.321	2.26945
0.2	0.5	0.5	41.97	1.68	0.5985	0.333	2.2785
0.2	0.5	0.1	42.175	1.59	0.60875	0.279	2.19875
0.2	0.2	0.5	40.22	1.66	0.511	0.321	2.171
0.2	0.1	0.2	52.81	0.9	1	-0.135	1.9
0.2	0.1	0.5	43.19	1.25	0.6595	0.075	1.9095
0.1	0.1	0.1	46.49	1	0.8245	-0.075	1.8245
0.2	0.2	0.2	45.16	1.03	0.758	-0.057	1.788
0.2	0.2	0.1	45.54	0.98	0.777	-0.087	1.757
0.5	0.1	0.2	52.81	0.54	1	-0.351	1.54
0.5	0.5	0.2	42.189	1	0.60945	-0.075	1.60945
0.5	0.5	0.5	41.97	1.02	0.5985	-0.063	1.6185
0.5	0.5	0.1	42.175	0.96	0.60875	-0.099	1.56875
0.5	0.2	0.5	40.22	1.007	0.511	-0.0708	1.518
0.5	0.1	0.5	43.19	0.76	0.6595	-0.219	1.4195
0.5	0.2	0.1	45.54	0.59	0.777	-0.321	1.367
0.5	0.2	0.2	45.16	0.62	0.758	-0.303	1.378
0.2	0.1	0.1	46.49	0.36	0.8245	-0.459	1.1845
0.5	0.1	0.1	46.49	0.22	0.8245	-0.543	1.0445

Table 5.8: Results of the quality - performance calculations sorted in descending order.

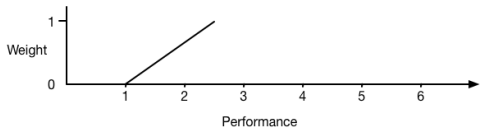


Figure 5.7: Performance weighting graph for the 2 phase IC

5.3.4 Network Performance Methodology

In Chapter 4 it was shown that amortisation performance deteriorated as the number of collaborating peers increased. This degradation was hypothetically linked to the over-saturation of the irradiance cache; in the network performance evaluation, the experiments of Section 5.3 are carried out with the 2PIC enabled. Since the new method provides a wider range of operating parameters than the traditional IC, the quality/performance arguments have been determined from the results of the respective quality and performance overhead tests above (see Section 5.3.1). More specifically, α_{local} and α_{global} were both set to 0.5 (see Table **??**).

Simultaneous Start

In simultaneous start, all peers join the network at the same time. This scenario is prone to redundant computation since on joining the network none of the peers have any computation results to share. The Town scene resulted in a network speed-up of $\times 2.44$ for the 8-peer network, while Inner Sanctum shown an speed-up of $\times 2.07$, on the same network setup. Scaling the network size to 16 peers saw a mean network speed-up of $\times 2.05$ achieved on both scenes. Figure 5.3.4 illustrates the performance both with and without 2PIC, while Figure 5.3.4 shows the samples exchanged.

Staggered Start

In the staggered start experiments, the peers do not join the network simultaneously, but rather one after the other; two staggered start scenarios were tested, one

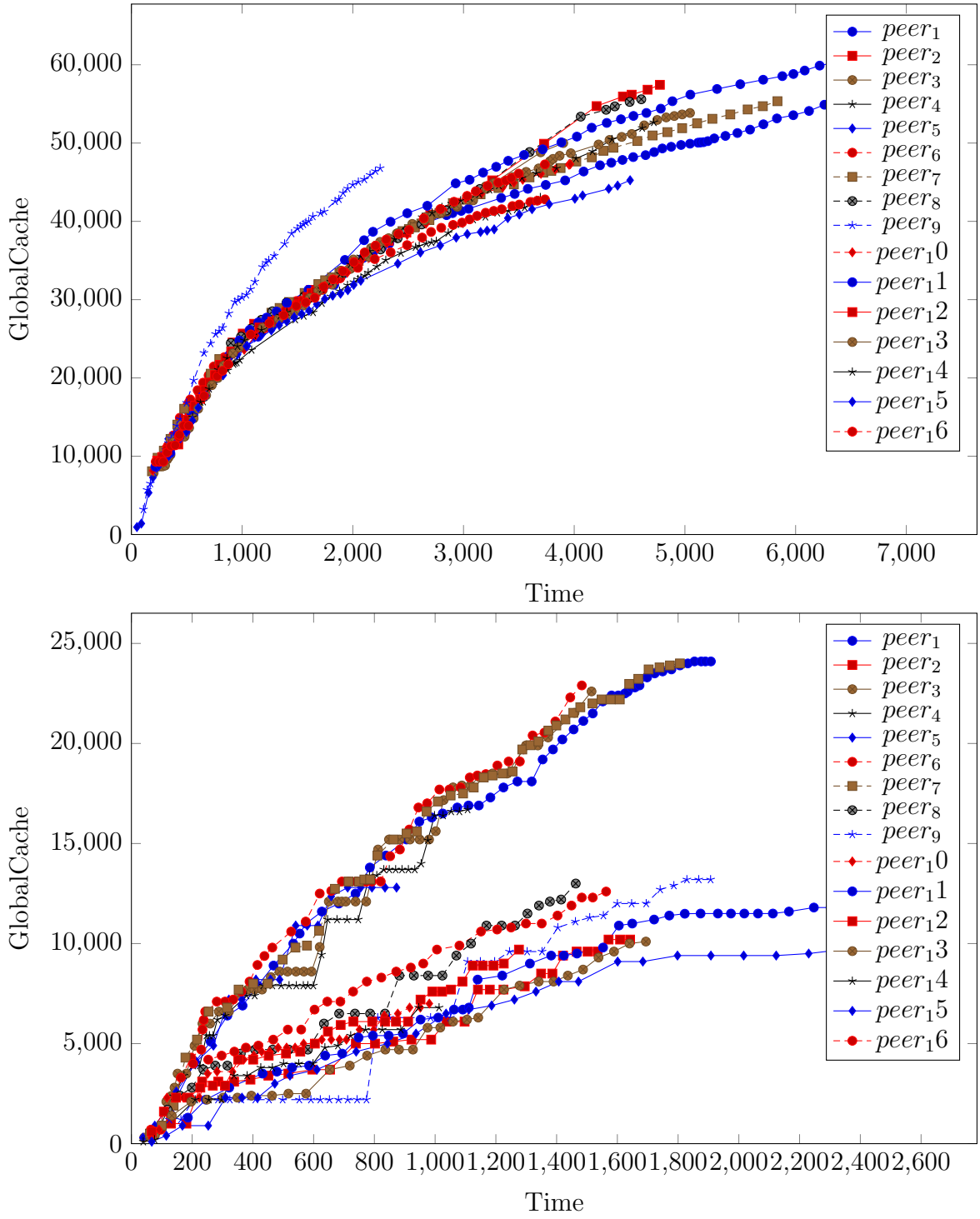


Figure 5.8: Comparison of the global caches samples count between 16 peer simultaneous(Top) and 16 peer simultaneous using 2PIC(Bottom).

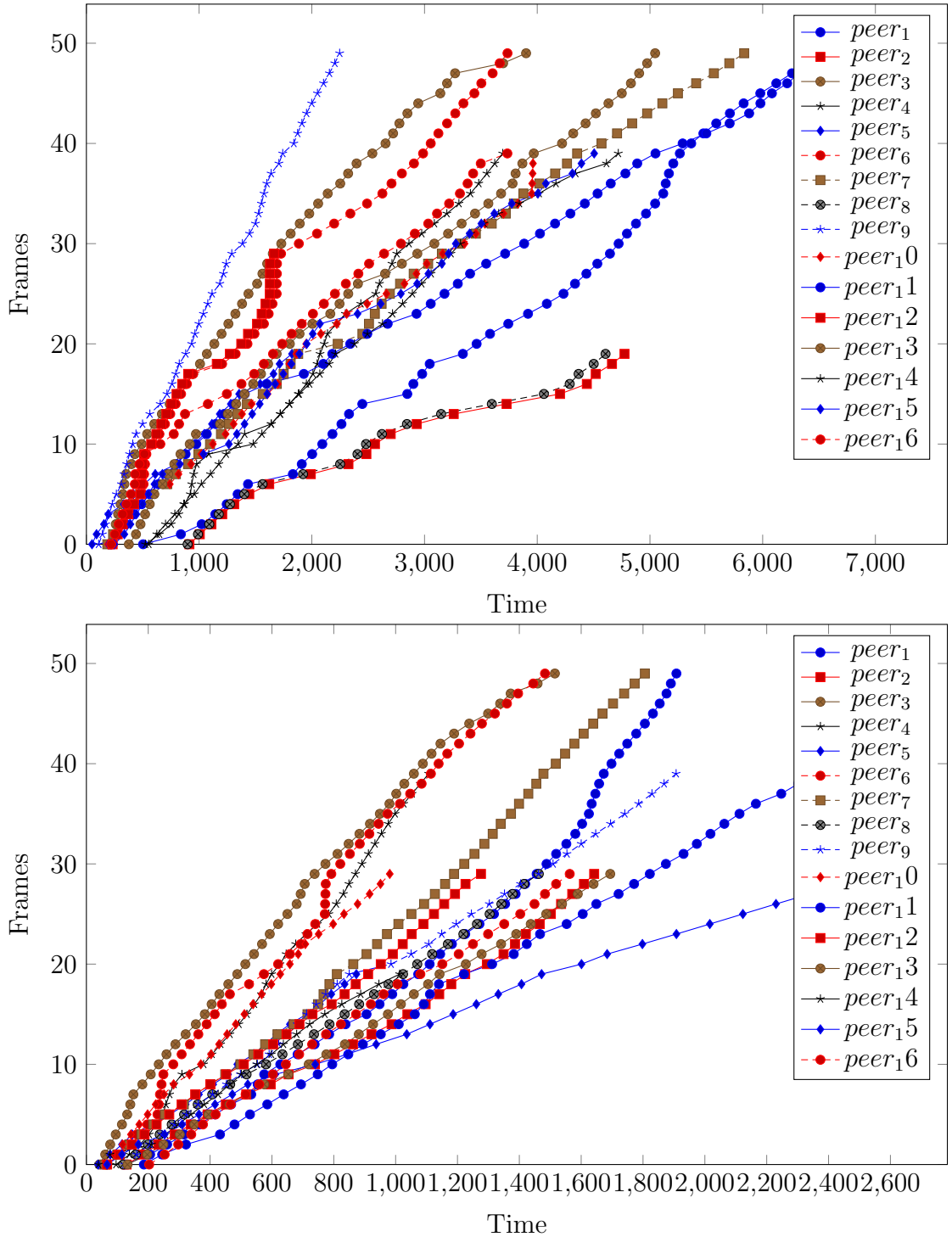


Figure 5.9: Comparison between the frame-rate in a 16 peer simultaneous scenario(Top) and 16 peers simultaneous scenario using 2PIC(Bottom).

Scene	Type	Speed-up 8	Speed-up 16
Town	simult	2.44	2.04
Inner Sanctum	simult	2.07	2.04
Town	stagg 400	2.51	2.11
Inner Sanctum	stagg 400	2.10	2.19
Town	stagg 800	2.51	2.15
Inner Sanctum	stagg 800	2.4	2.02

Table 5.9: Timing and performance between 8 peers and 16 peers for all scenarios.

with a peer inter-arrival time of 400 seconds and the second with an 800 second delay before a each new peer joins. In the first experiment, the Town scene saw a recorded speed-up of $\times$2.51 and the Inner Sanctum scene $\times$2.1 for 8 peers. When the number of peers was increased to 16, the Town scene recorded a speed-up of $\times$2.11 and $\times$2.19 on the Inner Sanctum. The results for the second experiment are equally positive, with over two-fold speed-up for both scenes, in 8 and 16 peer networks (see 5.9).

Table 5.9 highlights the two-fold improvement of the 2PIC with respect to the traditional IC, which, suffers from over-saturation as the number of collaborating peers on the network increased. The 2PIC not only improves the base speed-up seen in the original work by Bugeja et al. [15], but also sustains this two-fold improvement as the network grows - doubles.

5.4 Summary

This chapter introduced a novel method that extends the original IC to support large exchanges of samples that caused the data structure to over-saturate and led to performance degrading patterns. The 2PIC has not only been shown to improve on the traditional IC, but also to scale better in the context of collaborative rendering, increasing speed-up from amortisation to two-fold.

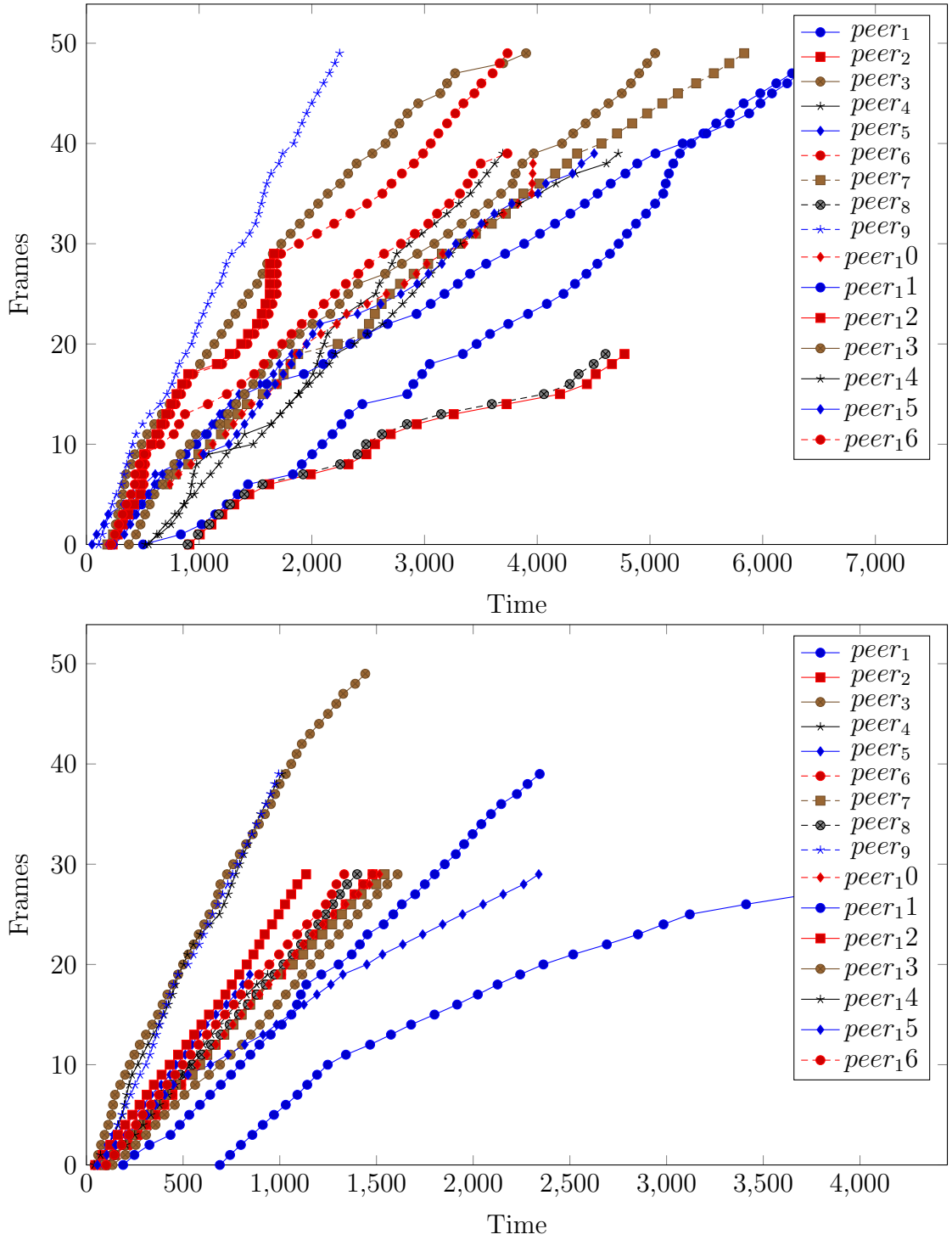


Figure 5.10: Frame-rate comparison, 16 peer staggered 800 collaboration using single IC(Top), 16 peer staggered 800 collaboration using 2PIC(Bottom).

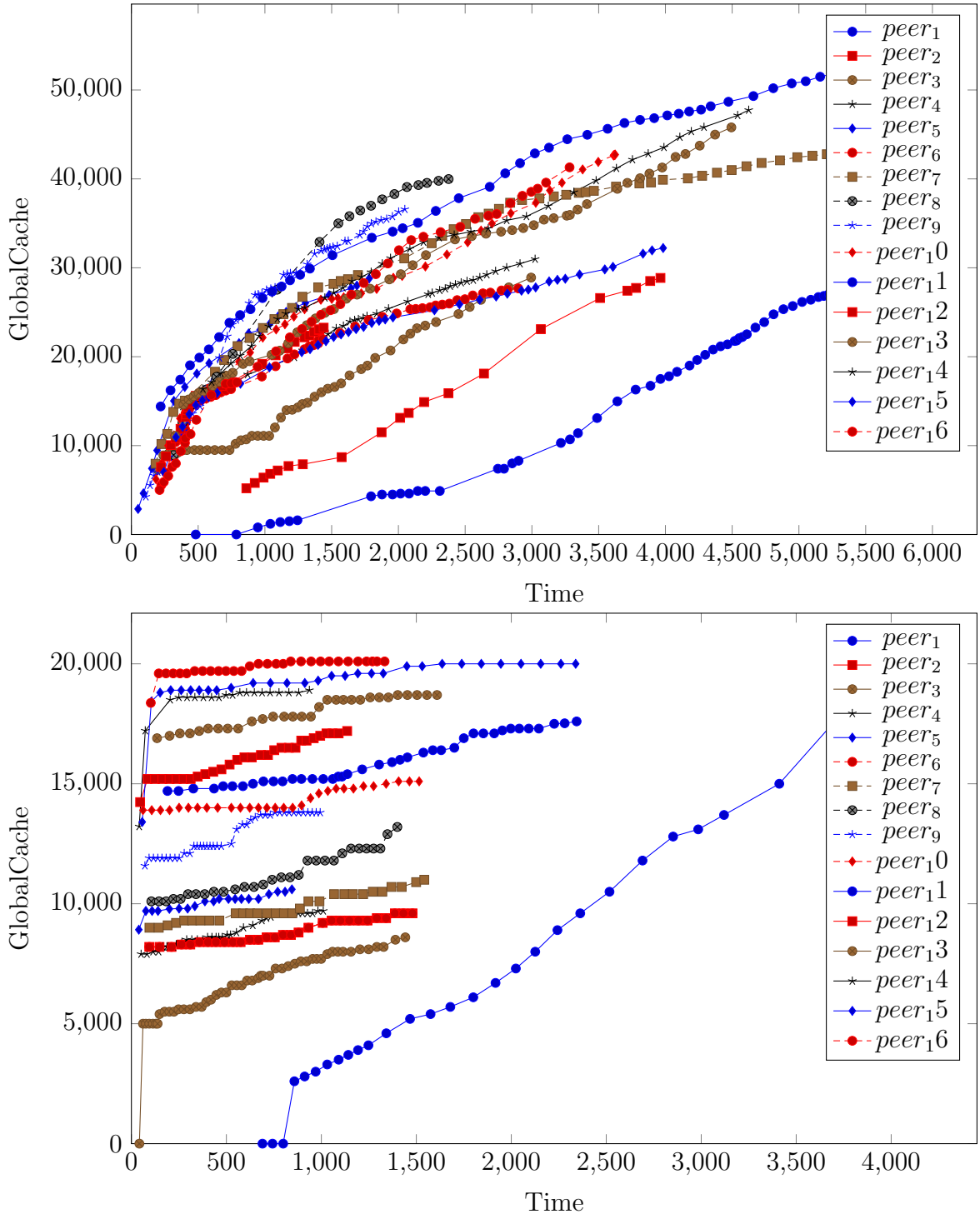


Figure 5.11: Comparison between the amount of samples exchanged over the network in a staggered 800 collaboration(Top usig single IC and bottom using 2PIC).

6. Context Aware Peer-to-Peer High-Fidelity Rendering

Epidemiological techniques can guarantee that at some point in time the updates that are sent by a peer will be delivered to all other peers on the network. Albeit guaranteed, events may not retain relevance by the time they are received by the peers, since the event would have been redundantly computed. Moreover, there exist *critical observable events* which are events that occur as a side-effect by a dynamic action that happened in the system, such as a flickering point light. Peers that are found in the area where the action is happening require the update event immediately, so that the lighting conditions and caches can be adapted. Even though peers found in a different area still require the update, they are not immediately affected by the sudden change. Thus, it is not required that the said peers receive the update in time to retain relevance.

In this chapter, a novel technique for an improved model of the dissemination of data will be evaluated through the use of a topic based publisher/subscriber overlay. The proposed method is both fault-tolerant and scalable; through the utilisation of an overlay made from microservers, meta-information is disseminated between the peers. Meta-information received by a peer, allows it to prioritise message passing, allowing for communication between neighbours at different frequencies.

The chapter is split as follows; Section 6.1 provides an introduction to the chap-

ter and Section 6.2 provides an overview of the novel method. The experimental setup is defined in Section 6.3, whereas the results are presented in Section 6.3. The outcome of the results is summarised in Section 6.5 while Section 6.7 provides a conclusion to the chapter.

6.1 Event dissemination problem

Peer to peer systems (P2P) provide resilience, scalability and fault tolerance to network systems. However, unlike their hybrid counterparts, unstructured P2P do not contain any central entity through which updates can be disseminated to all the other peers. In such cases, dissemination is implemented through the use of flooding, broadcasting or the use of random walk techniques [55]. Both flooding and broadcasting are efficient methods for the dissemination of data over a network. However, as the system scales, communication proves to be a bottleneck since the amount of traffic on the network increases[22]. To prevent this, Bugeja et al. [15] make use of epidemiological techniques; techniques that fall under the set of random walks. Epidemic techniques model data dissemination similarly to how diseases spread in the real world; when applied to unstructured networks they are capable of spreading an update to all the peers in a maximum number of $\log_2(n)$ steps, where n is the number of peers involved. Furthermore, each peer is not aware of all the elements interacting over the network and relies on the *small world* phenomena (see §3.5).

Their system does not share singular computed samples, yet it batches computed samples in groups of 100. These are known as observable events (OE). A subset of OEs are the critical observable events (COE). COE are events that are time dependant and might not retain relevance by the time the update has been shared between all the peers.

For improved data dissemination, the technique used by Bugeja et al. needs to be modified. Techniques have been proposed for the efficient dissemination of

data across an unstructured P2P system that makes use of an overlay. Similar to the approach used in Gnutella (see §3.5), the new techniques do not make use of flooding, yet, they maintain an overlay of privileged peers. The peers that are found in the overlay are named *publishers*, while the peers that connect to the publishers are known as *subscribers*. Thus, the application of the publisher/subscriber paradigm allows the dissemination of data produced by the publisher to interested subscribers. Publisher/Subscriber data dissemination techniques can be divided into two subsets; content based and topic based techniques. Topic based techniques make use of pre-defined topics and clients connect to the publisher that is publishing data related to a specific topic. On the other hand, content based techniques are more fine grained since the peers connect to publishers that are providing data that overlaps their spectrum of interests. Once the requirements change, the peers would then disconnect and connect to another publisher (if a said publisher exists). Each group of publisher and subscribers create another logical layer through which data can be separately exchanged. Techniques have been researched for the implementation of a publisher/subscriber paradigm for unstructured P2P systems [59]. More so, for ones that make use of an epidemic style data dissemination. This technique was extended further by Dattaz et al. [21], where their system allowed all peers found on the same overlay to become a publisher at some point in time. This was achieved by structuring the logical layer as a *directed acyclic graph*, whereas, Voulgaris et al. structured logical layers in a ring topology. Dattaz et al. name the current publisher a *non-sink node* while the current subscribers as *sink nodes*. This approach still makes use of the overlay to exchange data and, although network discovery may still rely on epidemic techniques, data dissemination occurs on the logical layers; where the updates exchanged on each layer concern only the topics currently 'discussed' in the overlay.

The method presented in this Chapter takes a completely decentralised for data dissemination. Throughout the execution of the system, an overlay of microservers is hosted, peers might subscribe to multiple microservers that offer a

different topic. No observable event is passed through the logical layer, on the contrary, the microservers will provide the subscribed peers with meta-information regarding their fellow subscribers. The received meta-information will allow the peers to disseminate updates with different frequencies; since through the use of the information received peers can prioritise data dissemination to their respective cached neighbours.

6.2 System overview

The proposed technique is based on the use of *contexts*. A context represents a state of a peer for a specific topic, such as, the position of the peer in the scene. Not all contexts can be used on their own. In fact, some contexts require coupling with other contexts so that they can be interpreted by a peer. As shown in Figure 6.1 and unlike *Context 2*, the data provided by context *Context 1* can be used on its own. Thus the meta-information sent by both contexts is intersected and coupled to provide the new context information. Dependability of context was created to make contexts data efficient and prevent redundant exchange of information.

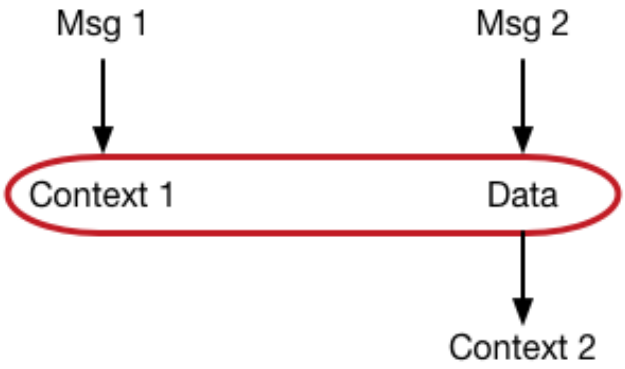


Figure 6.1: Dependability of contexts.

6.2.1 Overlay description

Each context represents a topic that can be hosted on the overlay, moreover, two contexts are considered equal when the two contexts host the same topic. Contexts can be hosted or decommissioned on demand. A context is hosted by a peer only when there is no other context of the same topic that is hosted elsewhere and that has accepted the said peer as a subscriber. On the other hand, if a context does not have any subscribers for a number of iterations, the said context is decommissioned. Discoverability of new contexts is handled in the base layer through the epidemic protocol that is used for the network discovery and event dissemination. Before the cache consistency procedure takes place, the two peers share the subscriber context details. Additionally, if the two peers have a different microserver for the same context, a method is required through which one can determine the preferred context for both peers. Similar to the observable events tie-breaker method, each context generates a UUID during the bootstrapping phase. When a peer is confronted with two contexts for the same topic, the peer uses the UUID to resolve the tie.

The contexts mechanism is handled by three separate threads, all of which reside on the peer node: the *context updater*, the *context initiator* and the *context listener*. The context updater periodically updates the contexts on which the peer is currently subscribed to. On the other hand, the context initializer is responsible for hosting contexts when a specified context is neither hosted locally nor hosted on another peer and referenced locally. Moreover, the context initiator takes care of decommissioning microservers that have been decommissioned as provided in Algorithm 12. The context initiator can only host one new topic for every period. Context instances are controlled through the use of a *context strategy* that is responsible for instantiating a particular object or returning an instance of an existing context, as required by the context initiator. The context listener's role is to receive meta-information updates, subscriber enrolments and unsubscriptions, all of which are enumerated below:

- Subscription request:

  A peer can only subscribe to a context hosted by another peer only if the micro-server replies positive to the subscription request issued. A microserver can deny a peer from joining the overlay if the maximum amount of subscribers is reached.

- Peer update message :

  This message is used to update the status of a peer at that specific topic.

- Unsubscribe message:

  Peers can only subscribe to a single context for a specific topic. If the tie-breaker favours the microserver that was shared by another peer, the first tries to subscribe to the new microserver and if it is accepted, it proceeds to unsubscribe from the previous microserver.

Moreover, each peer contains a context listener. This thread receives new updates from all the contexts for which the peer is registered to. Once an update U is received from a context (referring to the peer's neighbours cache as C), only $U \cap C$ will be updated with their new status. Peers that are not listed in the neighbour's cache will be ignored since the context is never used for network discovery purposes.

Algorithm 13 The *context initiator* pseudocode

```
1:  procedure ContextInitializer(bool* peerReady)
2:      while (!*peerReady) do
3:          for registeredContexts do
4:              if (!FindContext(index)) then
5:                  Host(index);
6:                  AddSuperPeer(index);
7:                  break;
8:              else
9:                  context → GetContext(index)
10:                 if context.HostedLocally()&context.IsDecommisioned() then
11:                     RemoveContext(context);
12:         this_thread.sleep(n)
```

Different devices have different internet bandwidth; tablets and mobiles may have lesser bandwidth than a high-end PC. For this reason, each peer contains a parameter called the *connection tier* that represents its internet connection quality. When a peer subscribes to a context, the peer also sends its respective connection tier. This allows the micro-server to discriminate certain peers upon an update and thus, send context updates less frequently. There exist five different connection tiers, ranging between one and five, where one represents a fast internet connection and five represents a slow internet connection. Thus, a peer having a connection tier of one is updated more frequently by the microserver, than one with a connection tier of five, which would receive an update every fifth of the time.

Upon start up, the context initializer starts hosting new contexts since no context references are found locally. A single peer can never host multiple contexts at once, rather, an incremental approach is taken. The context initializer periodically checks the state of locally hosted contexts and, if there is no reference to any microserver for a specific context, the context initializer would host the respective context. Once it is hosted, the context initializer is not be able to host another context until the next check iteration. This incremental approach allows one to prevent hosting of multiple contexts on a single peer at once, as to make use of already hosted contexts from the network. Furthermore, it is also assumed that until the next period, the peer has already communicated through the use of the epidemic protocol, and, new micro-servers have been discovered.

The overlay scales greedily, thus peers are grouped together when it is possible, to share the same microserver. Figure 6.2 shows how five peers hosting several microservers for the same two contexts consequently end up merging under two micro-servers, one for each context. The subscriber's limit for each context is assumed to be large enough to contain all the peers provided in the example. In case a micro-server can only contain a subset of the peers provided and is full, the new context would not accept new subscribers and the load is shared between two microservers.

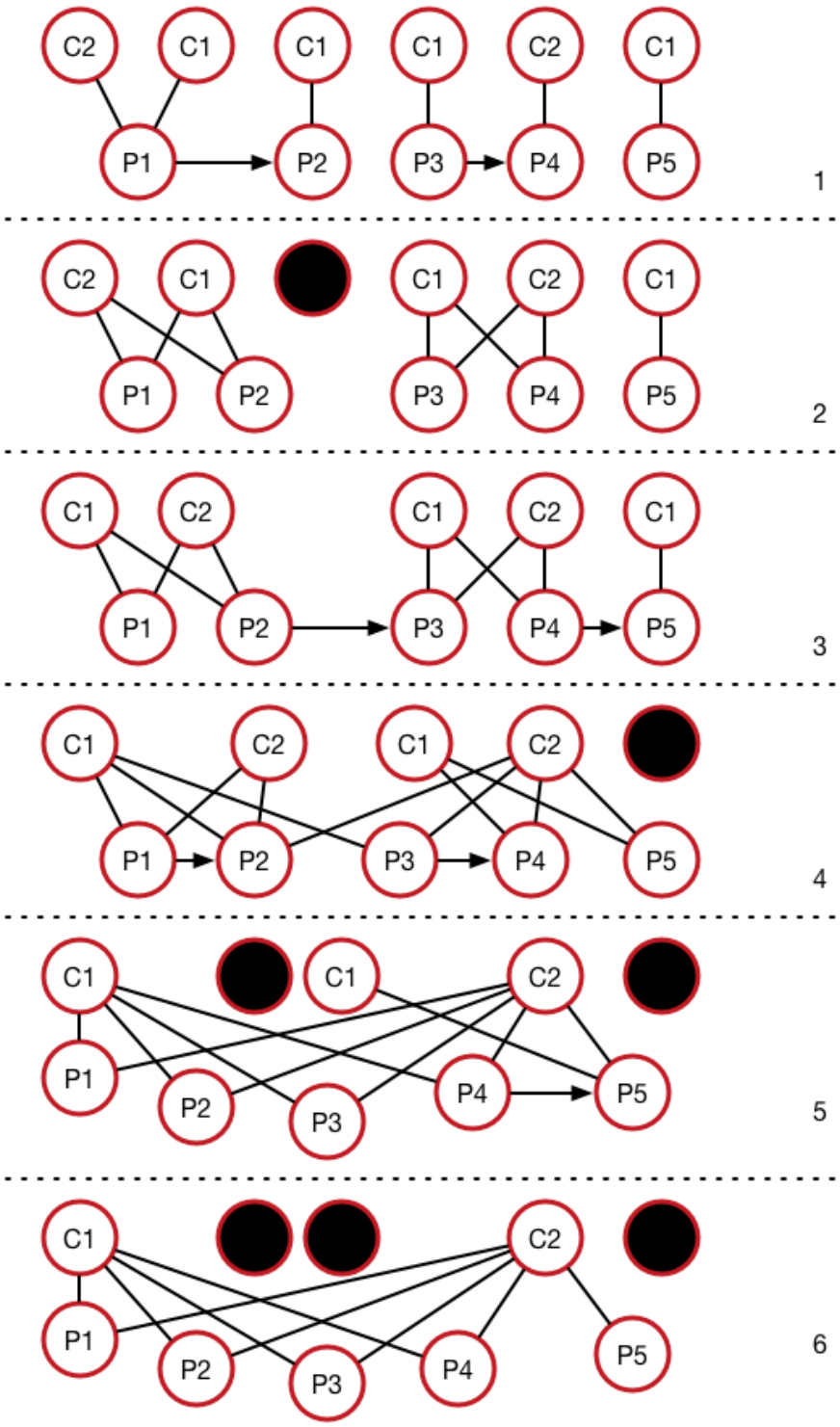


Figure 6.2: Microserver grouping for an overlay composed from two contexts.

Peers receive periodical updates from each micro-server. As stated previously, peers do not use the micro-servers for discovery purposes, thus, only information regarding cached neighbours is updated. Once the updates that are received are committed to the respective neighbours, the peer re-calculates the weighting for each peer found in the cache. The weighting function used to determine the weight of a peer is a summation of the weights for different contexts as shown in Equation 6.2.1.

$$w(x) = \sum_{j=0}^{j=N} w_j * p_j \tag{6.1}$$

The weight w_j signifies the priority of the respective context with regards to another context and is statically provided to the system. On the other hand p_j signifies the weight assigned to peer j. The weight assigned to the peers is based on an exponential function between 0.1 and 3 on the exponential distribution curve. Taking into consideration Figure 6.3, the image represents three peers interacting together in the scene. Through the use of the spatial position context, peer A receives the current position of both *neighbour 1* and *neighbour 2*. For this particular example, neighbour 1 is x units of distance away, while neighbour 2 is l units of distance away and $x < l$. Sorting the respective neighbours in ascending order with respect to the distance from A, the weight attributed to each peer is inversely proportional to their position in the list. This weight is then transferred to a probabilistic distribution function (PDF), that is used to apply importance sampling on the neighbour's cache. The resulting PDF affects the selection process for the data dissemination. Thus, upon selecting a random peer, certain peers have a greater probability of being selected. This is synonymous to communicating with different peers at different frequencies.

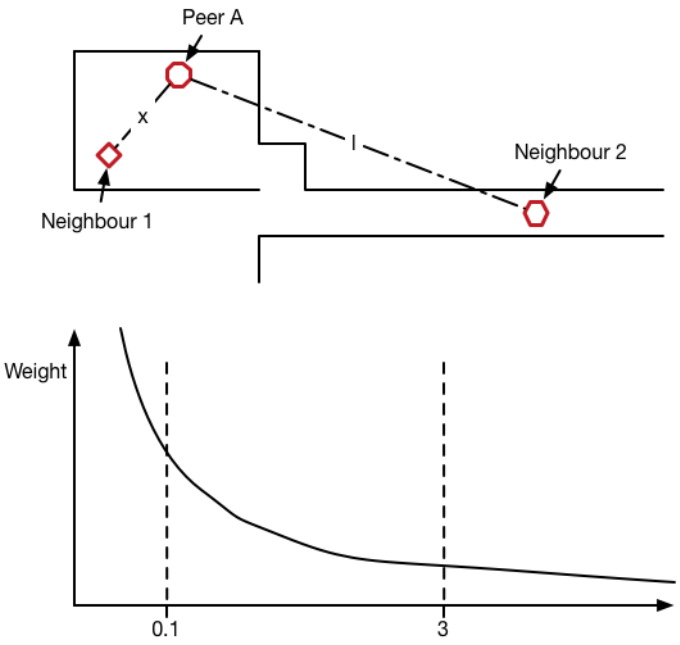


Figure 6.3: Spatial context example, *neighbour 1* gains a greater weight than *neighbour 2* since it is closer to *peer A*.

6.2.2 Different contexts

Different contexts can exploit different view of the peers status as to provide further information of the peers interacting in the system. For this reason, four different type of contexts where created, that are listed below and explained in further detail:

1. Spatial position:

   As stated in Section 6.2.1, the spatial position context provide prioritisation of the peers according to the spatial position in the scene. Closer peers have a tendency of providing samples required by the said peer at that specific time. Furthermore, this method might reduce the redundancy of computed samples, since selection is leaned towards peers that are near the respective peer.

2. Bounding Box index:

   The bounding box context provides a more coarse-grained spatial peer prioritisation. Unlike in the position context, the priority of the peers is deduced from the bounding box that the peers are currently located in. The bounding box context requires some form of pre-computation process to subdivide the scene into a series of bounding boxes. Moreover, the bounding box system can also be used for invalidation purposes. Partial invalidation requires bounding boxes to limit the branches of the IC need to be invalidated. As shown in Figure 6.4, the peer order is dependent on the distance from the centre of the peer's bounding box (labelled as P) to the respective neighbours bounding box centre.

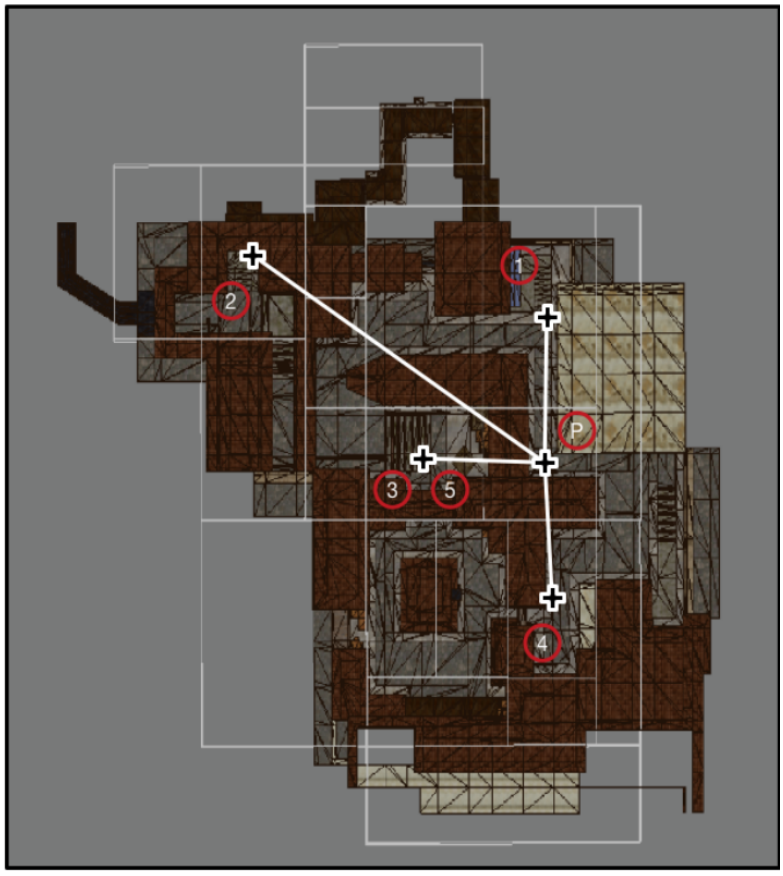


Figure 6.4: Bounding box context peer prioritisation technique.

3. Latency:

   Latency provides further insight of the amount of traffic occurring on the network. Higher latency between the peer and the respective microserver hosting the latency context might imply that there is lesser network speed from the peer. This way, peers can prioritise the frequency of communication with other peers according to traffic on the network and peers with higher latency gain a lower priority. Thus, the system attempts to prevent further congestion on the peers network. Figure 6.5 depicts this scenario where peers are ordered accordingly.

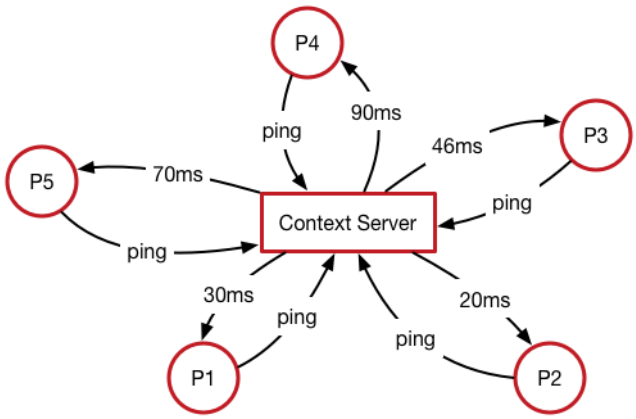


Figure 6.5: Latency involved in the contexts.

4. Viewpoint:

   The viewpoint represents the volume that the peer is currently rendering, as shown in Figure 6.6. The viewpoint is represented as a 3D frustum and for simplicity purposes, the viewing frustum was modelled as a triangle, thus, neglecting the use of the y-axis plane as shown in image 6.7.

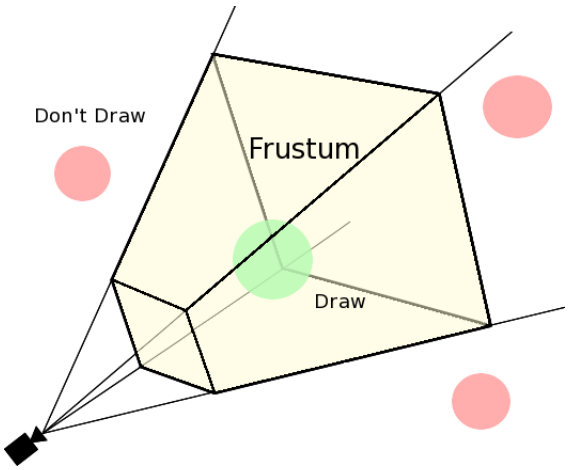


Figure 6.6: The 3D representation of the viewing system [2].

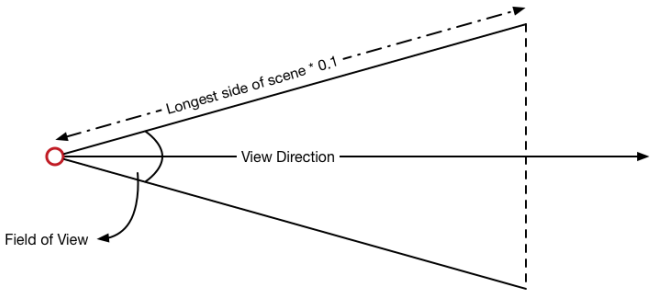


Figure 6.7: Simplified version of the viewing frustum.

The viewpoint context makes use of the current viewing direction of the peer. Albeit peers may reside near the said peer, they may not provide samples that are required by the peer for its current frame. Figure 6.8 provides an example of 4 peers (including the peer computing the prioritisation) interacting together. Although all peers may provide data that is in the vicinity, only peer 1 and 2 may provide data that the peer actually requires at the moment. Thus, this context may reduce the amount of redundant points computed by the peers. The prioritisation of the neighbours occurs according to the overlapping area between the frustas and ordered in descending order according to the overlapping area between the two corresponding frustas. Non-overlapping peers are not given any weighting for this context. This context depends on

the spatial position context, since the construction of the simplified frustas requires the spatial position of the peers involved.

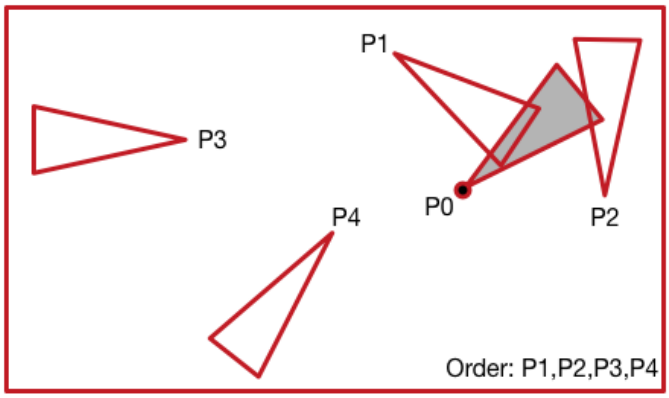


Figure 6.8: Prioritisation of the peers according to the amount of intersection happening between the viewing frustas.

6.2.3 Observable events (OE)

The use of an overlay provides the capability to disseminate updates in a more efficient manner. This enables peers to propagate critical observable events and observable events, in general, more efficiently. Invalidation events are OE that signal the receiver about a sudden change in the lighting conditions of the scene, thus, all the stored IC samples are required to be invalidated and recalculated on demand. On the other hand, partial invalidation events nullify only a subset of the IC. Partial invalidation is achieved through the use of the bounding boxes, the same bounding boxes structure that is used for the bounding box context (see Context 6.2.2). The bounds for partial invalidation are calculated by selecting the bounding box structure that contains the position where the invalidation occurred. The same bounding box is then used by the receiver to invalidate intersecting octree nodes in the IC. Similarly, to full invalidation, partial invalidation occurs because of a dynamic interaction in the scene. Thus, another OE type is required to represent

scene changes and the *scene event* was introduced for this purpose.

Observable events ordering

Observable events are ordered and merged into the respective peer's local state through the use of two forms of updates, the insertion OE E_{ins} and the invalidation OE E_{inv}. As stated by Bugeja [14], insertion events do not necessarily require any order when a sequence of such events are committed to the IC. Moreover, if the vector clock of the two events cannot be ordered, the OEs are considered as concurrent and require the use of a *tie-breaker* function. The tie-breaker function determines the order of the two events based on their respective UUIDs. The UUID is said to be unique with a high-probability. This identifier is capable of providing global order to events that have been deemed concurrent. Similarly, invalidation events can be ordered in the same manner that insertion events are ordered. The author introduces a composition operator for the types mentioned above, which is listed in Equation 6.2.3. Any two events of the same type, can be collapsed into a single event of the same kind. Furthermore, contiguous invalidation events do not provide any further change to the IC structure and their composition prevent unnecessary operations on the cache.

$$e_m^{obs} \circ e_n^{obs} = e_n^{obs} \circ e_m^{obs} = \begin{cases} e_n^{obs} & e_m^{obs} \implies e_n^{obs} \\ e_m^{obs} & otherwise \end{cases} \tag{6.2}$$

On the other hand, there exist scenarios where two consecutive events of different types can be composed into a single one, such as, $E_{ins}^n \Rightarrow E_{inv}^m$ which represents an insertion that is followed by an invalidation event, that collapses to E_{inv}^m. The change in the state produced by an insertion is made redundant by the invalidation. Consequently, in the previous example, E_{ins} followed by E_{inv} results in an E_{inv}. On the other hand, switching the sequence of the events, that is, first, an invalidation E_{inv} occurs followed by an insertion. An insertion committed post

and invalidation would re-change the invalidated state of the cache by inserting of new samples, thus, this order of events cannot be composed. Moreover, Bugeja categorises the composition of $E^m_{inv} \Rightarrow E^n_{ins}$ as undefined.

Taking into consideration an ordered list of events containing both invalidation and insertion events, the rules 12 will collapse a sequence of received events to a lesser sequence that still retains the same semantic equivalence [14]. The rules defined below specify the compositional behaviour that have been mentioned so far.

Rule 1: $\frac{E_{ins}, E_{ins}}{E_{ins}}$ (using compositional rule on events of type insertion)
Rule 2: $\frac{E_{inv}, E_{inv}}{E_{inv}}$ (using compositional rule on events of type invalidation)
Rule 3: $\frac{E_{inv}, E_{inv}}{E_{inv}}$ (using compositional rule on insertion followed by invalidation)

After two peers communicate and exchange observable events, a mutual list of events would be contained by each peer, both of which sustain the same ordering. Through the use of the above-mentioned rules, the sequence of events can be minimised while still resulting in the same final state of the IC. Bugeja provides an example of a sequence of received events that have been received after exchange of events between the two peers. The example provided by Bugeja represents E_{ins} as a while E_{inv} are represented as b to the string sequence $[aaaabbbaabaa]$. Composing the string from right to left through the use of the compositional rules, the said string representation could be evaluated to $[ba]$.

Partial invalidation event

The compositional rules defined in [14] do not take into account partial invalidations E^n_{pinv} (^{n} representing the bounds on the invalidation). On the contrary of E_{inv} and E_{ins}, same event type composition do not always apply to events of type E^n_{pinv}, since $E^n_{pinv} \Rightarrow E^m_{pinv}$ affect the IC in a different manner. On the other hand, consecutive partial invalidations of the form $E^n_{pinv} \Rightarrow E^n_{pinv}$ can be composed to E^n_{pinv} since only the first partial invalidation changes the state of the IC in a se-

quence of such invalidations.

Rule 4: $\frac{E^n_{pinv}, E^n_{pinv}}{E^n_{pinv}}$
(applying compositional rule to partial invalidations on events with the same bounds)

Partial invalidations do not only be collapse to other events of the same type but also to events of type invalidation and insertion. Unlike invalidation, an insertion of the type $E_{ins} \rightarrow E^n_{pinv}$ may still retain relevance following a partial invalidation. Thus, this behaviour can also be categorised as undefined. Similarly, $E^n_{pinv} \Rightarrow E_{ins}$ also returns an undefined behaviour with the same reasoning that is applied to $E_{inv} \Rightarrow E_{ins}$. Thus, at no point in time can partial invalidations and insertions be composed and reduced to a single event.

It can be noted/realised that $E^n_{pinv} \subset E_{inv}$, since E^n_{pinv} is invalidating a small subset of the cache while E_{inv} is invalidating the whole cache indiscriminately, thus, $E^n_{pinv} \Rightarrow E_{inv}$ results in E_{inv}, since the partial invalidation become redundant. Similarly, $E_{inv} \Rightarrow E^n_{pinv}$ still results in a composition to E_{inv}, since following an invalidation event, partial invalidation is redundant. Therefore, two new rules can be established about the composition of partial events in relation to other event types.

Rule 5: $\frac{E^n_{pinv}, E_{inv}}{E_{inv}}$
(applying composition of invalidation followed by partial invalidation)

Rule 6: $\frac{E_{inv}, E^n_{pinv}}{E_{inv}}$
(applying composition of partial invalidations followed by invalidation)

The composition example provided by Bugeja was extended to include partial invalidations, the following initial list of events was used $[aaad^1aabbbd^1aabad^1a]$ where a represents an E_{ins}, b represents E_{inv} and d^n represents a partial invalidation for a bounding box index n. The resulting composition, shown in Figure 6.9

simplifies to a sequence containing $[bad^1a]$. As shown in the example, the reduced string is semantically equivalent to the initial series, since everything that follows the last invalidation has to be committed. On the other hand, since there are no rules that can be applied to the composition of insertion and partial invalidation, the sequence $[ad^1a]$ could not be composed and needs to be committed.

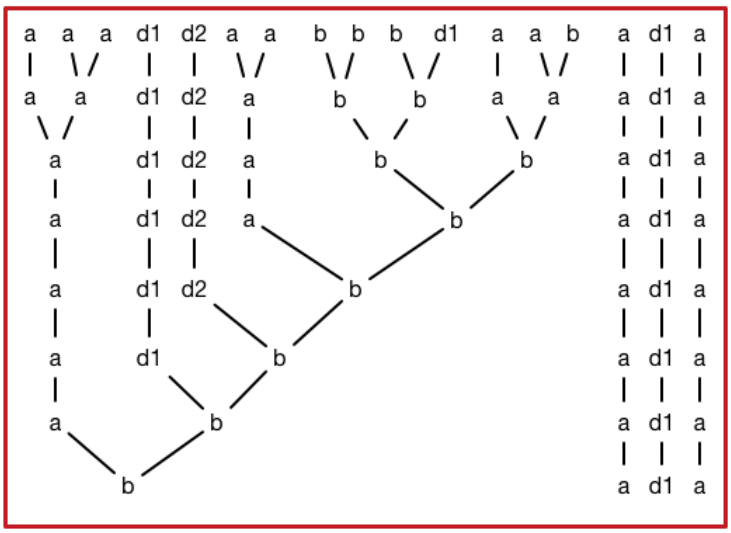


Figure 6.9: Composition of a list of events received from an interaction where a represents E_{ins}, b represents E_{inv} and $d(n)$ represents E^n_{pinv}.

Scene events are treated differently since they depend on E_{inv} and E^n_{pinv} to be committed. Thus, the composition of such events is regarded as redundant since they do not contribute to the state of the IC and the scene without the respective event reference.

6.3 Experimental setup

Experimental runs analyse the performance achieved through the use of contexts for amortised rendering. For these experimental runs, 16 peers were employed in a simultaneous scenario. It was observed that a larger amount of samples is

Run Number	Position	ViewPoint	Bounding Box	Latency
1	✓	✓	✓	✓
2	-	-	✓	-
3	✓	-	-	✓
4	-	✓	✓	-
5	✓	-	-	-
6	✓	-	✓	✓
7	✓	✓	-	✓
8	✓	-	-	✓
9	✓	✓	-	-

Table 6.1: Contexts combinations run on the Town scene using the 16 peers in a simultaneous scenario.

exchanged when contexts are used. For this reason, all the context runs were setup to utilise the 2PIC technique such that over-saturation of the individual caches does not affect the final performance. Additionally, recorded timings were not be compared to timings from quiescent runs (see §4.3.1) but were compared to 2PIC simultaneous runs for 16 peers. Different combinations of contexts were utilised to analyse the effectiveness of the individual contexts. The combinations of contexts that were executed are tabulated in Table 6.3. The capactiy of each microserver was set to 8, reflecting the size of the peer neighbours cache.

6.4 Results

The set-up for the performance analysis of the contexts overlay was set as explained in Section 6.3 and tests were run using 16 peers. Each experimental scenario was run twice, and the mean performance was extrapolated. As illustrated in Table 6.3, the viewpoint context was never tested without including the spatial position context, since it is dependent on the spatial position context (see §6.2.2). Moreover, latency on its own was never used for experimental runs. Since all the peers are

Run Number	Speedup
1	0.98
2	1.04
3	0.99
4	1.00
5	1.00
6	1.00
7	1.00
8	0.99
9	0.99

Table 6.2: Contexts combinations run on the Town scene, using the 16 peers in a simultaneous scenario.

found on the same LAN, ping queries do not return differing latencies, and therefore reduce to a random distribution PDF. The priority weighting w_j for each context was set to 0.2 to limit the number of variables in the experiments.

Table 6.4, shows the speedup gained when the contexts combinations specified in Section 6.3 were run. The individual performance of the peers ranged from a decrease in performance of 40% to a two-fold speedup over the timing gained from 2PIC. Figures 6.4, provide further insight of how the global cache of the peers changes according to the context which was applied on the network, while Figures 6.4 highlight the change in frame-rates when the contexts are applied.

6.5 Discussion

Contexts did not improve the overall performance of the collaboration. Some peers saw a speed-up, while others suffered a slow-down which was at worst 40% that of 2PIC in the simultaneous start scenario with 16 peers. Some peers were able to gain a two-fold speed-up using spatial contexts space (spatial position and bounding box contexts). Figures 6.5, illustrates the case when spatial position and bounding

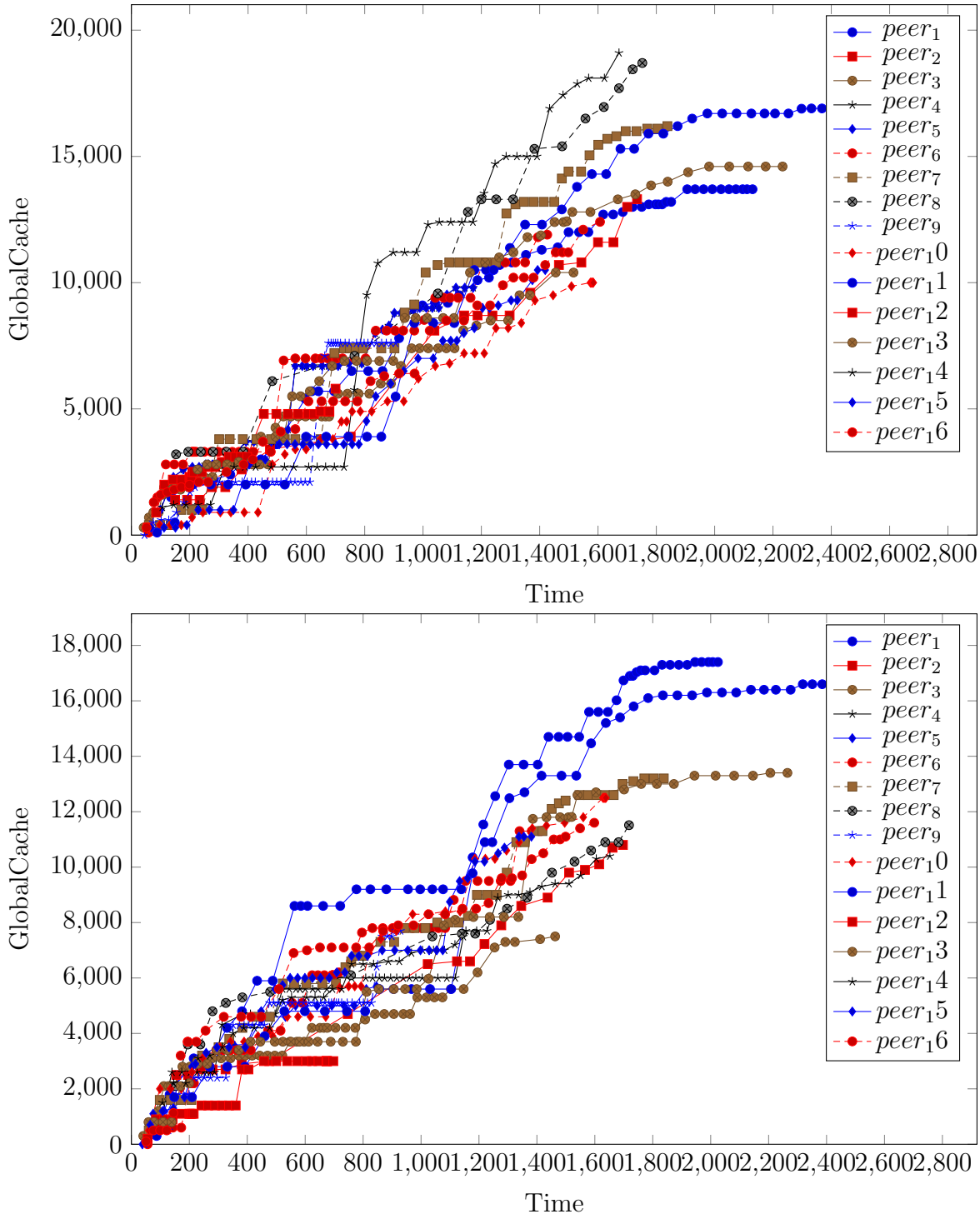


Figure 6.10: Comparison between the size of the global cache of the peers in the simultaneous 16 peer runs for the contexts Bounding Box run (Top) and spatial position(Bottom).

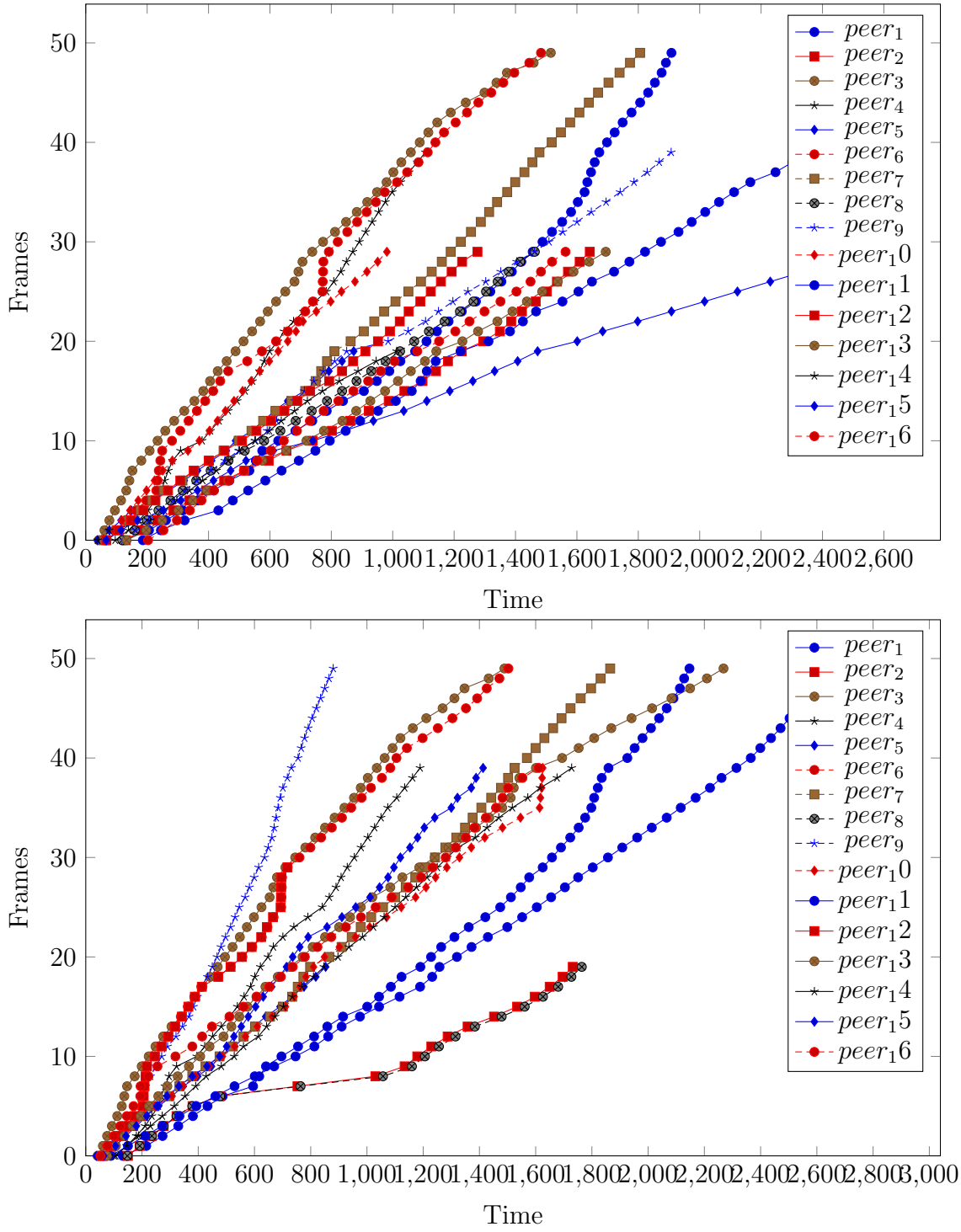


Figure 6.11: Comparison between the frame-rates achieved between the 2PIC simultaneous 16 peers(Top) and viewpoint context simultaneous 16 peers.

boxes were combined and when the peers were able to retrieve a substantial amount of samples from the network. Priorities generated by the spatial position context are fine-grained when compared to the bounding box context, which on its own allowed approximately the same number of samples to be retrieved as when used in combination with another context. Moreover, updates were more likely to be evenly distributed among peers in the same bounding box, although not all might have required the information received. When the two contexts were combined, the performance of the respective peers was rebalanced out.

The viewpoint context was not effective. Since the scene was not densely populated, the intersection of the peer's viewpoint was minimal. Although paths traversed by the peers overlap, the timing of the overlap must happen such that an intersection between the two view frusta can take place. If no intersection occurs, the peers would be provided with a weight of zero for the respective context. However, upon sorting the neighbours cache, the resulting sorted list of peers would be randomly sorted, allowing peers to gain less advantage from the network. Figure 6.5(bottom), shows that the same peers that showed a performance improvement from the spatial and bounding box contexts were still able to benefit from the viewpoint-spatial combination of contexts.

No overall drop in performance was recorded compared to the 2PIC technique, thus, retaining the two-fold speed-up. The new dissemination method suggests that it might be able to provide necessary updates to the peers that require them, as opposed to the original technique that made use of random walks to disseminate information on an unstructured P2P network.

6.6 Limitations and Future Work

The context priority weighting w_j (see §6.3) was empirically set at the start and was set to be equal for all contexts, such that, fewer variables would have to be taken into account and all context had equal priority. Specific contexts did not

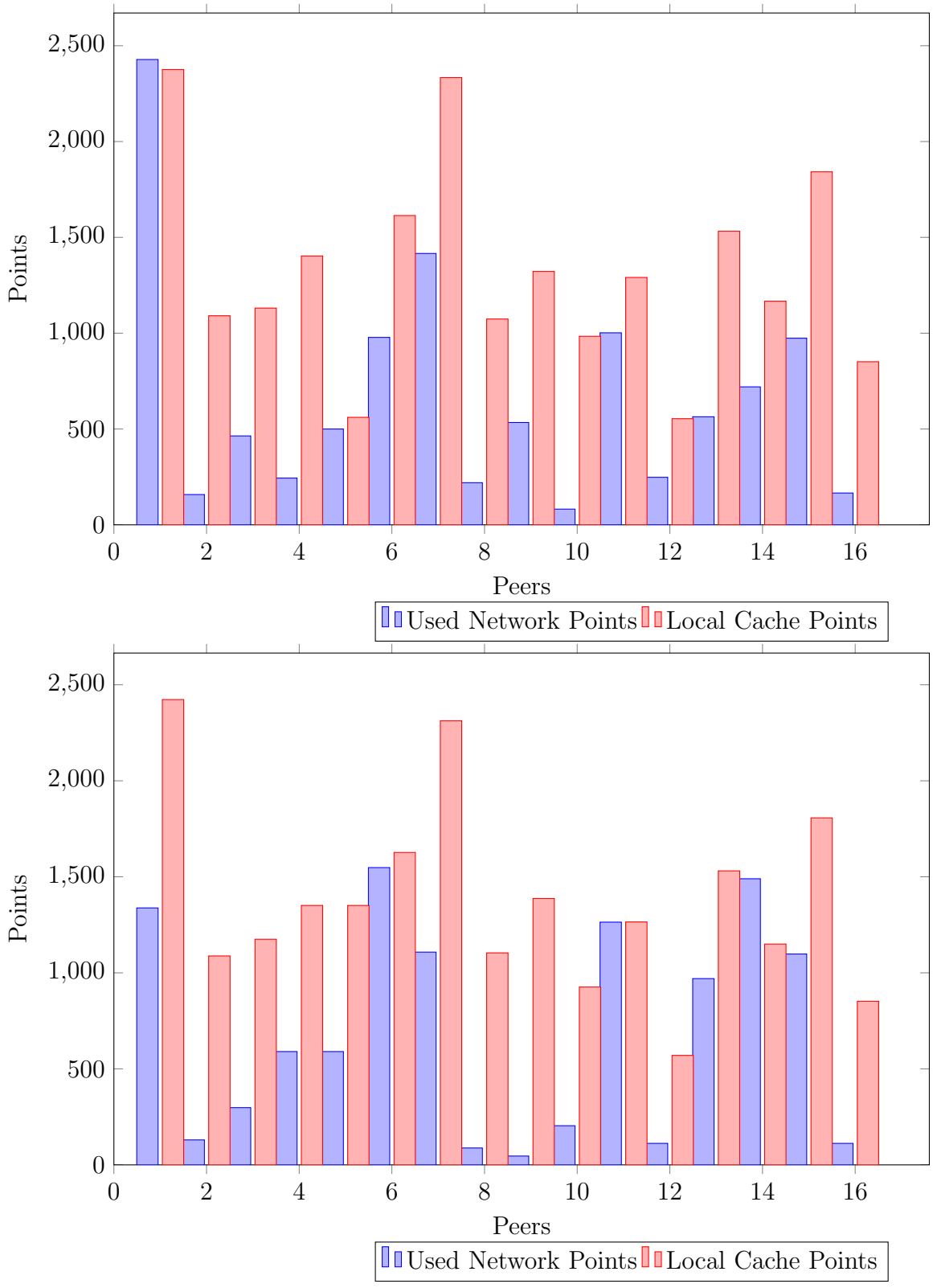


Figure 6.12: Ratio of the used samples in the application of the spatial position context(Top) and the application of the bounding box context(Bottom).

perform as well as the others since they did not deliver the expected improvement. The introduction of dynamic weighting allows contexts that are not immediately useful being to have their priority set to 0. Thus, the resulting PDF would only be affected by the context that can provide a difference.

The tie-breaker function used to decide which microserver retains the context is decided by the peers through the use of their respective UUID. Albeit deterministic, old microservers privilege to retain a context is only dependent on the generated UUID. Thus, later, a new context may be hosted which may have a smaller UUID, resulting in a scenario where the newly hosted microservers may still be preferred over old microservers. This would lead to scenarios where a microserver that is fully subscribed ends up losing peers, because they are subscribing to the newer microserver, resulting in loss of critical information. The implementation of vector clocks allows subscribers to determine the preference of the microserver based upon the time stamp. In the case that two microservers result are concurrent, a tie-breaker function would then still be able to break a tie deterministically through the use of the UUID of the two microservers.

The contexts utilised relied on the peer status, such as the current position or the current viewpoint of the peer. There was no context taking into account the history of a peer's path. Utilising the peer's history another peer can prioritise peers according to the peers that their path intersects with the current position of the respective peer. Unlike position, it does not make use of the current status of the peer, eliminating the possibility of computing redundant samples.

6.7 Summary

This chapter proposed a topic based anonymous publisher/subscriber for an improved data dissemination over an unstructured P2P system. The technique relied on an overlay composed of multiple microservers. The proposed technique is both fault-tolerant and decentralised, allowing the unstructured P2P system to retain its

properties. Experiments showed that overall, performance improvement was not gained from the overlay, it rather shifted to different peers, allowing peers to benefit from a two-fold improvement over the speedup gained from 2PIC, while other peers had a performance decrease of at maximum 40%. Results suggest that there was no overall performance drop when compared to the 2PIC. The new paradigm suggests that peers received updates that they require.

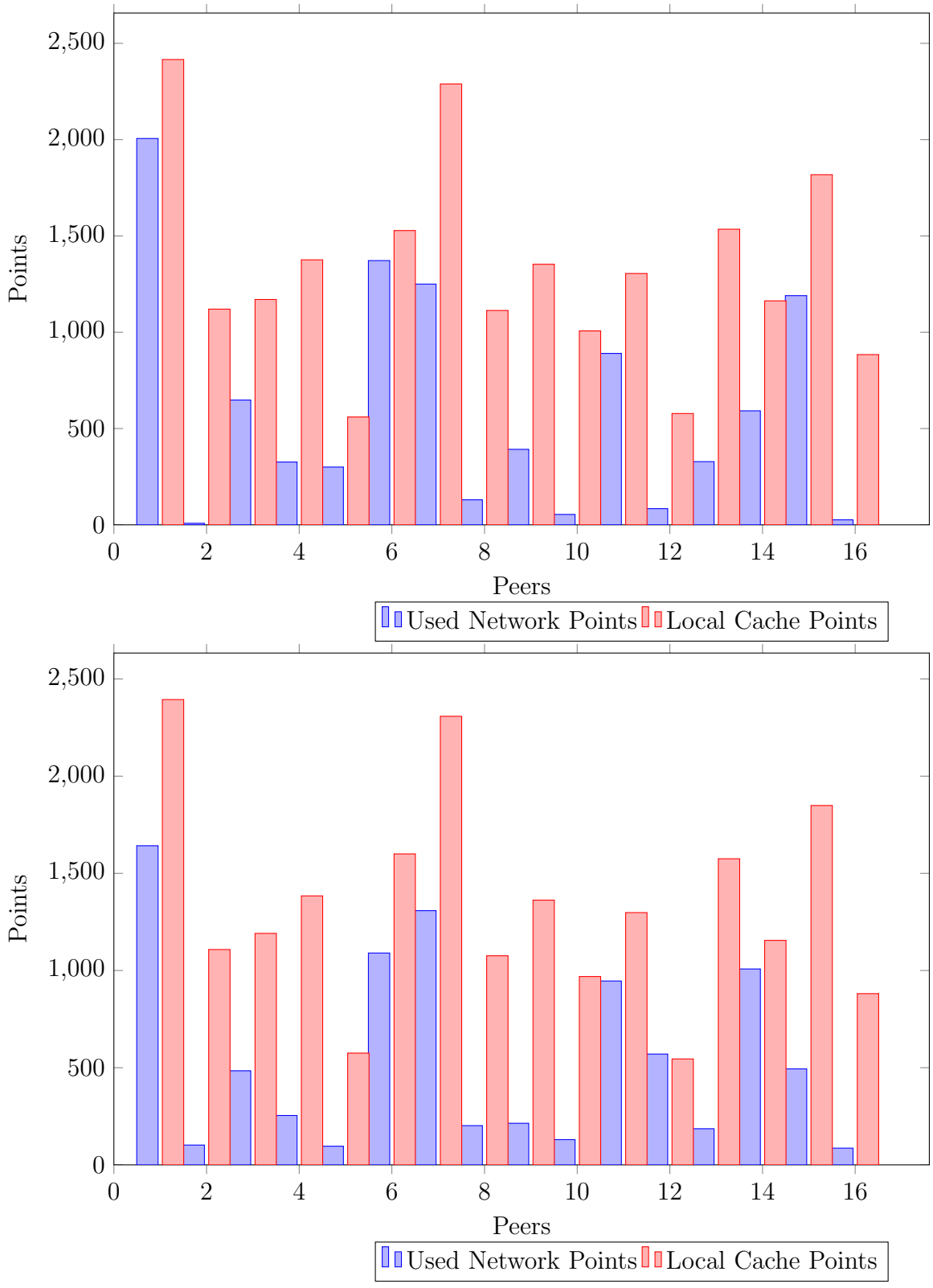


Figure 6.13: (Top)Combination of the bounding box and spatial position contexts. (Bottom)Combination of the spatial position and the viewpoint contexts.

7. Conclusion

High-fidelity rendering has witnessed widespread adoption in a number of disciplines and areas such engineering, archaeology, and the entertainment industry with video games and special effects amongst others. The pervasiveness of high fidelity rendering is such that any improvements to the fundamental techniques employed will see the benefits propagate to these areas of application. The focus of this work has been the collaborative aspect of distributed rendering, where given a number of users navigating a virtual environment, computation is shared among them to improve rendering time through amortisation, that is, the elimination of redundant computation. This chapter is structured as follows: Section 7.1 highlights the contributions of this thesis; Section 7.2 discusses the significance and impact of this work. Finally, Section 7.3 addresses the limitations of the system and proposes possible avenues for future work.

7.1 Contributions

The primary contribution of this dissertation is demonstrating that collaborative high-fidelity rendering over peer-to-peer systems is a viable alternative to traditional client-server approaches. More specifically, issues of scalability that adversely affected previous work [15] have been shown to be related to the rendering technique (see Chapter 4) rather than the networking paradigm used for communication; ad-

dressing the shortcomings of the employed rendering technique (see Chapter 5) has seen a marked improvement in system scalability and rendering performance. The epidemiological nature of information propagation does not prioritise content: peers do not receive events in order of importance. This behaviour is investigated in Chapter **??** and a context-based approach introduced to mitigate the problem.

7.1.1 Over-saturation of the Irradiance Cache

Bugeja et al. [15] hypothesise that amortisation in their collaborative peer-to-peer system suffers as the irradiance cache becomes over-saturated with samples; however, they do not test the claim. Over-saturation is a side-effect of the epidemiological techniques used for propagating information, which fills a peer's irradiance cache with samples that the peer itself might never need or use. The previous study has been replicated, dropping the ecological validity of the original experiments for a more controlled environment of heterogeneous machines; the new experiments test the claim of Bugeja et al. by scaling the number of peers from 8 to 16. The results show that as the number of peers increases, redundantly computed samples also increase. In particular, there is a 20-50% increase in the number of redundant samples, leading to a degradation in rendering performance.

List of Contributions

- peer-to-peer collaborative rendering benchmarks on a homogeneous network for up to 16 peers
- validate the claim that redundant samples are the cause for over-saturation of the irradiance cache

7.1.2 Two-phase Irradiance Cache

Over-saturation of the irradiance cache leads to poor rendering performance, reducing the overall network speed-up of the system. Insertions and searches of the

irradiance cache, which are computed hundreds of thousands of times per frame degrade the rendering performance when the cache is full of redundant samples, due to less optimal utilisation of CPU caches and an increase in interpolation times. The introduction of the two-phase irradiance cache (2PIC) has partitioned the traditional irradiance cache data structure into two: a global cache for storing samples acquired over the network and a local cache for use during local frame rendering. An image quality analysis of the 2PIC has been carried out, to quantify image degradation when compared to the original IC. Furthermore, the performance overhead of the technique has been assessed and a quality-performance function provided to allow setting optimal operating parameters for the technique when applied to peer-to-peer collaborative rendering. Accordingly, the experiments of Chapter 4 have been performed again with the 2PIC enabled, demonstrating a two-fold speed-up across board.

List of Contributions

- a novel method for reducing over-saturation during collaborative rendering
- a quality-performance evaluation of the two-phase irradiance cache (2PIC)
- a intuitive method for determining the operating parameters of the 2PIC for optimal rendering

7.1.3 Context-Aware Rendering

Epidemiological techniques employed in collaborative peer-to-peer rendering do not guarantee important events to be received as soon as peers require them. To address this limitation and improve update propagation, a context-based dissemination method was introduced wherein peers use meta-information to direct event updates. In particular, contexts provide information about peer attributes such as position, allowing the prioritisation of message distribution based on these attributes. The results show that context-aware rendering does not incur any re-

alisation penalties, with network performance still showing a two-fold speed-up. Furthermore, a higher update frequency between peers sharing the same contexts suggests that the method may be capable of providing more focused updates, possessing a greater relevance for the receiving peer.

List of Contributions

- an extension to epidemiological event propagation based on non-uniform probability distribution functions to provide directed dissemination in collaborative peer-to-peer rendering

7.2 Impact

Although the focus of this work is primarily that of improving collaborative peer-to-peer rendering, context-aware propagation may have applications outside of computer graphics, like for example, distributed database management systems. Within the remit of peer-to-peer rendering, directed dissemination can be used to share computation results for rendering techniques other than the irradiance cache. The two-phase irradiance cache, employed here for collaborative rendering, can be used in typical distributed rendering settings such as render farms, to accelerate off-line processes such as rendering an animation sequence. Finally, this dissertation focused on high-fidelity rendering, but it may be extended to other areas of computer graphics that might benefit from fault-tolerant systems and collaboration, such as non-photorealistic rendering.

7.3 Limitations and Future Work

This section outlines some limitations of the work presented, together with possible solutions and avenues for future work. Two sets of limitations have been identified: (i) system implementation, primarily the high-fidelity renderer; (ii) design decisions

on the application of contexts.

The performance of the rendering system was found to be greatly underwhelming when compared to the system used by Bugeja et al. [15]. The renderer uses the bag of tasks technique and image space subdivision, to create equally-sized tiles that are scheduled for rendering. Adding different tile sizes would ensure more granular load balancing during image synthesis, possibly mitigating CPU core idle times.

Contexts have been designed to exploit a peer's current attributes such as position or viewpoint; however, historical information such as the path traversed by the peer so far could provide additional value to the system. Path history, for instance, may provide a peer with results computed by another peer at some point in its past, provided these results have not been invalidated. The idea of context based on historical information opens up another possible avenue of further research for investigation.

The system supports individual weights for combined contexts, although, the effects have not been studied: weights are empirically set such that each context contributes equally in a composition. An interesting avenue for future work would be the introduction of dynamically-weighted contexts, which change during the lifetime of the application to provide the best context weights for the network configuration at the given time. Machine learning techniques could also be employed to help learn which contexts are more effective in a given situation or network configuration, and set the weights accordingly.

The irradiance cache is used to compute diffuse interreflections, which coupled with distributed ray tracing, provide the final image synthesis. The IC's use of irradiance samples in object-space, and their applicability to view-independent settings, allows them to be exchanged between peers without restrictions. However, the IC is not the only rendering algorithm that is view-independent; there exist other techniques that make use of object-space computations, such as finite element radiosity. Peer-to-peer rendering could be extended to use radiosity, with one

marked advantage over the irradiance cache: finite elements can be decided a priori using a deterministic algorithm, which means that during exchanges between peers, only the exitant radiance need be shared since each peer will already have the same list of patches computed - contrast this to the irradiance cache, where irradiance samples are based on points that are computed on demand and differ from peer to peer.

The context-aware dissemination results, although not conclusive, are promising enough to warrant further research. The results suggest that updates are more frequently carried out among peers subscribing to the same contexts; however, further studies need to be carried out to ascertain that this is indeed the case and that the exchanges between peers are meaningful with respect to topical updates. The problem of directed dissemination in peer-to-peer networks is not unique to computer graphics and thus, another potential interesting area for future work is the extension and application of context-aware dissemination to other distributed programs such as database management systems.

7.4 Final Remarks

The field of computer graphics, in particular high-fidelity rendering, has made quantum leaps forwards, towards achieving photorealistic rendering of images, in part due to the growth of parallel and distributed technologies. The methods presented in this work continue in the tradition of parallel and distributed rendering, albeit taking the unconventional approach of using collaboration to enhance peer performance through amortisation. This dissertation has addressed a number of important challenges in collaborative rendering, providing a firm foundation from which future research can build.

References

[1] Different materials light interaction.

[2] Frustum.

[3] Ray tracing concepts.

[4] Autodesk, 1982.

[5] Onlive, 2003.

[6] blender, 2004.

[7] Burp-boinc project, 2004.

[8] Render rocket, 2004.

[9] Playstation now, 2014.

[10] V. Aggarwal, K. Debattista, T. Bashford-Rogers, P. Dubla, and A. Chalmers. High-fidelity interactive rendering on desktop grids. *IEEE Comput. Graph. Appl.*, 32(3):24–36, May 2012.

[11] V. Aggarwal, K. Debattista, P. Dubla, T. Bashford-Rogers, and A. Chalmers. Time-constrained high-fidelity rendering on local desktop grids. In *Proceedings of the 9th Eurographics Conference on Parallel Graphics and Visualization*, EG PGV'09, pages 103–110, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[12] D. P. Anderson. Boinc: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, pages 4–10, Washington, DC, USA, 2004. IEEE Computer Society.

[13] M. Boulton.

[14] K. Bugeja. High-fidelity graphics using unconventional distributed rendering approaches. 2015.

[15] K. Bugeja, K. Debattista, S. Spina, and A. Chalmers. Collaborative High-fidelity Rendering over Peer-to-peer Networks. 2014.

[16] A. Chalmers. *Practical Parallel Rendering*. CRC Press, Hoboken, NJ, 2002.

[17] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 137–145, New York, NY, USA, 1984. ACM.

[18] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, Jan. 1982.

[19] C. Crassin, D. Luebke, M. Mara, M. McGuire, B. Oster, P. Shirley, P.-P. Sloan, and C. Wyman. Cloudlight: A system for amortizing indirect lighting in real-time rendering. Technical Report NVR-2013-001, NVIDIA, July 2013.

[20] C. Crassin, F. Neyret, M. Sainz, S. Green, and E. Eisemann. Interactive indirect illumination using voxel-based cone tracing: An insight. In *ACM SIGGRAPH 2011 Talks*, SIGGRAPH '11, pages 20:1–20:1, New York, NY, USA, 2011. ACM.

[21] A. K. Dattaz, M. Gradinariu, M. Raynal, and G. Simon. Anonymous publish/subscribe in p2p networks. pages 22–26, 2003.

[22] K. Debattista, L. P. Santos, and A. Chalmers. Accelerating the Irradiance Cache through Parallel Component-Based Rendering. In A. Heirich, B. Raffin, and L. P. dos Santos, editors, *Eurographics Symposium on Parallel Graphics and Visualization*. The Eurographics Association, 2006.

[23] D. E. DeMarle, S. Parker, M. Hartner, C. Gribble, and C. Hansen. Distributed interactive ray tracing for large volume visualization. In *Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, PVG '03, pages 12–, Washington, DC, USA, 2003. IEEE Computer Society.

[24] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '87, pages 1–12, New York, NY, USA, 1987. ACM.

[25] P. Dubla, K. Debattista, L. P. Santos, and A. Chalmers. Wait-Free Shared-Memory Irradiance Cache. In K. Debattista, D. Weiskopf, and J. Comba, editors, *Eurographics Symposium on Parallel Graphics and Visualization*. The Eurographics Association, 2009.

[26] P. Dutre, K. Bala, P. Bekaert, and P. Shirley. *Advanced Global Illumination*. AK Peters Ltd, 2006.

[27] P. Eugster, R. Guerraoui, A. M. Kermarrec, and L. Massoulie. From Epidemics to Distributed Computing. *IEEE Computer*, 37(5):60–67, May 2004.

[28] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces. *SIGGRAPH Comput. Graph.*, 18(3):213–222, Jan. 1984.

[29] D. P. Greenberg, K. E. Torrance, P. Shirley, J. Arvo, E. Lafortune, J. A. Ferwerda, B. Walter, B. Trumbore, S. Pattanaik, and S.-C. Foo. A framework for realistic image synthesis. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 477–494, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[30] P. S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 145–154, New York, NY, USA, 1990. ACM.

[31] D. S. Immel, M. F. Cohen, and D. P. Greenberg. A radiosity method for non-diffuse environments. *SIGGRAPH Comput. Graph.*, 20(4):133–142, Aug. 1986.

[32] H. W. Jensen. Global illumination using photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*, pages 21–30, London, UK, UK, 1996. Springer-Verlag.

[33] H. W. Jensen. *Realistic Image Synthesis Using Photon Mapping*. A K Peters, Ltd., Wellesley, Massachusetts, 2001.

[34] J. T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA, 1986. ACM.

[35] A. Keller. Instant radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[36] R. Koholka, H. Mayer, and A. Goller. *MPI-parallelized Radiance on SGI CoW and SMP*, pages 549–558. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[37] J. Krivanek and P. Gautron. *Practical Global Illumination with Irradiance Caching*. Morgan & Claypool, 2009.

[38] E. P. F. Lafortune, S.-C. Foo, K. E. Torrance, and D. P. Greenberg. Non-linear approximation of reflectance functions. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH

'97, pages 117–126, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[39] R. R. Lewis. Making shaders more physically plausible. Technical report, Vancouver, BC, Canada, Canada, 1993.

[40] M. Manzano, J. A. Hernández, M. Uruenña, and E. Calle. An empirical study of cloud gaming. In *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games*, NetGames '12, pages 17:1–17:2, Piscataway, NJ, USA, 2012. IEEE Press.

[41] M. McGuire and D. Luebke. Hardware-accelerated global illumination by image space photon mapping. In *Proceedings of the Conference on High Performance Graphics 2009*, HPG '09, pages 77–89, New York, NY, USA, 2009. ACM.

[42] J. Mitchell, G. McTaggart, and C. Green. Shading in valve's source engine. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, pages 129–142, New York, NY, USA, 2006. ACM.

[43] M. J. Muuss and M. J. Muuss. To wards real-time ray-tracing of combinatorial solid geometric models, 1995.

[44] F. E. Nicodemus. Directional reflectance and emissivity of an opaque surface. *Appl. Opt.*, 4(7):767–775, Jul 1965.

[45] H. Papadakis, P. Fragopoulou, E. P. Markatos, M. Dikaiakos, and A. Labrinidis. *Divide et Impera: Partitioning Unstructured Peer-to-Peer Systems to Improve Resource Location*, pages 1–12. Springer US, Boston, MA, 2008.

[46] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan. Interactive ray tracing for isosurface rendering. In *Proceedings of the Conference on Visualization '98*, VIS '98, pages 233–238, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[47] M. Z. Patoli, M. Gkion, A. Al-Barakati, W. Zhang, P. Newbury, and M. White. An open source grid based render farm for blender 3d. In *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, pages 1–6, March 2009.

[48] J. A. M. Ramos, C. Gonzlez-Morcillo, D. V. Fernndez, and L. M. Lpez-Lpez. Yafrid-NG: A Peer to peer Architecture for Physically Based Rendering. In C. Andujar and J. Lluch, editors, *CEIG 09 - Congreso Espanol de Informatica Grafica*. The Eurographics Association, 2009.

[49] J. H. Reif, J. D. Tygar, and A. Yoshida. Computability and complexity of ray tracing. *Discrete & Computational Geometry*, 11(3):265–288, 1994.

[50] E. Reinhard, A. Chalmers, and F. W. Jansen. Overview of parallel photo-realistic graphics. Technical report, Bristol, UK, UK, 1998.

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

[52] C. Schlick. A customizable reflectance model for everyday rendering. In *In Fourth Eurographics Workshop on Rendering*, pages 73–83, 1993.

[53] P. Shirley and S. Marschner. *Fundamentals of Computer Graphics.* A. K. Peters, Ltd., Natick, MA, USA, 3rd edition, 2009.

[54] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pages 149–160, New York, NY, USA, 2001. ACM.

[55] S. Tarkoma. *Overlay Networks: Toward Information Networking.* Auerbach Publications, Boston, MA, USA, 1st edition, 2010.

[56] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: The condor experience: Research articles. *Concurr. Comput. : Pract. Exper.*, 17(2-4):323–356, Feb. 2005.

[57] J. F. Torres-Rojas and M. Ahamad. Plausible clocks: constant size logical clocks for distributed systems. *Distributed Computing*, 12(4):179–195, 1999.

[58] S. Voulgaris, D. Gavidia, and M. van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 13(2):197–217, 2005.

[59] S. Voulgaris, E. Rivire, A.-M. Kermarrec, and M. V. Steen. Sub-2-sub: Self-organizing content-based publish subscribe for dynamic large scale collaborative networks. In *In IPTPS06: the fifth International Workshop on Peer-to-Peer Systems*, 2006.

[60] I. Wald, A. Dietrich, and P. Slusallek. An interactive out-of-core rendering framework for visualizing massively complex models. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[61] I. Wald, T. Kollig, C. Benthin, A. Keller, and P. Slusallek. Interactive global illumination using fast ray tracing. In *Proceedings of the 13th Eurographics Workshop on Rendering*, EGRW '02, pages 15–24, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[62] I. Wald, P. Slusallek, C. Benthin, and M. Wagner. Interactive distributed ray tracing of highly complex models. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 277–288, London, UK, UK, 2001. Springer-Verlag.

[63] G. J. Ward, F. M. Rubinstein, and R. D. Clear. A ray tracing solution for diffuse interreflection. *SIGGRAPH Comput. Graph.*, 22(4):85–92, June 1988.

[64] T. Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, June 1980.

[65] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE J.Sel. A. Commun.*, 22(1):41–53, Sept. 2006.