PEER-TO-PEER INTERACTIVE 3D MEDIA DISSEMINATION IN NETWORKED VIRTUAL ENVIRONMENTS

LIANG KE

A THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

AT

SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE

2012

Acknowledgements

This dissertation would not have been possible without the guidance and the help of my advisor Prof. Roger Zimmermann, whose sincerity and encouragement I will never forget. He contributed and extended his valuable assistance in the preparation and completion of this study. He lead me to the door into the world of research, handed me the torch that illuminated a few steps ahead in the unknown world, tolerated my mistakes, and fortified my mind when I felt helpless.

I would also like to express my gratitude to Prof. Ooi Wei Tsang. He shared with me his wisdom of teaching and doing research, and encouraged me on every step forward during the candidature.

The School of Computing, National University of Singapore offered me a scholarship and a good place to study. This opportunity changed my life so much that I will always be thankful during the rest of my life.

This research has been funded in part by A*Star PSF SERC grant 082 101 0028. I also acknowledge the support of the NUS Interactive and Digital Media Institute (IDMI).

Last but not the least, I thank my parents and my family for supporting me throughout all my University studies.

Publications

- **Ke Liang**, Beomjoo Seo, Andrew Kryczka, and Roger Zimmermann. "*IDM: An Indirect Dissemination Mechanism for Spatial Voice Interaction in Networked Virtual Environments*". The IEEE Transactions on Parallel and Distributed Systems (IEEE TPDS, regular paper), 2012.
- **Ke Liang**, Roger Zimmermann, and Wei-Tsang Ooi. *Peer-Assisted Texture Streaming in Metaverses.* The 19^{th} ACM International Multimedia Conference (ACM MM'11, long paper), Scottsdale, Arizona, USA, 2011.
- **Ke Liang**, Roger Zimmermann. "*Maximizing System Reachability for P2P-based Interactive Spatial Audio Applications in Networked Virtual Environments*". The 2011 IEEE International Conference on Multimedia and Expo (ICME'11), Barcelona, Spain, July 11-15, 2011.
- **Ke Liang**, Roger Zimmermann. "*Cross-Tree Adjustment for Spatialized Audio Streaming over Networked Virtual Environments*". The 19^{th} International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'09), Williamsburg, Virginia, USA. June 3-5, 2009.
- Roger Zimmermann, **Ke Liang**. "*Spatialized Audio Streaming for Networked Virtual Environments*". The 16^{th} International Multimedia Conference (ACM MM'08, long paper), Pan Pacific Hotel, Vancouver, BC, Canada, 2008.

Abstract

Recent years have witnessed a significant growth in networked virtual environments (NVEs) which are increasingly popular and represent a range of applications. Some online virtual worlds have a dedicated purpose, such as Massively Multiplayer Online Games (MMOG), while others implement more of foundational frameworks which are not necessarily applications *per se*, but form platforms to create applications. The latter type of NVEs is referred to as metaverses. One of the premier examples of the latter is Second Life from Linden Lab.

A significant shortcoming in current NVEs concerns the voice communication between virtual world participants. To enable the creation of an aural soundspace around the user that matches the visual experience, audio streams sent by different speakers cannot be mixed until they reach their destinations. I propose interactive spatial audio dissemination protocols for NVEs in a peer-to-peer (P2P) manner. This type of P2P media dissemination has not yet been significantly investigated, and numerous challenging problems have to be overcome – among them providing low latency, resilience to churn, effective load balancing and rapid convergence – in such highly dynamic environments.

Additionally, user extensible NVEs/metaverses need an effective way to disseminate massive and dynamic 3D contents (e.g., textures, animations, meshes, etc.) to online users, and at the same time maintain low consumption of server bandwidth. P2P (or peer-assisted) technologies have been widely considered as a desirable complementary solution to efficaciously offload servers in large-scale streaming applications. However, due to both the bandwidth constraints of heterogeneous peers and unpredictable access patterns of latency-sensitive 3D

textures, it is challenging to reduce the server bandwidth consumption in metaverses without degrading the user experiences.

In this thesis I investigate the design of P2P-based interactive 3D media dissemination protocols that can satisfy the demanding real-time requirements. Simulation results are presented that show the feasibility and utility of the design, which can achieve near-optimal system reachability (for 3D audio), near-optimal server bandwidth consumption (for 3D textures), and satisfy the tight latency constraints of interactive 3D media under conditions of churn, avatar mobility and heterogeneous user access network bandwidth.

Contents

List of Figures ix

List of Tables xi

1 Introduction 1

1.1 Networked Virtual Environment (NVE) . . . 1
1.2 3D Audio Streaming in NVEs . . . 2
1.3 Texture Streaming in NVEs . . . 6
1.4 Challenges and Investigation Goals . . . 8
1.4.1 3D Audio Streaming . . . 8
1.4.2 3D Texture Streaming . . . 10
1.5 Contributions . . . 11
1.6 Organization of the Thesis . . . 12

2 Existing Work 15

2.1 P2P Overlay Networks . . . 15
2.1.1 Unstructured P2P Overlay Networks . . . 16
2.1.2 Structured P2P Overlay Networks . . . 16
2.2 Content Delivery Topologies over P2P Overlay Networks . . . 18
2.2.1 Single Multicast Tree . . . 19
2.2.2 Multiple Multicast Trees . . . 20
2.2.3 Mesh . . . 22
2.3 3D Audio Streaming . . . 23
2.4 Texture Streaming . . . 27

3 Research Overview **29**
3.1 Intra-AoI Multicast Tree Approach for 3D Audio Streaming . . . 29
3.2 Game-theoretic Approaches for P2P-based 3D Audio Streaming . 31
3.3 Game-theoretic Approach for Peer-assisted Texture Streaming . . 32

4 Intra-AoI Approach for P2P 3D Audio Streaming **33**
4.1 System Model . . . 35
4.2 Problem Formulation . . . 38
4.3 The Intra-AoI Tree Approach . . . 39
4.3.1 MLT Construction . . . 41
4.3.2 Audio Stream Mixing . . . 45
4.4 Cross-Tree Adjustment (CTA) for the Intra-AoI Approach . . . 47
4.4.1 Problem Formulation . . . 47
4.4.2 Overview of CTA . . . 48
4.4.3 Voting Process . . . 49
4.4.4 Allocation Process . . . 54
4.5 Conclusion . . . 54

5 Game-Theoretic Approaches for P2P-based 3D Audio Streaming **57**
5.1 Preliminaries of Congestion Games . . . 59
5.2 Unweighted Congestion Game Formulation . . . 61
5.2.1 System Reachability in the UCG Formulation . . . 62
5.2.2 Schur Concavity of System Reachability . . . 62
5.2.3 Abstraction Layer Construction . . . 64
5.2.4 Load Balancing in the UCG Formulation . . . 65
5.2.5 Convergence Time Analysis of the UCG Formulation . . . 66
5.3 Weighted Congestion Game Formulation . . . 70
5.3.1 System Reachability in the WCG Formulation . . . 73
5.3.2 Proportional Load Balancing in the WCG Formulation . . 76
5.3.3 Low Latency Audio Dissemination . . . 78
5.3.4 Convergence Time Analysis of the WCG Formulation . . . 79
5.4 Conclusion . . . 85

6 Evaluation of Approaches for 3D Audio Streaming over NVEs 87
6.1 Evaluation Settings . . . 87
6.2 System Reachability . . . 89
6.3 End-to-end Latency . . . 94
6.4 System Convergence and Stability . . . 97
6.5 Overhead . . . 99
6.6 Angular Errors Introduced by Audio Mixing . . . 100

7 Game-Theoretic Approach for Peer-assisted Texture Streaming 103
7.1 System Model . . . 103
7.2 Problem Formulation . . . 105
7.3 Minimizing Server Bandwidth Cost . . . 107
7.4 Peer Selection Strategy . . . 109
7.5 Convergence Analysis . . . 112
7.6 Evaluation . . . 115
7.6.1 Data Collection and Simulation Setup . . . 116
7.6.2 Server Bandwidth Consumption . . . 117
7.6.3 Server Request Ratio . . . 119
7.6.4 Overhead . . . 120

8 Conclusion and future work 125

9 Appendix 129

References 131

List of Figures

1.1 Exemplar applications of NVEs: (a) Second Life, and (b) World of Warcraft. . . . 2
1.2 Conceptual spatial sound field reproduction example with avatar positions in the virtual space and user locations in the physical space. . . . 3
1.3 Composition of a 3D object in NVEs. . . . 6

4.1 Intra-AoI 3D audio dissemination topology with three speakers, v_0, v_5 and v_{12}. . . . 37
4.2 Audio streams mixing. . . . 40
4.3 The two processes of the Intra-AoI approach. . . . 43
4.4 Temporal multicast tree construction and node partition. . . . 49
4.5 Three cases of units scheduling at two conflict nodes, v_j and v_z. . . . 53

5.1 Audio dissemination in the UCG formulation. . . . 61
5.2 Water-fitting illustration: nodes only consider to switch from overloaded helpers in $[h_{a+1}, h_k]$ to underloaded helpers in $[h_1, h_a]$. . . . 66
5.3 The equivalent problem: minimizing the number of collisions in a Balls-into-Bins game. . . . 74

6.1 Four scenarios used in the evaluation. . . . 89
6.2 Distribution of number of neighbors. . . . 90
6.3 Distribution of download and upload bandwidth. . . . 90
6.4 System reachability with different HPSRs in Scenario **U-30**, where nodes are uniformly distributed in the NVE. The radius of each node is 30 m. . . . 92

6.5 System reachability with different HPSRs in Scenario **U-40**, where nodes are uniformly distributed in the NVE. The radius of each node is 40 m. 93
6.6 System reachability with different HPSRs in Scenario **C-30**, where nodes are clustered in the NVE. The radius of each node is 30 m. . . . 94
6.7 System reachability with different HPSRs in Scenario **C-40**, where nodes are clustered in the NVE. The radius of each node is 40 m. . . . 95
6.8 End-to-end latency performance in the scenario where the positions of avatars are uniformly distributed in the NVE. 96
6.9 End-to-end latency performance in the scenario where the positions of avatars are clustered in the NVE. 96
6.10 Worst-case convergence rate of UCG and WCG. 98
6.11 Number of online users in Secondlife during a day. 98
6.12 Value of potential function over time. 99
6.13 Fraction of nodes updating their helpers over time. 100
6.14 Distribution of the time interval between two successive helper updating operations 101
6.15 CDF of angular error when mixing audio streams with a fixed talking probability of 50% in the two scenarios. 102

7.1 2D positions of textures (left) and the number of concurrent users (right) in different regions, each of which is 256×256 meters. . . 121
7.2 Distribution of textures sizes. 122
7.3 Distribution of peer bandwidth. 122
7.4 Distribution of server request ratio (smaller is better) in different regions during a day, when the latency constraint of textures is set to 5 seconds and 10 seconds. 123

List of Tables

2.1 Categorization of interactive audio streaming techniques. 24

4.1 List of terms used in the Intra-AoI approach. 38

5.1 List of terms used in WCG. 72

6.1 List of parameters used in the simulation. 88

7.1 List of notations and their definitions. 105

7.2 Total number and size of textures and user hours of different regions during a day. 116

7.3 Server bandwidth consumption (smaller is better) and the fraction of saved bandwidth (larger is better) of different algorithms during a day, when the latency constraint of textures is set to 5 seconds. 117

7.4 Server bandwidth consumption (smaller is better) and the fraction of saved bandwidth (larger is better) of different algorithms during a day, when the latency constraint of textures is set to 10 seconds. 118

7.5 Expected server request ratio (smaller is better) in different regions, when the latency constraint of textures is 5 seconds. 119

7.6 Expected server request ratio (smaller is better) in different regions, when the latency constraint of textures is 10 seconds. . . . 119

7.7 Averaged communication overhead per peer under different latency constraints (i.e., d) of textures in different regions. 120

LIST OF TABLES

1

Introduction

1.1 Networked Virtual Environment (NVE)

The Internet has become an indispensable tool for people to interact on a global scale. One type of large-scale interactive applications that is emerging on the Internet are networked virtual environments (NVE) where a user can move her/his representation (also known as an avatar) in a shared virtual world and interact with each other. One of the most well-known examples of an NVE is Second Life (www.secondlife.com), as shown in Figure 1.1(a). Such virtual worlds are interesting for a number of reasons. Most importantly, some of these virtual environments are not applications *per se*, but they form the foundation for the creation of specific applications. For example, Second Life has been used for virtual meetings, training and recruitment by large corporations such as IBM. The term metaverse has also been used to describe such generic NVEs. Of course, one of the biggest application areas for NVEs are games, also termed Massively Multiplayer Online Games (MMOG) (see Figure 1.1(b)).

NVEs are an active field of research and present a plethora of technical challenges. Among them are scalability of network traffic and server environments, end-to-end delay of user interactions, human-computer interface issues, visual representations and more. For some of these problems considerable progress has been achieved. For example the visual quality of the three-dimensional environments in these shared worlds is pleasing and constantly improving through better hardware and software algorithms. However, the two areas that have been sorely

(a) Second Life. (b) World of Warcraft.

Figure 1.1: Exemplar applications of NVEs: (a) Second Life, and (b) World of Warcraft.

lagging behind are natural three-dimensional (3D) audio streaming and 3D texture streaming. The former aims to provide a 3D aural experience that matches the corresponding visuals NVEs, and the goal of the latter is to minimize the server bandwidth consumption for 3D texture dissemination without degrading the user experience. In this thesis, I propose effective solutions for both issues.

1.2 3D Audio Streaming in NVEs

The most widely used mode of communication in NVEs is through text chat. While this is a mature technology that is relatively easy to implement, it lacks much of the natural and immersive characteristic that good-quality voice communication could provide for users.

A number of monophonic[1] voice communication approaches have been proposed in the literature and some commercial systems exist (e.g., offerings from Ventrilo and TeamSpeak). However, existing solutions, with few exceptions, fall short in that they are based on centralized client-server architectures, which may raise the problem of single point of failure and limited scalability as the number

[1]Monophonic or mono denotes that all the audio signals are mixed together into a single stream which is routed through a single audio channel. The mixed audio stream contains no level and arrival time/phase information that would replicate or simulate directional cues.

of online users scales up. In addition, they do not provide a 3D aural experience for users that matches their visuals in NVEs, since the mono audio contains no spatial information.

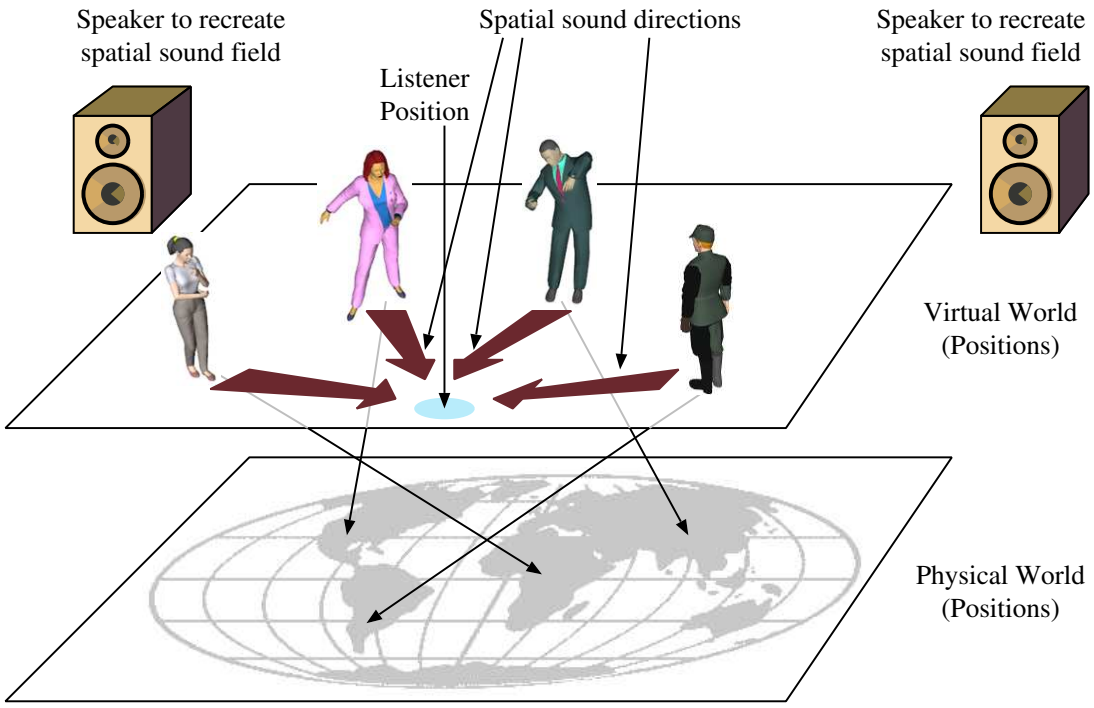


Figure 1.2: Conceptual spatial sound field reproduction example with avatar positions in the virtual space and user locations in the physical space.

The most important disadvantage of existing monophonic audio streaming systems is that they are not designed to utilize the spatial information (e.g., location, distance and directionality) between avatars. However, with a spatialized voice service, users can more easily identify who is speaking if several people surround them (see Figure 1.2). By matching users' aural environment to their visual perception more immediacy and dynamism is lent to their interaction with others, providing an overall more immersive experience.

To achieve natural, immersive interactive communication for virtual world participants two important concepts need to be realized:

- First, communication should naturally be enabled within a certain proximity of an avatar. More specifically, users are more likely to interact within a

close distance range measured in virtual coordinates. This concept, which is commonly referred to as *Area of Interest* (AoI), has been well studied [1,2] and used in Second Life. Once the AoI is defined, the users within the same AoI are expected to be clustered to reduce delay and control message overhead.

- The second component for natural interaction is the creation of an *aural soundscape* that matches the visual landscape of an environment. Spatialized, three-dimensional audio rendering is now possible on commodity personal computer hardware. While surround-sound speakers (e.g., 5.1 channels) can provide full 360 degree rendering, even stereo speakers are capable of creating roughly a 180 degree sound field in front of the user. Libraries such as OpenAL implement the signal processing algorithms necessary to position sound sources in specific locations.

For interactive 3D audio stream dissemination, existing solutions can be categorized into two approaches: client–server and peer-to-peer (P2P) mixing. The former utilizes a client-server (C/S) streaming model, where every node sends its talk-spurts to a central server which then creates a personalized stereo mix for each user and sends it back to its corresponding client (e.g., exemplar systems are Vivox, Dolby Axon). This scheme is insensitive to churn, avatar mobility, and access network bandwidth and provides users with relatively small one–hop latencies. However, it requires immense server resources (both processing power and network bandwidth). The latter mixes multiple audio channels into a mono channel (or stereo channels) along dissemination paths, whose dissemination topology is typically based on a P2P model (e.g., AudioPeer [3], PeerTalk [4]). While mixing reduces audio traffic significantly, it loses the spatial positioning information of individual streams, i.e., the mixing process is irreversible (note that some audio coding techniques such as Binaural Cue Coding (BCC) allow de-mixing with a small amount of side information [5], but they are too computationally expensive to be adopted in real systems.).

A third, alternative approach to build a dissemination topology is to leverage a distributed P2P topology. The benefits of a P2P architecture lie in its scalability, i.e., the number of users that can be supported. In the C/S scheme,

a centralized server with a FastE connection (100 Mb/s) can support approximately 4,000 concurrent users at a rate of 25 kb/s per stream (each user sends one mono stream to the server and in turn receives one spatial mixed stereo stream). Note that voice transmission bandwidth – while low compared to video – is quite substantial compared to game control traffic. The same FastE connection could support an order of magnitude more users if only game control traffic is exchanged. To give an indication of the potential of a P2P solution, *PPLive* software architect Gale Huang reported the support of 250,000 users on their P2P video streaming network (500 kb/s video stream rate) supplied from a single PC with a 10 Mb/s connection [6]. While the details of our spatial audio system are quite different from PPLive, its architecture can be expected to provide similar scalability. In fact, since voice streams are generated in a distributed manner in the network, strictly speaking no traditional server is necessary (except for user account administration). Since a business FastE (or GigE) connection likely incurs a significant expense, a P2P architecture represents an attractively cost-effective solution that would allow even small and middle-tier game publishers to offer spatial voice features.

This motivates me to build 3D audio dissemination topologies in a P2P manner, such that network bandwidth of participating users can be utilized to deliver audio streams. A number of one-to-one, one-to-many and few-to-few audio communication applications have used P2P topologies to disseminate streams (e.g., [4, 7–10], Skype). However, these solutions are in general monophonic, hence providing no audio spatialization. Furthermore, they often ignore access bandwidth constraints to make the topology construction computationally tractable [10]. One of goals of this thesis is to investigate P2P-based dissemination topologies for interactive 3D audio services in NVEs that can deliver 3D audio streams from many speakers to many receivers (positioned within every speaker's AoI) while achieving low delay, accommodating system dynamics, and respecting peers' network bandwidth constraints. Since the bandwidth capacities of users in a P2P network are limited and heterogeneous, a challenging problem in this design is how to fully utilize users' limited and heterogeneous network resources to deliver interactive 3D audio streams within latency constraints of interactive audio.

1.3 Texture Streaming in NVEs

3D contents in metaverses (e.g., Second Life) are objects such as buildings, terrain, and trees. Primitives (see Figure 1.3(a)) are the basic in-world building blocks, and they provide the framework around which objects can be created. Textures are essentially bitmap images that are used to provide surface coloring for objects, as shown in Figure 1.3(b). Considering the fact that a significant portion (over 50%) of the metaverse server bandwidth is consumed by transmission of the textures of 3D contents in metaverses [11], we focus on texture streaming in this thesis and refer to 3D contents as textures throughout the thesis.

(a) Primitive. (a) Texture. (c) 3D object.

Figure 1.3: Composition of a 3D object in NVEs.

Unlike massively multiplayer online games (MMOGs), where users install the whole static 3D contents in advance via DVDs or patches downloaded offline from the Internet, the metaverses need to stream interactive 3D contents to users live over the Internet, according to their visibility or interests. Because users in metaverses can contribute to the metaverses by creating and uploading their own objects which everybody can see and interact with. Consequently, when avatars move around in a metaverse, their immediate environment will be downloaded dynamically from the metaverse server to the corresponding users.

Most commercial metaverses (e.g., Second Life [12] and ActiveWorlds [13]) are currently deployed using a client/server (C/S) scheme, where centralized servers are used to maintain the states of metaverses and online users, and to distribute contents (throughout the thesis, we use the term content, texture, and 3D content

interchangeably) to users. Since users do not store the entire virtual worlds, but download the contents from dedicated servers when needed, the server bandwidth costs can be huge when the amount of contents and the number of online users increases. For example, according to the economic report from Second Life [14] in the third quarter of 2010, the user hours (i.e., the sum of the combined length of all user sessions) per month is roughly 35 million hours. From Tables 7.2 and 7.3 in Section 7.6.1, we can observe that the average amount of texture data sent from the servers per user hour is 263 MB. Therefore, roughly 9 petabytes (PB) of texture data will be delivered from the servers to users monthly. As a consequence, one of the most important and formidable challenges facing the metaverse providers is to reduce the server bandwidth consumption.

A natural way to offload servers is to utilize bandwidth and caches of online users using P2P technologies, i.e., enable online users to serve one another. With the assistance of *peers* (we use the term peer, avatar and user interchangeably thereafter), the server bandwidth cost can be reduced, and the scalability and affordability of the system can be improved. More specifically, a peer can download the required contents from other peers via sending *requests* to them, which may send back the corresponding contents if they have available bandwidth. In order not to degrade the user satisfaction of metaverses, all the required contents need to be downloaded within a certain latency constraint measured in seconds. If a peer fails to download the required textures from other peers within the latency constraint, it will send the requests directly to dedicated servers, thus consume bandwidth on the server side. Since the server is not replaced but complemented by the peers, we refer to such systems as *peer-assisted systems*. To minimize the server bandwidth cost, peers need to select appropriate peers as their content *providers*, respecting the available bandwidth of peers and latency constraints of the requested contents.

Peer selection strategies, which are based on P2P technologies and aim to reduce the server bandwidth consumption through selecting appropriate content providers for each peer, have been extensively studied in peer-assisted media streaming systems. Note that the media contents in both live and on-demand streaming systems are viewed as one-dimensional (i.e., time) data files. Each peer is assumed to request the media contents sequentially. In texture streaming

systems, however, the contents are viewed as data files that are stored in a multi-dimensional space (i.e., locations). In addition, peers have different download and upload capacities in practice (see Figure 7.3), and they download the contents on demand according to their behaviors *per se*, such as walking, running and flying in metaverses. These unpredictable and non-linear request patterns in peer-assisted texture streaming systems with heterogeneous peers make the design of peer selection strategies much more challenging compared with media streaming systems.

Additionally, a practical peer selection strategy should be performed by all the peers in a decentralized manner. I believe the peer selection strategy in such peer-assisted texture streaming systems can be characterized in a game-theoretic setting, and thus inherit its decentralized and practical nature. Specifically, requests sent by peers can be viewed as players in a game, where each is associated with a weight which is the size of the requested texture. Each player in the game is selfish and will try to minimize her own cost. In other words, peers will send their requests to underloaded providers.

This motivates me to design a light-weight, decentralized game-theoretic peer selection strategy for peer-assisted texture streaming systems with heterogeneous peers. The proposed peer selection strategy needs to be able to minimize the server bandwidth consumption while respecting the latency constraints of textures. In addition, it should have a rapid convergence rate due to churn, unpredictable request patterns of textures, and time-varying bandwidth capacities of online users. It is important to point out that I seek to reduce the server bandwidth consumption via fully and optimally utilizing the available network capacities of peers, hence the focus of this thesis is not on incentive mechanisms [15] which, however, aim to increase the available network resources.

1.4 Challenges and Investigation Goals

1.4.1 3D Audio Streaming

Providing P2P-based interactive 3D audio services for NVEs is very challenging, mainly due to the following three reasons. Firstly, a large amount of network

bandwidth is required to deliver the audio streams, since a speaker may need to send the audio stream to multiple receivers. Note that the technique of audio mixing can reduce the bandwidth usage by mixing multiple audio streams into a single mono stream, and then distribute the mixed stream to all recipients. However, this will result in a loss of spatial information (e.g., location, distance, and directionality)[1]. Therefore, audio streams should be delivered in a many-to-many manner, i.e., multiple streams need to be sent and received at each user. To this end, the P2P-based 3D audio dissemination topologies should be well designed to achieve high *system reachability*, defined as the fraction of the receivers that successfully receive their streams, via efficiently utilizing the bandwidth of participating peers.

Secondly, all the interactive audio streams have tight latency constraints and must be delivered within a certain time limit. Specifically, the specified maximum acceptable one-way latency in an interactive audio application is about 200 ms [16]. If the one-way latency is higher than that, users will find it disturbing and the quality-of-service starts to suffer. Therefore, the audio delivery path from senders (i.e., speakers) to receivers should be as short as possible. Due to the bandwidth constraints of participating peers, the desirable dissemination topologies should minimize the overall average latency of audio streams, in the meantime, maintaining high system reachability.

Thirdly, the NVEs are highly dynamic and heterogeneous environments due to both the equipments of the participating users which have heterogeneous bandwidth and the system dynamics, which can be categorized into system churn (users leave and join the system freely), avatar mobility (users move freely) and the unpredictable speaking patterns (users speak freely). Therefore, the audio dissemination topologies need to be robust such that the quality-of-service (i.e., delay, jitter and interrupt rate) will not be vulnerable to the system dynamics.

[1]Spatial audio mixing or binaural cue coding technique can preserve all spatial cues of input streams, but are not widely used in multi-hop overlay networks due to its higher processing requirements.

1.4.2 3D Texture Streaming

Peer selection strategies, which aim to reduce the server bandwidth consumption, have been extensively studied in peer-assisted media streaming systems, including live [17, 18] and video-on-demand (VoD) [19, 20] P2P streaming applications. However, current peer selection strategies for those live or VoD P2P streaming systems cannot be easily applied to 3D content streaming in metaverses, due to the unpredictable and non-linear content request patterns of participating users.

More specifically, in live P2P streaming systems, every peer has the same content request pattern and it does not usually request the contents that precede the playback timestamp (assuming the downloaded contents have been stored in the caches of users if back-seeking is allowed). In VoD P2P streaming systems, every peer can (mostly) predict the contents it needs in the near future. Therefore, the media contents in both live and VoD streaming systems can be regarded as to one-dimensional (i.e., time) data files. In 3D texture streaming systems, however, the contents are viewed as data files that are stored in a multi-dimensional space (i.e., locations). As a result, the to-be-required contents of metaverse users cannot be predicted precisely due to the avatar mobility. In other words, metaverse users with different download and upload capacities need to download the contents on demand according to their locations, visibility and interests.

Furthermore, the number of potential content providers for each peer in live or VoD P2P streaming systems is large in general, and thus the sets of content providers of peers are less overlapped (results in less conflicts). However, in 3D texture streaming systems, the potential content providers of a peer (e.g., an avatar) are mostly its neighbors in the virtual world, hence the number of potential content providers is small and the sets of content providers of peers are highly overlapped.

Consequently, the design of peer-assisted 3D texture streaming is much more challenging compared with live or VoD P2P streaming systems. Besides, prior research on peer selection strategies in peer-assisted texture streaming systems has paid little attention to the issue of optimally utilizing the bandwidth of heterogeneous peers while respecting the latency constraints of textures, hence it is impractical.

1.5 Contributions

For 3D audio streaming in NVEs, I proposed two different types of solutions. One is a Intra-AoI tree approach [21, 22], which is based on traditional multicast tree technology. The Intra-AoI approach is straightforward but has sub-optimal performance in terms of system reachability and end-to-end latency. Moreover, it is vulnerable to system dynamics.

To overcome these limitations of these tree-based solutions, I introduce the novel designs of P2P-based 3D audio dissemination topologies that applies the concepts of game theory – unweighted congestion game (UCG) and weighted congestion game (WCG). Two *game-theoretic* solutions (termed UCG and WCG, respectively) have been proposed in this thesis, which have resulted in the following important contributions related to 3D audio dissemination in NVEs:

1. **Maximizing the system reachability.** Consider that the aggregate available bandwidth of the enclosed peers varies across different AoIs, thus the creation of intra-AoI dissemination trees [21] [22] results in sub-optimal performance. By introducing the concept of location-unrestricted *helper* nodes in the proposed game-theoretic solutions to relay audio streams I proved that the expected system reachability is maximized when the loads of helpers are balanced proportionally to their bandwidth capacities. I proposed two distributed load balancing algorithms that implement this concept.

2. **Satisfying tight latency constraints.** Low end-to-end delay is a requirement in interactive audio streaming applications. To this end, the intra-AoI distribution tree algorithm is based on shortest path tree (SPT) algorithm, which can achieve a relative low latency. Due to the bandwidth constraints of peers, the intra-AoI distribution tree algorithm will be outperformed by C/S solutions in terms of latency performance. The proposed game-theoretic solutions, however, can surpass the latency performance of C/S solutions since every audio delivery path has zero or one intermediate node.

3. **Adapting to system dynamics.** System dynamics are a significant challenge in P2P systems. The performance of intra-AoI distribution tree algorithm will be severely affected by the avatar mobility, since the intra-AoI distribution tree is built according to the end-to-end latency between neighbors of the speaker. However, the game-theoretic solutions can moderate the impact of peer joins and leaves by utilizing two or more helper nodes. Furthermore, because helpers are not restricted to be located within AoIs of speakers, avatar mobility is mitigated and connections can last longer (resulting in less maintenance overhead). Significantly I prove that the convergence of our load balancing algorithm is achieved rapidly within $O(\log \log n)$ time intervals given a population of n peers, thus resulting in a very scalable solution.

In this thesis, I introduce a peer-assisted 3D texture streaming system that minimizes the server bandwidth cost without degrading the end-user satisfaction. I formulate the problem of server bandwidth minimization problem as a congestion game, and use the concept of congestion games to design a simple but effective peer selection strategy. The proposed algorithm is light-weight, and can efficiently utilize the bandwidth of heterogeneous peers in a decentralized manner by enabling each peer to repeatedly update its content providers independently and concurrently. I evaluate the algorithm through an extensive comparison study based on simulations using realistic texture information and avatar mobility traces collected from Second Life. As shown by our simulation results, the proposed algorithm can effectively reduce the server bandwidth cost and increase the scalability of metaverses.

1.6 Organization of the Thesis

The remainder of this thesis describes our approach in detail. We will start with a survey of the related work and techniques in Chapter 2. Research topics in the thesis are described in Chapter 3. Two different approaches for 3D audio streaming, as a starting point of my research in this area, are given in Chapters 4 and 5, respectively. Peer-assisted texture streaming topologies are discussed in

Chapter 7. Evaluation results of the proposed approaches for 3D audio and texture streaming are presented in Chapters 6 and 7, respectively. Finally, Chapter 8 draws conclusions and outlines future work in this space.

2

Existing Work

Recall that the focus of this thesis is how to deliver interactive 3D media in networked virtual environments (NVEs) using network resources of the participating users. The key idea is to maximize the system performance via effectively utilizing the limited network resources of the users in P2P overlay networks.

In this chapter, I first describe the existing work on P2P overlay networks, based on which the 3D media dissemination topologies in this thesis was proposed. Then some current media (video and mono audio) streaming topologies are discussed. Finally, existing work on 3D media streaming are discussed.

2.1 P2P Overlay Networks

Peer-to-peer (P2P) networks can be regarded as to abstract overlay networks, which are built upon the application layer, on top of the native or physical network topology. Such overlays are used for indexing and peer discovery and make the P2P networks independent from the physical network topology. In addition, P2P overlay networks can be regarded as distributed systems in which all peers can act as forwarders and end systems, and all communication is symmetric. The participating peers, termed nodes, can establish network connections with each other in P2P overlay networks. Based on the ways that the nodes in the overlay network are linked to each other, the P2P overlay networks can be categorized into two types: 1). unstructured, and 2). structured P2P overlay networks.

2.1.1 Unstructured P2P Overlay Networks

Unstructured P2P networks are formed in cases when the overlay links among peers are established arbitrarily. Such networks can be easily constructed such that a newly joined peer can contact the nodes that an existing peer has already connected, thus form its own links in the overlay network. In an unstructured P2P overlay network, if a peer wants to find a desired piece of data in the network, the query has to be flooded through the network to peers by relay. In particular, if a peer receives a query that it cannot satisfy, it will forward this query to every peer linked with it. This is time-inefficient because the queries for the contents that are not widely replicated may be sent to a large fraction of peers until the contents are found.

Another disadvantage of the unstructured P2P overlay networks is that the queries may not always be resolved. Popular contents are likely to be found at several peers, thus the queries for those contents can be easily resolved. But if a peer is looking for rare data shared by only a very few other peers, then it is highly unlikely that the query will not be resolved. Since there is no correlation between the location of a peer and the content it has, there is no guarantee that flooding will find a peer that has the desired data. Besides, flooding also incurs a high amount of signaling traffic in the network since a peer will forward the queries it cannot resolved to all the peers that it connected with. There are many popular unstructured P2P networks such as Gnutella [23].

Considering that efficient peer and object discovery is required in the proposed algorithms to deliver media contents with tight latency constraints, unstructured P2P overlay networks are not considered to be employed by the proposed algorithms, and will not be further discussed in this thesis.

2.1.2 Structured P2P Overlay Networks

The structured P2P overlay networks, on the contrary, employ a globally consistent protocol to ensure that any node can efficiently route a search to some peer that has the desired data, even if the data is extremely rare. More specifically, the P2P overlay network assigns keys to data items and organizes its peers via a graph which maps each data key to a peer. Such a guarantee necessitates a

more structured pattern of overlay links. By far the most common type of structured P2P network is formed using the distributed hash table (DHT) [24], where a variant of consistent hashing is used to assign the ownership of each file to a particular peer, in a similar way that a traditional hash table assigns each key to a particular array slot.

In a structured P2P overlay network, each peer and to-be-stored object is randomly assigned a long identifier which can uniquely identify the peer or the object with a high probability. This approach enables peers to find their required data in an efficient way, thus structured P2P overlay networks are desirable for the proposed algorithms in this thesis. A lot of research has been conducted in this area. Pastry [25] is used as an example in this chapter to show how the nodes in structured P2P overlay networks discover other nodes (i.e., peers), objects, and how the node join and leave events are handled.

- **Node identification and locating.** In Pastry, the identifiers of the nodes (node ids) and the objects (keys) can be thought of as a sequence of digits in base 2^b (b denotes the integer chosen by the administrator). In order to locate a node in Pastry, each node will maintain a routing table to support message forwarding. In particular, to send a message (with key M) from S to D, the sender first computes the longest prefix match between M and S, e.g., l digits, then she will choose a node in her routing table who has a longer prefix match (i.e., $l+1$ digits). Therefore, the number of possible destination nodes will be reduced by a factor of 2^b. That is, if there are N nodes in Pastry, every query can be resolved within $log_{2^b}N$ steps.

  Besides the routing table, each node also maintains a set of leaf nodes (leaf set) as well as a set of neighbor nodes (neighborhood set). The leaf set is a set of nodes whose identifiers are numerically closest to the local node. The neighborhood set, on the other hand, is a set of nodes who are closest to the local node according to the proximity metric. It is worth noting that the neighborhood set is not used in routing messages but to maintain the locality property of Pastry.

- **Node arrival and departure handling.** The nodes in Pastry detect the failures of nodes by sending keep-alive messages to others. Specifically,

each node periodically sends keep-alive messages to its neighbor nodes and leaf nodes, such that all the leaf nodes of the failed node will be notified immediately, and update their leaf sets accordingly. However, the nodes in the routing table entries of the failed node will be updated lazily. In particular, if a node contacts a node in its routing table but receives no responses, the failure of the node will be detected. To repair the failed routing table entry R_l^d at row l and column d, the node contacts the rest of nodes in the same row and retrieves the entries of those nodes for R_l^d. If no entries are found, the node will contact nodes in the next row, so on and so forth. It has been showed in [25] that this procedure is highly likely to find a node if one exists.

When a node, e.g., n_i, joins a Pastry network, it will contact a nearby node, e.g., n_j, in the Pastry network. Node n_j then routes a special message whose key is the id of the newly-join node n_i. This message will finally reach a node, e.g., n_k. Then node n_i will import the m^{th} row of the routing table of the m^{th} node in the routing path from n_i to n_k, and copies the leaf set of n_k to its own leaf set.

2.2 Content Delivery Topologies over P2P Overlay Networks

As mentioned above, P2P overlay networks have been of interest in the research community for a decade. Numerous file-sharing systems based on its concepts have been developed [26]. The applicability of P2P overlay networks to the problem of media streaming has been well investigated [9, 17–20]. In P2P multimedia streaming systems, the audio or video streams consist of small data blocks. Each block has a numerical sequence number by which the corresponding receivers can correctly reorder the received blocks for playback. In addition, there is a playback deadline for each block, which is supposed to be delivered to the destinations within the latency constraints. Due to the dynamic nature of users' uptime and limited bandwidth, it is very challenging to provide robust and scalable transmission topologies for such systems.

The P2P media dissemination topologies can be categorized into three types: single multicast tree, multiple multicast trees and mesh. All of these topologies are built upon P2P overlay networks as discussed in the previous section. Compared to IP multicast, which relies on the routers supporting IP multicasting, these P2P media topologies are deployed at the application layer where users can act as both forwarders and receivers in the multicast session. Therefore, P2P media dissemination topologies are application-level multicast (ALM) topologies which can be easily deployed over the Internet. Three typical ALM algorithms for media dissemination will be briefly summarized in this chapter.

2.2.1 Single Multicast Tree

A simple and straightforward content delivery topology is the single multicast tree topology. That is, a multicast tree is constructed for a multicast group. One example of the single multicast tree topology is Scribe [27].

When a multicast tree is created in Scribe, the node whose id is numerically closest to the group id will act as the rendezvous point for the multicast group, and this rendezvous point is the root of the multicast tree. Intermediate nodes in the multicast tree are called forwarders, each of which will maintain a set of children who will receive the content from the forwarders.

If a node wants to join the multicast group, it needs to route a 'JOIN' message with the group id to the rendezvous point of the multicast group. All the nodes, except for the rendezvous point and the newly joining node, along the path of the JOIN message will be added to the group as the forwarders of the multicast group. If the JOIN message first arrived at an existing forwarder, then the newly joining node will be accepted as a child of the forwarder. In a single multicast tree, the contents are disseminated from the rendezvous point to all the members of the group.

The members of a multicast group are maintained through probing heartbeat messages from each intermediate node to its children. If a child does not reply, she will be deleted from her father's children table. If a child did not receive heartbeat messages from her forward for a period of time, she will route a JOIN message to the rendezvous point to re-join the multicast group.

The main advantage of this simple multicast tree topology is that the multicast tree can be easily deployed over the Internet due to its simplicity. When a node wants to create a multicast session, it just needs to create a CREATE message with the group id as the key, and send this message to the Pastry network (Scribe is based on Pastry [25]). The destination of the CREATE message is the rendezvous node of the to-be-created multicast session. All the nodes that want to join the multicast group need to send the JOIN messages to the corresponding rendezvous node in the Pastry network.

There are three disadvantages of this single multicast tree topology. Firstly, the single tree structure is vulnerable to the system dynamics (i.e., churn) and the failures of the nodes in the topology. Secondly, the network resources are utilized non-uniformly in the P2P network such that the intermediate nodes in a multicast tree will have much higher bandwidth cost, and communication overheads than the leaf nodes. Lastly, as every node has only one content provider (i.e., the father node) in the multicast tree, the incoming throughput of the leaf nodes will be significantly affected by the outgoing bandwidth of their father node.

2.2.2 Multiple Multicast Trees

Note that the intermediate nodes in a single tree multicast, as shown in the above section, carry the load of forwarding data to their downstream nodes (i.e., leaf nodes). It may incur some problems if those intermediate nodes do not have enough out-going bandwidth to forward contents to their downstream nodes. Hence the throughputs of leaf nodes are constrained by the intermediate nodes in the multicast tree. Moreover, the out-going bandwidth of leaf nodes are not fully used to delivering contents, and thus the network resources are not efficiently utilized.

To circumvent these drawbacks of the single multicast tree topology, researchers have used multiple multicast trees to deliver multimedia contents. SplitStream [9], for example, uses multiple separated multicast trees to disseminate the media contents. It divides the to-be-sent contents into multiple streams (denoted by stripes), and then constructs multiple multicast trees. Each multicast tree is used to disseminate one single stream, and any node in the multicast session

will at most be an intermediate node of one multicast tree. In other words, an intermediate node in a tree cannot be an intermediate node in the other multicast trees.

Since a multicast tree is formed by the routes from all the nodes to the rendezvous node, the numerical identifiers (id) of all the intermediate nodes will have some common digits with the group id of the tree. Remember that, in the routing table of each Pastry node, entries in the i^{th} row have i common digits (in base 2^b) with the node. Therefore, all the intermediate nodes in a multicast tree will have at least one common digit. By choosing b, SplitStream can at most divide the content into 2^b streams. It should be noted that the media content is encoded using MDC (Multiple Description Coding) so all the streams are independent of each other. MDC allows receivers to obtain different subsets of the data but still maintain a decodable stream. Furthermore, a receiver can obtain progressive quality enhancements if it receives more streams, i.e., the more streams it receives, the higher quality it can obtain.

SplitStream can utilize the network resources more efficiently compared to the single multicast tree topology, since the leaf nodes of a multicast tree can be intermediate nodes of other multicast tree.

The disadvantages of SplitStream are three-fold. Firstly, the complexity of multiple multicast tree construction is high, especially when the trees are not well balanced. Secondly, compared to the single multicast tree topology, it consumes much more bandwidth to maintain those multicast trees and it has a higher latency in case of node dynamics (i.e., node joining, leaving) and node failures. Lastly, each multicast tree will affect the decoding quality of all the receivers. It may also result in low decoding quality since the media content is divided into multiple streams using MDC, and each stream is disseminated separately to all the receivers. The latency of multicast trees can be quite different from each other due to the physical topology of Internet. If some multicast trees have high latency, it will result in a big problem that the streams delivered by those trees become useless for receivers due to the time constraint of real-time media applications.

2.2.3 Mesh

We see that a leaf node's throughput is determined by its unique father in a multicast tree. A lot of problems will be posed in the tree structure. For example, any loss high up the tree will reduce the bandwidth lower down the tree (if the transport protocol is TCP or related), and the incoming throughput of a node is limited by its unique father. Moreover, the tree structure is vulnerable to system dynamics and node failures.

Mesh topology is a remedy solution such that a node can download data from multiple upstream nodes simultaneously (e.g., Bullet [28]). This can utilizes the network resources of all the participating users more efficient than tree structures, and adapt to the system dynamics.

Since a node can receive contents from multiple upstream nodes in a mesh, the data in those upstream nodes should be disjointed as much as possible. Otherwise a lot of bandwidth is wasted to transmit duplicated data. Therefore, each node in the mesh will purposefully disseminate disjoint objects to other nodes. The portion of data a child node can receive is proportional to the number of its descendants and the available bandwidth between the child node and its father.

Bullet uses RanSub [29] to locate disjoint objects within the mesh. Each Bullet node will receive a random subset which consists of all the nodes excluding its descendants (in the basic multicast tree). In order to recover the missing data from an upstream node without receiving duplicate packets, each node uniformly divides the sequence space in its current working set (which contains the sequence number of received packets) and then sends the requests to its upstream senders separately.

Compared to the single multicast tree topology (e.g., Scribe) and the multiple multicast tree topology (SplitStream), Bullet constructs a mesh to disseminate content to receivers which can receive contents from different upstream nodes simultaneously. The mesh topology can efficiently utilize the outgoing bandwidth of all the nodes in the system. In addition, Bullet uses the TCP friendly rate control protocol to transmit data which can fairly allocate bandwidth for downstream nodes and remedy the loss events that occur at the upstream nodes.

Since each receiver in a mesh topology needs to reconcile the received data packets to avoid receiving duplicate copies, it is desirable that every node sends disjoint objects to its children. When the number of children is small[1], it is hard to maintain the diversity of objects. Moreover, communication and computing overheads are incurred for reconciliation among multiple upstream nodes. Lastly, since mesh of Bullet is constructed based on a multicast tree, it is also vulnerable to node dynamics and failures.

2.3 3D Audio Streaming

Recent interest in NVEs (e.g., metaverses and virtual worlds) has resulted in a lot of research that has been conducted in the area of application-level game infrastructures for Massively Multiplayer Online Games (MMOG) [2, 30–36]. The currently favored design for transmitting game state information in NVEs is based on a client-server architecture that partitions the game world into many different *game zone* regions (often based on a grid layout). Each game zone is handled by one server machine to achieve load balancing across the cluster in terms of computations and network game traffic. Regions may be statically or dynamically assigned to servers [30].

One natural inclination is to extend the same client-server architecture to support interactive voice communication. An alternative approach is to use a P2P distribution architecture, which utilizes network resources of participating users. We can further differentiate between voice communication systems that provide monophonic audio and those that support spatialized, proximity audio. Table 2.1 categorizes existing voice streaming systems into four quadrants along two axis': client-server (or proxy) versus peer-to-peer, and monophonic versus spatialized audio. I will briefly describe some representative work of each quadrant.

- **Client-Server & Mono Audio.** From a technical perspective, this category represents the most traditional approach. For NVE applications, where the user is already situated in front of a computer, voice-over-IP

[1] It is true in reality that peers have much more incoming bandwidth (for downloading) than outgoing bandwidth (for uploading).

	Monophonic Audio	Spatialized Audio(3D)
Client-Server or Proxy	Ventrilo,TeamSpeak	DiamondWare, DICE [37]
Peer-to-Peer	Skype,AudioPeer [3]	**Proposed in this thesis**

Table 2.1: Categorization of interactive audio streaming techniques.

(VoIP) systems are the natural choice. Some such systems are completely separate from any games, while others are targeted towards virtual worlds and provide some form of integration (e.g., the commercial offerings from Ventrilo and TeamSpeak).

- **Peer-to-Peer & Mono Audio.** Peer-to-peer application-level multicasting or streaming topologies have become popular and successful academic prototypes and commercial systems have been built. One natural benefit of P2P is that each participating node contributes resources such as memory, bandwidth and processor cycles, which allows for very scalable designs. Some existing systems in this category are completely separate from any NVE application (e.g., Skype, AudioPeer [3] [38]), while others are targeted towards virtual worlds and provide some form of integration. In a P2P design distributed stream mixing needs to be performed [4]. Several P2P overlay multicast designs have been evaluated by Castro et al. [9].

  From the perspective of NVEs, the most important shortcoming of these monophonic systems is that they do not take advantage of the location, distance or the directionality between user avatars in the virtual world. Therefore, the aural environment does not match the visual perception of the user and – while functional – does not provide a seamless experience. Consequently, the work that is more related to our approach is concerned with providing an immersive, 3D aural environment. Spatialized audio rendering is now possible on commodity personal computer sound cards and is widely used for local (i.e., non-streaming) game sound designs. Libraries such as OpenAL allow for efficient rendering implementations and interest is growing to use spatialized sound in interactive communication applications.

- **Client-Server & 3D Audio.** With a client-server topology, 3D audio rendering is supported by capturing mono streams at each client and forwarding them to the central server for processing. There, complete information about all avatar locations is available and a personalized 3D sound mix is rendered and packed into a stereo stream to be sent back to each client. However, with this approach significant audio bandwidth and computational power for signal processing is required at the server. This can easily place an undue burden on the already taxed game server infrastructure. Boustead and Safaei [39] have surveyed a range of delivery options including central-server. They concluded that the capacity of central server will be an scalability barrier, and some form of distribution, such as peer-to-peer and hybrid model would be beneficial. To remove the central server bottleneck some proxy-based solutions [40] have been proposed to load-balance the bandwidth and processing load (e.g., DICE [37]). In this infrastructure, a group of users in a given geographical location are assigned to a geographically nearby server, which is referred to as a proxy, is responsible for the creation audio scenes for its attached participating users either using unicast or multicast. A commercial product in this space is offered by DiamondWare (which has been licensed by Vivox). In addition, Skype multiparty audio conferences currently support up to 25 people at a time, including the host due to the constrained server bandwidth, the client-server scheme is not a desirable solution for 3D audio streaming in NVEs, where multiple speakers need to send their audio streams to multiple receivers at the same time.

- **Peer-to-Peer & 3D Audio.** To the best of my knowledge there exists no comparable platform to the fully decentralized, peer-to-peer architecture that I am proposing for spatial audio streaming in this thesis. The closest variants are the proxy-based solutions mentioned in the previous paragraph. I expect that improved audio capabilities will become increasingly important to catch up with the visual developments of virtual worlds as more academic and commercial entities seek to build multimodal shared virtual

spaces for person-to-person interaction in mass populated online communities.

The work that is most related to my research is concerned with *many-to-many* spatial audio streaming applications, where every bandwidth constrained user can receive multiple audio streams from different speakers concurrently. Up to now, only little work has been done to address on the three important challenges in spatial audio interaction over NVEs.

To maximize the utilization of upload bandwidth, an approximation algorithm is proposed in [41] which constructs a depth-2 multicast tree (i.e., the depth of the tree is at most 2) for each user in a centralized manner. More specifically, the algorithm proposed in [41] first accommodating the nodes that have high requirement (e.g., large number of neighbors), but only little upload capacity (and hence can use only few relay nodes), and then it proceeds in a greedy fashion and allocates the bandwidth requirements (i.e., forms multicast trees) of nodes in non-increasing order of the ratio r_i/c_i, where r_i, and c_i denote the requirement and the capacity of node i, respectively. However, this solution only considers the static case where the number of recipients for each speaker is fixed and known as a priori. Hence, the system dynamics will incur frequent re-computation operations at the server which requires a lot of computational resources, and thus is not desirable in practice.

In this thesis, I propose two types of solutions based on P2P technologies. The first one is the Intra-AoI approach [21], where a minimum latency tree (MLT) is constructed at each speaker. The root of each MLT, i.e., the speaker, can only utilize the upload bandwidth of its recipients. Inevitably the system reachability will degrade significantly as the number of speakers increases. Moreover, the QoS will degrade as well due to the changes of the dissemination topology resulting from system dynamics. To further improve the system reachability, cross-tree adjustment technique (i.e., CTA) is proposed to allocate the bandwidth on the conflict nodes incurred by multiple speakers.

As described in [39], many factors influence the appropriateness of the delivery architecture including: the characteristics of the virtual environment, rate of

movement, distribution and density of avatars. No current solutions (i.e., client-server, peer-to-peer, and hybrid model) can achieve high system reachability and low latency at the same time, Hence I propose two game-theoretic approaches which can achieve both the near-optimal system reachability, and nearly the same low latency of the client-server approaches.

2.4 Texture Streaming

Peer-to-peer or peer-assisted 3D streaming systems for networked virtual environments have been studied recently [42] [43] [44]. They allow peers to store the 3D contents they receive in caches, and share their network resources and caches to serve others.

Hu *et al.* [42] proposed FLoD, which is the first framework for peer-assisted 3D streaming in metaverses. FLoD enables each peer to send its requests to a randomly chosen AoI neighbor which may send the 3D contents to the peer, thus reduce the server bandwidth cost. If the AoI neighbors fail to respond, the peer will send the requests to the server. To minimize the server bandwidth cost in FLoD respecting the bandwidth constraints of peers, Chien *et al.* [44] proposed a bandwidth aware peer selection algorithm, named BAPS, that improves the utilization of peer bandwidth via bandwidth reservation. In particular, BAPS allocates the upload capacities of peers into channels and each peer will select the provider that in the currently connected channels; if the provider is overloaded, it will pick a peer from AoI neighbors as the provider; if that fails, it will randomly select a peer beyond of the AoI; eventually the request will go to the server if no appropriate provider is found consuming the bandwidth of the server. Varvello *et al.* [43] proposed a structured P2P architecture (implemented on the top of Kad [45]) where all the 3D objects are stored in the participating peers instead of centralized servers. Peers need to query their required objects in Kad, and wait for the data before rendering those objects in their virtual environments. This pure P2P architecture suffers from low consistency (due to the query delay) and persistency (due to the avatars' mobility).

Our objective is to minimize the server bandwidth cost by enabling peers to rapidly select the underloaded content providers in a decentralized manner. The

key idea is to balance the loads of peers. A number of algorithms have been proposed for dynamic load balancing, which is modeled as a weighted congestion game with selfish and non-cooperative users. A steady state in this context is a pure Nash equilibrium, in which no user is willing to change her/his strategy. Since the idea of using a potential function to measure the closeness to a Nash equilibrium was introduced by Even-Dar *et al.* [46], a number of protocols (e.g., [47] [48]) have been proposed to balance the loads sequentially, i.e., the migration events take place one at a time, and costs on users are updated immediately. More recently, concurrent protocols (e.g., [49] [50] [51]) for load balancing have been proposed to improve the convergence time by allowing users to change their strategies concurrently. We adopt the concept of these load balancing techniques by applying it to peer selection strategy in peer-assisted 3D streaming systems.

The proposed peer-assisted texture streaming system distinguishes itself from all existing systems in the following two important aspects. First, I consider the latency requirements of 3D contents, and design a light-weight peer selection strategy to minimize the server bandwidth respecting the latency constraints of contents and the bandwidth constraints of peers. Second, the peer' bandwidth is optimally utilized in a decentralized manner.

3

Research Overview

In this chapter, I will outline the research topics that I have addressed in this thesis. To the best of my knowledge, there exists no comparable platform to the architecture that we are proposing for P2P-based 3D audio streaming over NVEs. My goal is to design a server-less 3D audio streaming architecture for NVEs while leveraging the scalable properties of P2P distribution topologies and achieving a low delay, accommodating system dynamics, and respecting the network bandwidth constraints of heterogeneous peers.

Note that the objective of peer-assisted texture streaming is to minimize the server bandwidth consumption through efficiently utilizing the network resources of the participating users. However, current live or VoD P2P streaming approaches cannot be easily applied for texture streaming applications in NVEs due to the unpredictable and non-linear request patterns of users. In this thesis, I investigate the feasibility of applying game-theoretic approaches to 3D content delivery in NVEs, and propose a game-theoretic peer selection strategy for peer-assisted texture streaming applications over NVEs.

3.1 Intra-AoI Multicast Tree Approach for 3D Audio Streaming

The multicast tree approach is a straightforward technique to deliver media data to a large number of receivers, and it has been extensively applied to large-scale

media streaming applications. In these applications, there is only one or little amount of *fixed* sources but a large number of receivers. In contrast, every user in NVEs can speak at any time any where, and this can result in a large number of senders as well as a large number of receivers. In addition, a user may receive multiple audio streams from different speakers at the same time, which mimics the communications in the real world environments.

My objective is to support live mass interactive 3D audio communication in NVEs. Therefore the content delivery topology I am proposing in this thesis is to disseminate audio streams from speakers to all their receivers (users in their AoIs) as soon as possible. Due to the limited bandwidth of users, not all users can successfully receive the audio streams they are supposed to. An optimal multicast tree should be constructed for each speaker respecting to the bandwidth constraints of all the participating users, and achieving the minimum average latency. This problem can be formulated as a degree-bounded minimum shortest path tree problem which has been proved as a NP-hard problem [52].

Although some recent works [10, 53, 54] have proposed approximate solutions to this problem, my work will differ from those previous work such that I focus on multiple multicast trees, which make it more difficult since the multicast trees may influence each other. I propose an Intra-AoI dissemination tree topology, called MLT [21], which utilizes the upload bandwidth of neighbors of each speaker via constructing a heuristic Minimum Latency Tree (MLT) rooted at the speaker.

Recall that MLTs are constructed based on the neighbors of speakers in NVEs. As well as joining or leaving the NVEs, avatars are moving around in NVEs at will and the set of neighbors of every avatar will be updated frequently and unpredictably. Consequently, it is difficult to maintain the multicast trees in NVEs, since the state messages need to be exchanged more frequently within AoIs, and this may increase the network load. In addition, multicast trees need to be modified more frequently which may increase the computational complexity. In [55, 56], a centralized algorithm is proposed to adjust the edge weight changes in the shortest path tree. However research on multiple overlapped multicast tree adjustment still need to be conducted to further improve the performance. To this end, I propose a heuristic distributed algorithm [22] which is performed at

each user and enables users to dynamically adjust the multicast trees respecting to the bandwidth limits of neighbors.

The proposed design is targeted to delivery 3D audio with scarce network resources. Stream mixing can applied to increase the number of successful receivers. However some spatial information (i.e., locations of speakers) could be lost when mixing is applied. Boulstead et al. [57] have introduced a foreground versus background distinction that places more importance on the accurate sound rendering from close sources. This concept can be combined with clustering and incorporated into the design of 3D audio streaming to further improve the system reachability.

3.2 Game-theoretic Approaches for P2P-based 3D Audio Streaming

Game theory has been widely used to model both the behavior of participants in P2P networks, and the efficiency of P2P networks. This enables the design of dissemination topologies which allows each user to operate individually and simultaneously, and achieves an optimal performance with respect to bandwidth limits of users, churn, and avatars' mobility in NVEs. Therefore, I believe that a 3D audio streaming system can be characterized in a game-theoretic setting, and thus inherit its decentralized and practical nature.

Note that the convergence rate of distributed algorithms is a particularly important factor for enhancing the performance of time-critical applications (e.g. media streaming) running on large networks such as the Internet. My goal is to design game-theoretic distributed algorithms for 3D audio dissemination that achieve fast convergence to a near-optimal performance in terms of system reachability and end-to-end latency.

Considering that users have heterogeneous bandwidth limits, I design two game-theoretic algorithms to utilize users' bandwidth in two different ways: 1). Firstly, I propose an abstraction layer on the top of the P2P overlay network, and all the helpers (derived from users in the P2P overlay network) in the abstraction layer are used to forward audio streams. By setting the capacity of every helper

to a fixed value, I transform the heterogeneous problem into a homogeneous problem, i.e., how to use these helpers with the same capacity to deliver 3D audio streams. 2). On the other hand, I propose a fully distributed game-theoretic algorithm which targets the heterogeneous problem directly. I prove that both algorithms converge to an approximately balanced state from any prior state within $O(\log \log n)$ rounds, where n is the number of users. The evaluation of both algorithms will be shown in Chapter 5.

3.3 Game-theoretic Approach for Peer-assisted Texture Streaming

For peer-assisted texture streaming in NVEs, I formulate the problem of server bandwidth minimization problem as a congestion game, and design a peer selection strategy to enable peers to efficiently utilize the network resources. Different to the game-theoretic approaches for 3D audio streaming, I use a different game model which is calibrated to model the server bandwidth minimization problem. In addition, I use different techniques to prove the convergence rate of the proposed approach, which will be detailed in Chapter 7.

4

Intra-AoI Approach for P2P 3D Audio Streaming

In this chapter, I will introduce a straightforward approach for P2P 3D audio streaming over NVEs. This is the starting point of my research, and the goal is to delivery the audio streams from speakers to their receivers while respecting the bandwidth limits of the participating users. A multicast tree, termed MLT (i.e., Minimum Latency Tree), is constructed for each speaker. Since all the members of the multicast tree reside in the AoI of the speaker, this approach is called *Intra-AoI* approach. To further improve the reachability (which is defined as the fraction of the receivers that successfully receive their streams.), audio mixing and cross-tree adjustment (CTA) techniques can be applied in the Intra-AoI tree approach.

With traditional voice conferencing only a single, mono channel is required and the voices of different participants can be mixed together at appropriate node locations within the network. However, proximity 3D audio requires that multiple voice channels be kept separate until they are rendered and merged into a three-dimensional aural voice field on the participant's client computer. Consequently, 3D voice traffic is voluminous and may consume more bandwidth compared with other types of game traffic, and thus the design of an architecture for 3D audio streaming presents formidable challenges.

Firstly, to create an immersive audio experience for users, audio streams from different speakers need to be rendered according to the spatial and directional

relationship between receivers and speakers in the virtual world. Therefore, audio streams cannot be mixed until they reach their destinations. This indicates that each speaker needs to construct a multicast tree, and an avatar in the NVE may receive multiple audio streams from different speakers at the same time. Since an avatar can be in multiple multicast trees, a challenging problem is how to allocate the bandwidth of corresponding users.

Secondly, the physical movements of the avatars naturally control the audio stream distribution through the underlying streaming topology. An avatar can stroll through the virtual setting listening to conversations of interest and speaking to other avatars when desired. This may incur frequent changes of multicast trees, since the movement of avatars has significant impact on the distribution topology.

Finally, the end-to-end delay from the source to the receiver(s) needs to be minimized so that meaningful real-time interaction can occur. The delay performance of an overlay multicast tree is determined by several factors such as link quality, topology configuration, bandwidth capacity. A streaming architecture must carefully minimize the delay based on all these different parameters.

I feel that a successful solution will enable capabilities beyond what is in existence today. The shared space will much more realistically imitate a physical setting by adjusting sound, volume and directionality according to avatar proximity and position. Because the audio distribution within the virtual environment is based on proximity and hence localized, a massive scale is possible that cannot be achieved with traditional voice conferencing.

Before I detail the Intra-AoI approach, I will briefly review the relevant concepts of area of interest (AoI) modeling for the hearing range and spatialized audio rendering as they relate to the approach.

Area of Interest Modeling. In NVE-type applications, the area of interest (AoI) of a user is usually defined as the space around the user in this shared virtual world where interaction, either between users or between user and the world, is likely to happen. Previous research has explored the temporal and spatial locality for interaction among users in large-scale systems [1, 2]. It has been shown that users are more likely to interact with each other when their

positions are close in the virtual world. I believe that this observation extends to streaming services in NVE-type applications, i.e., users who share their AoIs are also more likely to stream from each other. In our audio streaming application scenario, the AoI corresponds to the hearing range of each avatar. I currently model the hearing range as a circular area centered at the user's position. In reality large objects (such as walls, buildings, etc.) may obstruct the audio field. However, I will use the simplified, circular model initially and expand my work to include a more accurate sound environment in the future. I currently do not explicitly model different levels of audio volume within the AoI. The audio volume could be attenuated towards the edge of the hearing range. This can be achieved in the rendering engine, based on the distance information that is available for each audio source.

Spatialized Audio Rendering. Audio rendering to position sound sources across a spatial volume is now available on commodity personal computers. Such capabilities are usually contained in software libraries that can take advantage of different sound card hardware and transducer setups. For example, audio may be rendered via 2, 5.1, 6.1, 7.1 speakers, or via headphones. Most existing 3D sound processing modules require that the audio from different sources is kept separate until rendered. There are algorithms to re-generate 3D immersive audio from only one or two audio input channels, but the results are not very satisfactory. Because the send and receive network capacity for each node in a peer-to-peer network is limited and heterogeneous, it becomes very challenging to utilize the network resources effectively to deliver several separate audio streams while at the same time keep the overlay delay bounded.

4.1 System Model

The NVEs that I am considering operate on two different planes (see Figure 1.2). The physical space consists of routers connected by links. The end systems are connected to this network at different points through access links. The overlay network is modeled as a graph $G(V, E)$, where V is the set of vertices and $E = V \times V$ is the set of edges. Each vertex $v \in V$ represents a user or end system.

Every edge $e \in E$ denotes the unicast path between the two end vertices of the edge.

The virtual space is represented as a 2D square of side-length l and formalized as a dynamic graph. Each avatar uses coordinates $\langle x, y \rangle \in [0 \dots l, 0 \dots l]$ to represent its location in the virtual world. The distance between two avatars is defined as the Euclidean distance between them. There is a one-to-one mapping between users in the physical space and avatars in the virtual space. The term *node* is referred to both a user in the physical space and an avatar in the virtual space. In the rest of the chapter we will use the terms node, user, peer, and avatar interchangeably.

For identification purposes each node has a unique global id in the virtual space. For example, node i is denoted by n_i. The users (avatars) in an NVE can move in the system at will and any two users can communicate with each other when their distance is less than r, which is the radius of a node's area-of-interest (AoI). The AoI (i.e., hearing range) of an avatar is represented as a circle centered at the avatar's position, with the AoI radius being equal for all avatars. An avatar v_1 within an avatar v_0's AoI is termed a *neighbor* of v_1 (see Figure 4.1 for examples). The set of neighbors of node v_i, denoted by N_i, is the set of receivers of the audio stream sent by v_i. It is assumed that location and AoI information of each node are maintained at the game server. In addition, every user periodically exchanges information (i.e., bandwidth usage and end-to-end latency) with its neighbor nodes. Existing algorithms and methods exist for this purpose and I do not provide any details in this thesis.

The access link of each node can sustain a certain upload and download bandwidth (which usually is asymmetric). The upload/download capacity is measured in units, where 1 unit is defined as the required bandwidth for a single audio stream and measured in bits/s. In this thesis, I assume that audio communication may only use a certain fraction of the available bandwidth and that the rest is allocated to the game traffic. The upload and download capacity of node i, as the number of audio streams it can send to or receive from other nodes, are denoted as u_i^{out} (outdegree) and u_i^{in} (indegree), respectively.

Transmission of any audio stream is restricted within the AoI of the speaker who sent the stream. That is, all nodes outside of the hearing range of the

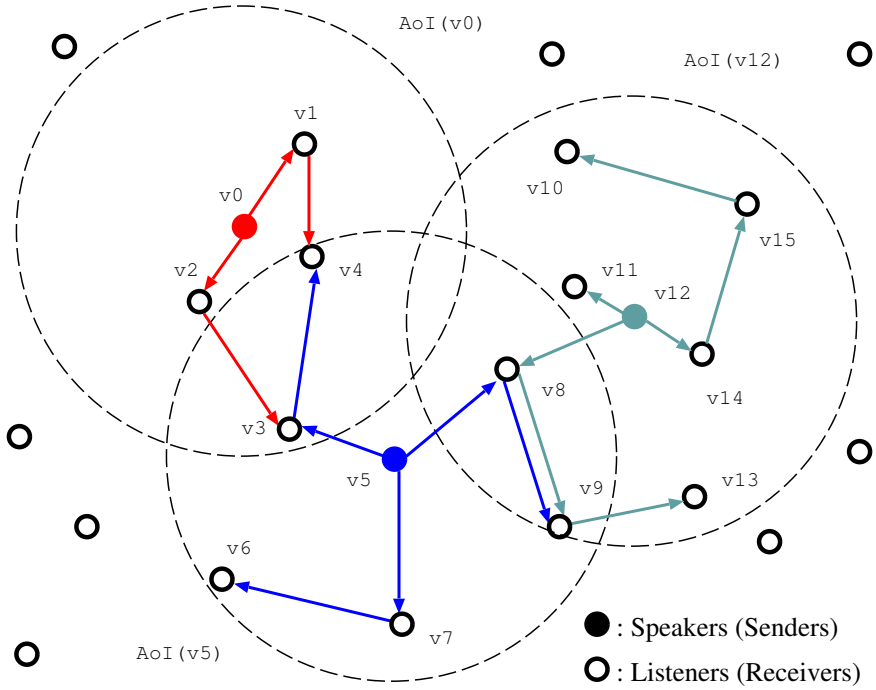


Figure 4.1: Intra-AoI 3D audio dissemination topology with three speakers, v_0, v_5 and v_{12}.

speaker will not receive or forward the audio stream. Therefore when a node starts speaking, all its neighbor nodes are candidate receivers for the audio stream sent from the speaker. Due to the bandwidth limits of the neighbor nodes and the speaker, only a fraction of the candidate receivers, called successful receivers, may actually be able to receive the stream. The *system reachability* is defined as the fraction of the receivers that successfully receive their streams.

The goal is to form a distribution topology through which each sender can successfully reach all of the receivers within its AoI. Multiple topologies have been suggested for this purpose in the past, for example meshes, rings, trees, and multi-trees. One important objective of the design is to respect the upload and download capacities of each node. Oftentimes, existing work on peer-to-peer streaming either considers artificial or no bandwidth limits for end systems. The Intra-AoI approach is based on spanning trees, since tree structures provide a good compromise between the maximum path length (which affects the end-to-end latency) and the in- and outgoing bandwidth required at each node. The

Term	Definition
G	Completed graph representing the virtual space
V	Set of all nodes in G
E	Set of edges in G
S	Set of senders in G
N_i	Set of neighbors of node v_i, $v_i \in V$
u_i^{in}	Download capacity of node i
u_i^{out}	Upload capacity of node i
$L(i,j)$	End-to-end latency of path from node i to node j
T_i	Multicast tree rooted at node i, T_i is represented by adjacency-list model.
$V(T_i)$	Set of nodes in T_i
$D_i(j)$	Set of descendants of node j in T_i
$Core_i$	Set of non-leaf nodes of T_i
$Leaf_i$	Set of leaf nodes of T_i
O_i	Set of unsuccessful receivers of T_i.

Table 4.1: List of terms used in the Intra-AoI approach.

main challenge is that many variants of degree-constrained minimum spanning trees (including Intra-AoI trees) have a high computational complexity (or are NP-complete [58–61]). Next I will state the problem more formally and then investigate solutions that are augmented with heuristics.

4.2 Problem Formulation

Recall that avatars in an NVE can speak at will, there are usually multiple speakers at any time. Based on the assumption that every avatar within the AoI of the speaker is supposed to receive the audio stream, the P2P 3D audio streaming problem can be treated as a many-to-many audio streaming problem where every node in the overlay network could be the source of one audio stream and the receiver of one or several others. Since the set of receivers is (usually) different for each sender, an applicable solution is the construction of a forest of multicast trees, with the trees rooted at all the senders. Furthermore, the

indegree and outdegree of every node in the overlay network is bounded. The define the problem as constructing a *forest of degree-bounded minimum latency multicast trees*:

Given an undirected graph $G = (V, E)$ representing the virtual space, degree bounds u_i^{in} and u_i^{out} for each node v_i in V, and a set of senders S in V. Construct a minimum latency multicast tree (e.g., T_i) for each sender (e.g., v_i) such that the maximum system reachability R is achieved. That is,

$$\text{Maximize:} \quad R = \sum_{i \in S} R_i \tag{4.1}$$

$$\text{Minimize:} \quad \frac{1}{R} \sum_{i \in S} \sum_{j \in N_i} L(i,j), \tag{4.2}$$

$$\text{Subject to:} \quad \sum_{j \in V} e(j,i) \leq u_i^{in}, \forall i \in V \tag{4.3}$$

$$\sum_{j \in V} e(i,j) \leq u_i^{out}, \forall i \in V, \tag{4.4}$$

where $R_i, i \in S$ denotes the reachability of the multicast tree rooted at sender v_i, and $e(j,i)$ is a indicating variable indicating whether node j is transmitting a audio stream to node i. $L(i,j)$ refers to the end-to-end latency of the path from node i to node j.

4.3 The Intra-AoI Tree Approach

Finding an optimal solution to the multi-objective problem stated above is very challenging. Combining the different objectives into one scalar goal and finding a Pareto optimal point may not be feasible due to the computational complexity that is introduced by the degree bounds. Therefore, our proposed method is to find a solution via a multi-stage process.

Fortunately, the streaming of audio samples has a characteristic that can help us to reduce conflicts in the in- and outdegree usage. A *conflict* occurs when the upload (or download) bandwidth required for a node exceeds its upload (or download) capacity, as shown in Eq. 4.4 and Eq. 4.3 respectively. Recall that not all candidate receivers may be able to receive the stream due to degree constraints,

multiple streams may be merged (i.e., mixed) together by combining their digital sample values. The resulting stream requires the same bandwidth as each of the input streams. As shown in Figure 4.2, node i can mix two audio streams (denoted by $AS(v_m)$ and $AS(v_n)$ respectively) together and send the mixed stream (AS_{mix}) to node j. In this way, one outdegree of node i and one indegree of node j are saved while node j can still render the two streams. The spatial information of the mixed stream is derived from the spatial information of original streams and the receiver. However, spatial information loss may occur in this process.

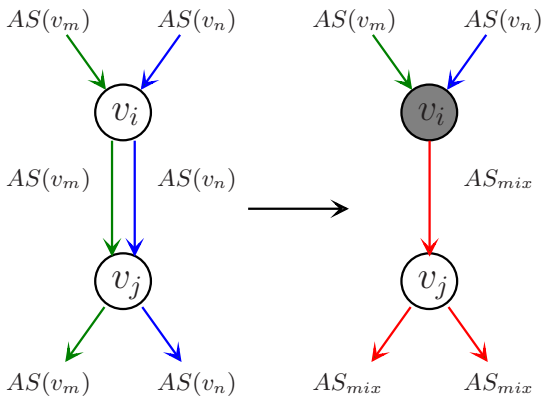


Figure 4.2: Audio streams mixing.

Two important issues arise with stream mixing. First, the mixing process is *irreversible* which means the combined streams cannot be split again. As a result, streams should only be mixed under specific circumstances which will be described in a subsequent section. Second, some spatial information loss may occur due to the mixing process. Recall that each audio source is associated with a sender location in the virtual space. When merging two sound streams their spatial locations also must be combined. For example, node j cannot differentiate the locations to the two speakers (senders of the two audio streams) based on the mixed stream. To minimize the error one may associate the merged stream with a 'phantom' source located in the middle between the two original sources. The spatial information loss of the stream merging procedure will be quantified in the experimental section. Note that spatial information loss of a closely positioned source may be more noticeable than for a source located at

a greater distance. However, more investigation is necessary to quantify such detailed effects adequately.

With the addition of the stream mixing procedure the optimization goals of the Intra-AoI approach are: (1) maximizing the system reachability, (2) minimizing the end-to-end latencies and (3) minimizing the spatial information loss when some streams are mixed. The Intra-AoI approach is divided into two processes. The first process is performed on every speaker. Subsequently the second process is executed on the neighbors of each speaker.

Process 1: **MLT Construction.** This process is executed at every speaker, and it has three phases with the goal of creating a intra-AoI multicast tree T_i rooted at the speaker i while respecting the in- and outdegree limits of all the nodes within the AoI of i (i.e., the speaker and its neighbors). During this process the nodes within the AoI of the speaker are partitioned into three disjoint sets denoted by $Leaf_i$, $Core_i$ and O_i. $Leaf_i$ contains all the leaves of tree T_i, and $Core_i$ is defined as $V(T_i) - Leaf_i$. O_i contains all the receivers that were unsuccessful in connecting to T_i due to the in- and outdegree constraints of nodes in $V(T_i)$ (see Figure 4.3 for an example). At the end of this process the nodes in $Leaf_i$ and O_i will be notified to execute the process 2.

Process 2: **Audio Stream Mixing.** This process is executed at the receivers. During this process, unsuccessful receivers and leaf nodes in the intra-AoI multicast tree T_i, i.e, $Leaf_i$ and O_i, can select nodes in the core of T_i to mix streams for them. The goal is to improve the reachability. If there are two or more nodes in the core that can mix streams for the unsuccessful node, the one which results in the least spatial information loss is chosen.

4.3.1 MLT Construction

Now I will elaborate on the details of the MLT construction process. As an assumption, the state information is exchanged periodically among nodes such that every node is aware of information about its neighbors (i.e., the latency matrix, and available outdegrees and indegrees within the AoI). With the state information of its neighbors, each node can construct a intra-AoI multicast tree using three sub-phases of MLT construction, the output of which is a partitioning

of all the nodes within the AoI of speaker i into three sets: $Core_i$, $Leaf_i$ and U_i. As illustrated in Algorithm 1, the MLT construction process works as follows:

Algorithm 1: Process 1 of Intra-AoI approach - MLT construction.

```
1 foreach i ∈ S do
2     O_i ← ∅, Leaf_i ← ∅, Core_i ← ∅;
3     foreach j ∈ N_i do
4         if Σ_{k∈V} e(j,k) = u_j then
5             O_i ← O_i ∪ {j};
6     T_i ← MulticastTreeConstruction(i, L_i, N_i);
7     T_i ← MulticastTreeAdjust(T_i);
```

1. In the first phase of MLT construction process, the unsuccessful receiver set O_i is derived from all the neighbors that have no more available indegrees (they may be participating in other multicast trees), as shown in lines 3-5 of Algorithm 1.

2. During the second phase a Intra-AoI tree is constructed based on all the nodes within the AoI of the speaker i expect the nodes in O_i, and rooted at the speaker. Here we employ a traditional algorithm such as minimum spanning tree [62] (MST) in this phase, i.e., MulticastTreeConstruction(i, L_i, N_i) (indicated by line 6 of Algorithm 1), where L_i and N_i are the latency matrix and the set of neighbors of speaker i. In this stage, a MLT is constructed by combining all shortest paths from each speaker to its neighbors without considering the bandwidth limits of the neighbors, and it ensures that all the successful receivers observe the minimum latency. The time complexity of MLT construction is $O(|N_i| \log |N_i|)$, where $|N_i|$ denotes the number of neighbors of speaker i.

3. Note that these standard algorithms do not respect the in- and outdegree limits, the traditional MST algorithm is modified in the third phase such that the number of MST members is maximized with respect to the in- and outdegree limits of nodes. After this step, not all neighbors are able

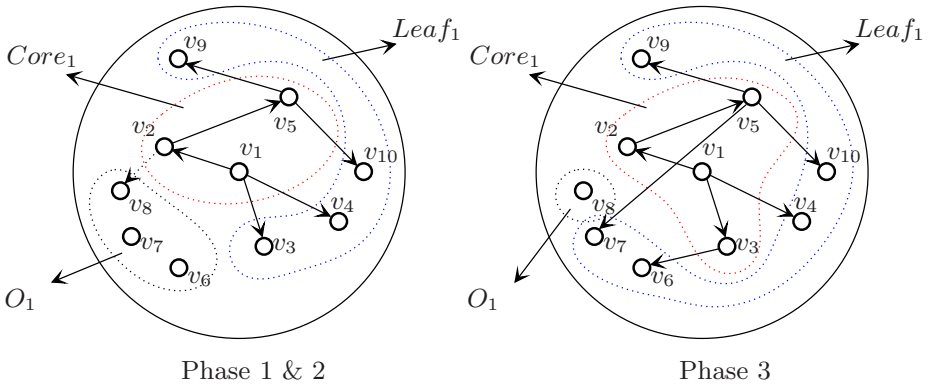


Figure 4.3: The two processes of the Intra-AoI approach.

to attach to the tree. Therefore MulticastTreeAdjust(T_i) (line 7 of Algorithm 1) is applied in the third phase to maximize the reachability. The heuristic method used in MulticastTreeAdjust(T_i) (line 3 of Algorithm 2) calculates a *contribution value* (CV) for each node in the MST, and the CV of a node reflects the maximum number of nodes that can be reached in its subtree under conditions of its in- and outdegree bounds. In other words, the contribution value is the weight of the node, denoting how many number of descendants it can forward the audio stream to according to the upload/download bandwidth limits of thenode. As illustrated in Algorithm 3, a depth-first search (DFS) is performed to calculate the CV of each node in the MST that was constructed in the second phase. At last, nodes in O_i will be attached to T_i to improve the reachability, as shown in line 21 of Algorithm 1.

The time complexity of MulticastTreeAdjust(T_i) is $O(|N_i| \log N_i + 3N_i)$. Specifically, the loops of lines 1-3 and lines 16-20 require $O(N_i)$ time. Lines 5-15 apply a breadth-first search, and thus take $O(N_i)$ time. ComputeCV(T_i, CV) (shown in Algorithm 3) takes $O(N_i \log N_i + N_i)$ time. As a result, the total time complexity of MLT construction is $O(2N_i \log N_i + 3N_i)$.

The example shown in Figure 4.3 illustrates the overall process. All the nodes within the AoI of speaker v_1 will be partitioned into three sets: $Core_1$, $Leaf_1$ and O_1. After the MLT construction process, $Core_1 = \{v_1, v_2, v_5\}$, $Leaf_1 =$

Algorithm 2: MulticastTreeAdjust(T_i) in Process 1 of the Intra-AoI approach.

```
1  foreach j ∈ V(T_i) do
2      CV[j] ← 0;
3  ComputeCV(T_i, CV, i);
4  Let M ← {i};
5  foreach j ∈ M do
6      Let count be min(a_j, |T_i[j]|), where a_j and |T_i[j]| denote the available
       oudegree of j and the number of children of j in T_i, respectively;
7      Sort T_i[j] by the value of CV;
8      while count > 0 do
9          T_i ← T_i − {T_i[j][0]} − D_i({T_i[j][0]});
10         T_i[j] ← T_i[j] − {T_i[j][0]};
11         M ← M + {T_i[j][0]};
12         count ← count − 1;
13     if |T_i[j]| ≠ 0 then
14         O_i ← O_i + T_i[j] + D_i(T_i[j]);
15     M ← M − {j};
16 foreach j ∈ T_i do
17     if |T_i[j]| > 0 then
18         Core_i ← Core_i + {j};
19     else
20         Leaf_i ← Leaf_i + {j};
21 T_i ← AttachToTree(O_i);
```

$\{v_3, v_4, v_9, v_{10}\}$, and $O_1 = \{v_6, v_7, v_8\}$. Each node in $Core_1$ and $Leaf_1$ will be assigned a CV by Algorithm 3. In this example, node v_2 only has 1 available outdegree but it has two children in T_1, i.e., nodes v_5 and v_8. By Algorithm 2, it will choose the node which has the largest value of CV (node v_5 in the example) as its receiver, and as a result, node v_8 will be moved to O_1. At last, after Algorithm 2 is executed, all nodes are partitioned into three sets, and the sender

Algorithm 3: ComputeCV(T_i, CV, j) in Process 1 of the Intra-AoI approach.

```
1 visited[j] ← 1;
2 if |T_i[j]| = 0 then
3     CV[j] ← 0;
4     return;
5 foreach k ∈ T_i[j] do
6     if visited[k] = 0 then
7         ComputeCV(T_i, CV, k);
8         CV[j] ← CV[j] + CV[k] + 1;
```

v_1 will send messages to nodes in O_1 to trigger audio mixing process.

4.3.2 Audio Stream Mixing

As described above, when a node becomes a speaker, it will send the audio stream through the Intra-AoI tree it maintains. Due to the in- and outdegree bounds of the neighbors, some candidate receivers may not be able to receive all their streams. The audio mixing technique is applied to further improve the reachability by mixing two or more audio streams without violating the rule that each avatar can only hear from other avatars within its AoI.

To initiate the audio mixing process, the speaker i will broadcast a message to nodes in O_i to trigger the mixing node selection. As shown in Algorithm 4, every node j in O_i will search for a node in $Core_i$ which can mix audio streams for it based on Theorem 1. Furthermore, if two or more nodes in $Core_i$ can accomplish the mixing, the node which will result in less spatial information loss is chosen. The objective of audio mixing is to further increase the reachability while at the same time saving network resources. A drawback of this approach is that it will cause some spatial information loss for some nodes, since it is a irrevocable process. However, I think it is generally preferable to receive a stream, albeit slightly misaligned in space, compared to receiving no streams at all.

Theorem 1. *Let v_j be a node in $Leaf_m$ or U_m of speaker n_m. Let v_i be a node in $Core_n$. The stream sent by v_n can be mixed at v_i if and only if $v_i \in Core_m \wedge v_j \in T_m[i] \wedge v_j \in Leaf_m$, where $v_m, v_n \in S$ are two different speakers.*

Proof. I prove the audio mixing theorem by contradiction. If v_i can mix the audio streams for v_j, the following conditions should be satisfied: (1) $v_i, v_j \in N_m \cap N_n$, where $v_m, v_n \in S$ denote two different speakers. That is, both v_i and v_j must be neighbor of at least two speakers, since a mixing operation needs at least two audio streams. (2) $v_i \in Core_m, v_j \in D_m(v_i)$. That is, a node (e.g., v_j) should be a descendant of the other node (v_i), since the mixed stream ($AS_n + AS_m$[1]) that v_j received should be forwarded by v_i. (3) $v_j \in Leaf_m$. Since the descendants of v_j in T_n may not be in the AoI of v_m, which means they should not receive the mixed stream, v_j should be a leaf node in T_m. Otherwise, v_j will forward the mixed stream ($AS_m + AS_N$) to descendent nodes in T_m, and this may violate the rule that only neighbors can receive the audio stream. (4) $v_j \in T_m[i]$. That is, node j is a child of node i in T_m. If there is one or more nodes on the path from v_i to v_j in T_m, and these nodes are not the members of v_n, the two streams AS_m and AS_n cannot be mixed at v_i. As a result, the four conditions can be combined into the expression $v_i \in Core_m \wedge v_j \in T_m[i] \wedge v_j \in Leaf_m$, where $v_m, v_n \in S$ are two different speakers. □

Algorithm 4: Process 2 of the Intra-AoI Approach - Audio Mixing.

```
1 foreach j ∈ O_i do
2     M_j ← ∅;
3     foreach k ∈ Core_i do
4         if ∃n ∈ S, k ∈ Core_n ∧ j ∈ T_n[k] ∧ |T_n[j]| = 0 then
5             M_j ← M_j + {k};
6     Sort M_j in a non-decreasing order of ∠ijn;
7     Mixing the audio streams AS_i and AS_n at node M_j[0];
```

[1] AS_m could be a mixed stream.

4.4 Cross-Tree Adjustment (CTA) for the Intra-AoI Approach

As described in the preceding sections in this chapter, I have proposed an Intra-AoI approach [21], which is based on local tree adjusting (LTA) algorithm (i.e., MLT construction) designed to maximize the reachability while keeping the latency for those receivers as low as possible. This straightforward approach can be quickly employed at each speaker, but it is unfair to new speakers. Because LTA can fully utilize the available network resources for each speaker to construct an Intra-AoI tree, and the network resources may be overspent by existing speakers.

In this section, I present a heuristic cross-tree adjustment (CTA) algorithm to address this fairness issue by constructing and adjusting multiple Intra-AoI trees using a greedy approach. With CTA, speakers construct multicast trees incrementally. The newly constructed multicast tree and the existing multicast trees who have overlapped node with the new tree will be adjusted to respect to the bandwidth limits of nodes. The simulation results demonstrate that CTA can outperform LTA both in terms of reachability and the ratio of receivers that experience the minimal latency.

4.4.1 Problem Formulation

Recall that CTA addresses the fairness issue, the problem can be formulated to a NP-complete problem – constructing a forest consists of multicast trees rooted at all the speakers such that (1) the system reachability is maximized, and (2) the average latency of all the receivers is minimized:

$$\text{Maximize:} \quad R = \sum_{i \in S} R_i \tag{4.5}$$

$$\text{Minimize:} \quad \frac{1}{R} \sum_{i \in S} \sum_{j \in T_i} L(i,j) \tag{4.6}$$

$$\text{Subject to:} \quad \sum_{j \in V} e(i,j) \leq u_i^{out}, \forall i \in V, \tag{4.7}$$

where the first optimization objective is to maximize the system reachability subject to nodes' bandwidth limits. It should be noted that the limits of download

bandwidth of nodes are relaxed. Based on a realistic upload/bandwdith capacity dataset collected from dslreports.com [63], users usually have asymmetric network connections such that they have much more download bandwidth than upload bandwidth, thus we assume the download bandwidth is not the bottleneck of the system. Once the first objective is satisfied, the average latency of those successful receivers is to be minimized.

It is important to note that the Intra-AoI approach treats MLTs individually, while CTA enables multiple MLTs to corporate to improve the system reachability. Instead of achieving the two objectives sequentially, CTA makes a compromise between the two objectives. The key challenge of CTA is how to allocate the upload bandwidth of heterogeneous nodes for multiple Intra-AoI trees (MLTs) they are associated with.

4.4.2 Overview of CTA

With CTA, the multicast tree for each speaker i, T_i, consists of a shortest path tree (SPT), denoted by T_i^{spt}, and other orphan nodes attached to it. Hence, T_i^{spt} is a subgraph of T^i, i.e., $T_i^{spt} \subseteq T^i$. As shown in line 6 of Algorithm 1, the temporal multicast tree for speaker v_i is defined by T_i^{tmp}, which is a SPT constructed without considering in- and outdegree bounds of nodes. The heuristic is that the larger $|T_i \cap T_i^{tmp}|$ is, the more likely the multicast tree will have both high successful receiving ratio and low average latency. Therefore the goal of CTA is to maximize $|T_i \cap T_i^{tmp}|$ for all multicast trees in the system.

The newly multicast tree, T_i (for speaker i), is constructed in two stages. Firstly, once a new speaker i starts speaking, a temporal multicast tree, T_i^{tmp}, for speaker i is constructed (by Algorithm 2) based on information of its neighbors (every node is aware of available bandwidth of its neighbors and end-to-end latency between any two neighbors via periodical state information exchange). T_i^{tmp} is constructed without considering upload bandwidth limits of nodes within its AoI. The time complexity of construct T_i^{tmp} is $O(|N_i| \log |N_i|)$ when Fibonacci heap is applied.

After T_i^{tmp} is constructed, post-order tree traversal (with time complexity $O(|N_i|)$) is applied to divide the nodes of T_i^{tmp} into different sets. Let $T_i^0 = T_i^{tmp}$,

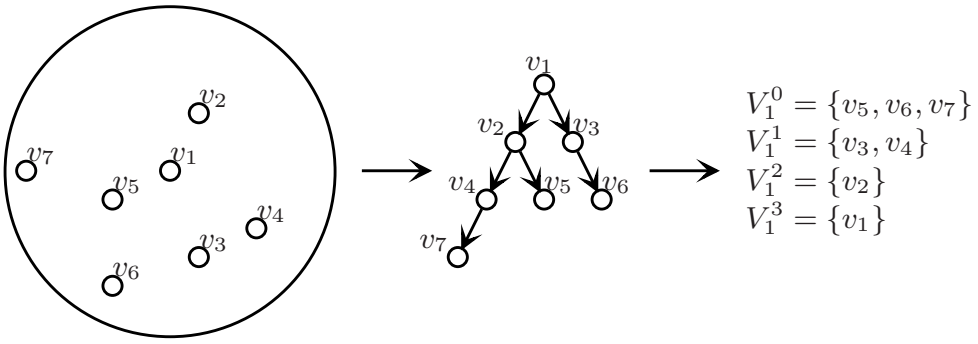


Figure 4.4: Temporal multicast tree construction and node partition.

then T_i^1 denotes the SPT which is deduced by removing all leaf nodes of T_i^0 to V_i^0. By parity of reasoning, T_i^{n+1} is deduced by removing all leaf nodes of T_i^n, and at the same time those leaf nodes are grouped into V_i^n, $V_i^n = V(T_i^n) \setminus V(T_i^{n+1})$, as shown in Figure 4.4.

In the second stage, CTA is applied, and the upload bandwidth of nodes that have conflicts will be re-allocated using CTA. Some edges in T_i^0 or other existing multicast trees which have overlapped intermediate nodes with T_i^0 will be removed in this step. The idea is allocating the upload bandwidth of those conflict nodes to maximize $|T_i^{spt}|$. Different from the Intra-AoI approach which constructs multicast trees for new speakers independently, the to-be-constructed multicast trees with CTA will incur adjustments of existing multicast trees that have overlapping conflict nodes with the newly constructed multicast tree.

There are two processes for CTA: 1). Voting process, which votes potential edges that need to be removed, including edges in the local multicast tree and other existing multicast trees. 2). Allocation process, which allocates upload bandwidth for all nodes and modifies the topologies of multicast trees according to the voting results from the first process. Now I will describe the two processes in detail.

4.4.3 Voting Process

Recall that each node in a multicast tree has only one forwarding node which is responsible for delivering the audio stream to the node. If the edge between the

node and its forwarder is removed, the node and all its descendant nodes will not receive the stream anymore. The *weight* of an edge, $w(e)$, is defined as the number of descendant nodes. For example, as shown in Figure 4.4, $w(e(v_4, v_7)) = 1$ and $w(e(v_1, v_2)) = 4$ since there is only one down-stream node $e(v_4, v_7)$ can support while $e(v_1, v_2)$ can support four descendant nodes. The weight of every edge (e.g., $w(e(j, k))$) in T_i^0 is computed at its end vertex (i.e., v_k), since every node has only one father in T_i^0. Weights of all edges are computed in a *bottom-up* manner, starting from nodes in V_i^0 to V_i^{m-1}, where m is the length of the longest path.

Every node i has a *global weight set*, denoted by W_i^g, which stores the outgoing edges and their weights in existing multicast trees. That is, every entry of W_i^g is a edge-weight pair. For example, let a entry of W_i^g be $W_i^g[k]$, then the edge and its weight are denoted by $W_i^g[k].edge$ and $W_i^g[k].weight$, respectively. It should be noted that the size of W_i^g is determined by its outgoing bandwidth limits, that is, $|W_i^g| \leq u_i^{out}$, where u_i^{out} is the outgoing bandwidth limit of node i. In addition, W_i^g is sorted in a non-increasing order of weights. Every node j in a newly constructed SPT (e.g., T_i^{tmp}) has a temporal *local weight set* (denoted by W_j^l) which stores the same type of information as W_j^g does, and it would be merged into W_j^g afterwards. W_j^g is initialized when the node join the system, and W_j^l is initialized during the post-order tree traversal in the temporal multicast tree construction.

The voting process starts from the leaf node set V_i^0, as shown in Algorithm 5 (lines 2-3). In non-leaf node sets, nodes without conflicts will be processed first. If a non-leaf node v_j has no conflicts (Algorithm 5, lines 6-7), it will calculate w_j ($w_j = w(e(f(j), n_j))$) and propagate it to its forwarder' local weight set, i.e., $W_{f(j)}^l$, where $f(j)$ denotes the father node of j. For a non-leaf node v_j with conflicts, all the remaining upload units of v_j will be allocated to edges in T_i^0, as shown in line 10 of Algorithm 5. Furthermore, some upload units of node j to tree T_i^0, which are indicated in B_j while being occupied by other SPTs, will be re-allocated as shown in line 12 of Algorithm 5. Note that B_j is defined as the *backup unit set* of v_j, and will be explained later. If all upload units of the conflict node, v_j, are allocated to nodes in SPTs, w_j is calculated and propagated to $f(j)$. When some remaining local edges have larger weights than edges in W_j^g, upload

Algorithm 5: Voting Process.

```
1  T_i ← T_i^0;
2  foreach j ∈ V_i^0 do
3      W^l_{f(j)}[j].weight ← 1;
4  foreach j ∈ V_i^x,  x = 1, ... do
5      if node j has no conflicts then
6          w_j ← Σ_{k∈W_j^l} W_j^l[k].weight;
7          W^l_{f(j)}[j].weight ← w_j;
8      else
9          Let a be the number of available upload units of node j;
10         w_j ← a > 0? Σ_{k=0}^{a-1} W_j^l[k].weight : 0;
11         if B_j ≠ ∅ then
12             b ←UnitReallocate(B_j, w_j, a, |W_j^l|);
13         if a ≠ u_j then
14             UnitSchedule(w_j, W_j^l, W_j^g);
15     W^l_{f(j)}[j].weight ← w_j + 1;
16     Update(T_i);
```

unit scheduling algorithm, as shown in line 14 of Algorithm 5, will be applied to calculate w_j and propagate it to the forwarder's local weight set, $W^l_{f(j)}$. Finally, T_i is updated (Algorithm 5, line 16) by removing the remaining edges in W^l_j and their down-stream edges and nodes.

Now let's look at how CTA allocates upload units of conflict nodes cross multicast trees, as shown in line 14 of Algorithm 5. Let v_s be the speaker of the tree T_s, which has a overlapped conflict node, v_j with tree T^i (the speaker is node i). Let v_z refers to a descendant node of v_j both in T_i^0 and T_s^{spt} (subtree of T_s), and v_z is a conflict node in T_i^0. If v_z does not exist, or n_z is a leaf node in T_s^{spt}, the upload unit allocated for the edges in T_s will be re-allocated to the edges in T_i (Algorithm 6, line 6) since the re-allocation may bring in more receivers that have minimum latency. As a result, a tuple $(v_j, e(W_j^g[k].edge))$ will be inserted into E, which is defined as the set of *to-be-removed* global edges set. E will be

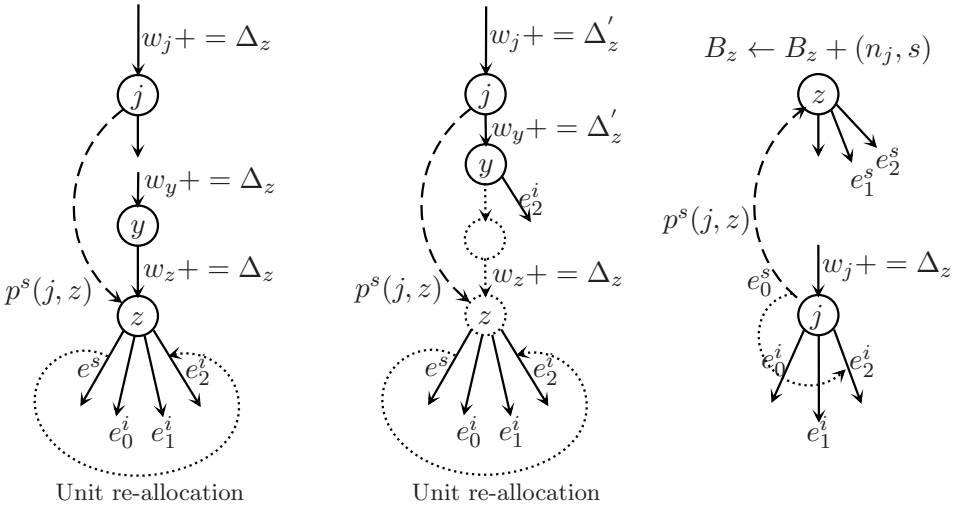
Algorithm 6: UnitSchedule(w_j, W_j^l, W_j^g)

```
1  k ← |W_j^g| − 1, x ← 0, E ← ∅;
2  while x < |W_j^l| do
3      if W_j^g[k].weight < W_j^l[x].weight then
4          Let v_s be the speaker of tree T_s that the edge W_j^g[k] belongs to;
5          Let v_z be a descendant node of node j in tree T_i that has conflict;
6          E ← (v_j, W_j^g[k].edge);
7          if v_z ≠ ∅ and v_z ∉ Leaf(T_s^spt) then
8              if v_z has been processed in the voting process then
9                  if v_z ∈ T_i^spt then
10                     Propagate Δ_z from v_z to v_j;
11                 else
12                     Propagate Δ_z from v_z to v_y;
13                     Propagate Δ'_z from v_y to v_j;
14             else
15                 w_j ← w_j + Δ_z;
16                 B_z ← B_z + {(v_j, s)};
17         else
```

used to adjust other multicast trees in the the allocating process.

If v_z is a forwarder in T_s^{spt}, the occupied upload units at v_z for T_s^{spt} may be re-allocated for T_0^i. There are three possible cases, as shown in Algorithm 6 and Figure 4.5, needed to be considered: 1). v_z has been processed in the voting process, and it has not been removed from T_i. 2). v_z has been processed in the voting process, but it has been removed from T_i. 3). v_z has not been processed in the voting process yet.

As illustrated in Figure 4.5, two conflict nodes, v_j and v_z, both in T_i^0 and T_s^{spt}. Recall that v_j is the node we are processing, and v_z is a descendant of v_j in both trees (T_i^0 and T_s^{spt}). Let $p^s(j,z)$ denote the path from v_j to v_z in T_s^{spt}, and similarly, let $p^i(j,z)$ denote the path from v_j to v_z in T_i^0.



(a). Case 1 (b). Case 2 (c). Case 3

Figure 4.5: Three cases of units scheduling at two conflict nodes, v_j and v_z.

For Case 1 (line 10 in Algorithm 6), the weight of edges in the path $p^i(j,z)$ will be updated as the follows: First, the upload units occupied by edges in T_s^{spt} starting from v_z (e^s) will be re-allocated to edges in T_i^0 starting from v_z (e_2^i). Then, propagate the increased weight, Δ_z ($\Delta_z = W_z^l[e_2^i].weight$ in the example), from v_z to v_j along the path $p^i(j,z)$. It is worth noting that the example only show the case when there is only one unit is re-allocated. It can be easy to extend to multiple units re-allocation.

For Case 2 (Algorithm 6, lines 12-13), the increased weight Δ_z ($\Delta_z = W_z^l[e_2^i].weight$ in the example) will be propagated from n_z along the path $p^i(z,j)$ until v_y is reached, which is the first node in T_i when traversing from v_z to v_j. The weight of edges between v_z and v_y will be increased by Δ_z (Algorithm 6, line 12), and the weight of edges between v_y and v_j will be increased by $\Delta_z^{'}$ (Algorithm 6, line 13). As shown in Figure 4.5 (b), $\Delta_z^{'}$ in this example is computed as the follows:

$$\Delta_z^{'} = \begin{cases} W_z^l[e_2^i].weight - W_y^l[e_3^i].weight & \text{if } W_z^i[e_2^i].weight > W_y^i[e_3^i].weight \\ 0 & \text{if otherwise} \end{cases}$$

For Case 3, where v_z has not been processed yet . w_j is increased by Δ_z,

$\Delta_z = W_z^l[e_2^i].weight$. A tuple (v_j, s) will be inserted into the backup unit set of v_z, denoted as B_z. (v_j, s) in B_z means that once v_z is processed, it can re-allocate units at v_z occupied by T_s^{spt} if v_j is not removed.

The time complexity of unit re-allocation for case 1 and case 2 is $O(|T_i^0)|)$. The time complexity for case 3 is $O(1)$. Therefore the voting process is a polynomial algorithm with complexity $O(|T_i^0)|^2)$.

4.4.4 Allocation Process

After voting process is completed, a *top-down* allocation process is applied to allocate units at all nodes in T^i and adjust other SPTs according to E generated from the voting process. For every node, v_j ($v_j \in T_i$), the global weight set, W_j^g will be updated: $W_j^g \leftarrow W_j^g + W_j^l$. For the to-be-removed edge set, E is not empty, those edges in other SPTs may be removed. For example, assume $(v_j, W_j^g[x]) \in E$. If $v_j \in T_i$ then edge $W_j^g[k].edge$ and all its down-stream edges in its SPT will be removed.

When all nodes' local weight set is merged to their global weight set, all orphan nodes (those nodes removed in the voting process) will be attached to corresponding SPTs. It is worth to note that orphan nodes only attach to nodes who have minimum latency.

4.5 Conclusion

In this chapter, I proposed a tree-based approach (i.e., Intra-AoI approach) which aims to achieve a high system reachability, while at the same time, achieves a relatively low end-to-end latency.

There are two stages to construct a dissemination tree using the Intra-AoI approach. Firstly, a minimum latency tree (MLT, i.e., a combination of shortest path trees) is constructed at the speaker without considering the bandwidth limits of the neighbors of the speaker. At the second stage, the speaker will adjust the minimum latency tree according to the limited bandwidth of the neighbors, aiming to achieve the maximum system reachability.

Cross-tree adjustment (CTA) and audio mixing can be applied to the Intra-AoI approach to further increase the system reachability by optimizing the bandwidth allocation of the conflicting node incurred by multiple speakers.

Although the Intra-AoI tree approach is straightforward, the performance of this approach, in terms of system reachability and the end-to-end latency cannot be desirable in practical. The reason is two-folded. Firstly, since the speakers can only utilize the bandwidth of their neighbors, a high system reachability cannot be achieved, especially when the number of the speakers or the neighbors is large, as shown in the simulation results presented in Chapter 6. Secondly, the length of the audio delivery path in a dissemination tree could be very large as the number of neighbors increases or the bandwidth of neighbor is low, thus the end-to-end latency can be significantly affected.

To overcome the shortcomings of the Intra-AoI approach, I propose two game-theoretic approaches in the next chapter to achieve a near-optimal system reachability while at the same time, achieve a similar end-to-end latency as the client-server scheme, respecting churn, bandwidth limits of users, and avatar mobility.

5

Game-Theoretic Approaches for P2P-based 3D Audio Streaming

In Chapter 4, I described the Intra-AoI approach which creates a minimum latency tree (MLT) for each node (i.e., an end user) when it starts to speak. The root of each MLT, i.e., the speaker, can only utilize the upload bandwidth of users within its AoI. Note that not all the speakers can send their audio streams directly to all the receivers (i.e., via unicast) due to the limited bandwidth of end users. Compared to the shortest path tree (SPT) approach, the Intra-AoI approach (e.g., [21] [22]) can improve the system reachability by utilizing the bandwidth of the receivers within the AoI, but it results in a larger end-to-end latency. Moreover, the collective upload bandwidth of all the nodes cannot be utilized in a system-wide manner, thus the system reachability that the Intra-AoI approach can achieve is suboptimal, especially when the number of speakers and receivers scales up.

Douceur *et al.* [41] presented a centralized approximation algorithm which builds a depth-2 multicast tree for every user. This solution can utilize the network resources in a system-wide manner, but it only addresses the static case where the number of users and their neighbors are fixed and are known a priori. Hence, system dynamics will impose frequent re-calculation operations on the server which requires extensive computational resources.

To overcome these limitations I introduce the novel design of game theoretic approaches for 3D audio streaming in NVEs, which can achieve the following

important contributions:

1. **Achieving near-optimal system reachability.** By introducing the concept of location-unrestricted *helper* nodes to relay audio streams, I am able to prove that the expected system reachability is maximized when the loads of helpers are balanced proportionally to their upload capacities. I then design game-theoretic load balancing algorithms that implement this concept.

2. **Satisfying tight latency constraints.** Note that low end-to-end delay is a requirement in interactive streaming applications. With the uniform game setting, the length of every audio delivery path between sender and receiver is bounded by the helper capacity and the number of neighbors. By tuning the helper capacity according to the number of neighbors, the tight latency constraints can be satisfied. With the weighted game setting, the length of every audio delivery path is at most 2 hops. Thus it can achieve a latency performance similar to client-server solutions.

3. **Adapting to system dynamics.** System dynamics are a significant challenge in P2P systems. The game theoretic solutions moderate the impact of peer joining and leaving by utilizing two or more helper nodes. In addition, the helpers of each peer are not restricted to be located within the AoI of the peer, the impact of avatar mobility on the performance is mitigated and the connections between peers and their helpers can last longer (resulting in less maintenance overheads). Moreover, I prove that the convergence rates of the proposed load balancing algorithms in the two game settings are achieved within a rapid $O(\log \log n)$ time intervals given a population of n peers, thus the game theoretic approaches result in very scalable solutions.

The key idea of the game-theoretic approaches is to maximize the system reachability by balancing the loads of helpers proportionally to their upload capacities. A number of algorithms have been proposed for dynamic load balancing, which is modeled as a congestion game with selfish and non-cooperative users. A steady state in this context is a Nash equilibrium, in which no user is willing to change her strategy. Since the idea of using a potential function to measure the closeness to a Nash equilibrium was introduced by Even-Dar *et al.* [64], a

number of protocols (e.g., [47, 48]) have been proposed to balance the loads in a weakly distributed manner, where the migration events take place one at a time, and costs on users are updated immediately. More recently, concurrent protocols (e.g., [49–51]) for load balancing have been proposed to improve the convergence time by allowing users to change their strategies concurrently.

All the protocols above only address load balancing, not considering the impact of heterogeneous user capacity[1] on the performance of the system. In this thesis, I adopt the concept of these load balancing techniques by applying it to the problem of interactive many-to-many streaming in dynamic heterogeneous environments.

In this chapter, I propose two different game-theoretic approaches, based on unweighted congestion game (UCG) and weighted congestion game (WCG), respectively. In the unweighted congestion game formulation, an abstraction layer is constructed to deliver audio streams for receivers. The abstraction layer consists of helpers, each of which has the same capacity and is decomposed from a peer. In other words, a peer can be regarded as to multiple helpers, each of which has the same upload capacity. In the weighted congestion game formulation, however, each peer is regarded as to a single helper. Before I describe the proposed approaches, the concepts of congestion games are described in Section 5.1.

5.1 Preliminaries of Congestion Games

The classical congestion games have been investigated for many years. A congestion game, denoted by G, can be defined as a tuple $(N, M, (A_i)_{i\in N}, (f_e)_{e\in M})$, where $N = \{1, ..., n\}$ denotes the set of *players* and $M = \{1, ..., m\}$ denotes a set of *facilities*. Each player $i \in N$ is assigned a finite set of strategies A_i and a cost function f_e is associated with facility $e \in M$.

To play the game, each player i selects a strategy $a_i \subseteq A_i$, where $A_i \subseteq M$ is the strategy set of player i. The strategy profile, denoted by $\mathbf{a} = (a_i)_{i\in N}$, is defined as a vector of strategies selected by all the players. Similarly I use the notation $\mathcal{A} = \times_{i\in N} A_i$ to denote the set of all possible strategy profiles. A

[1] Note that users are served by different ISPs, and choosing different broadband packages, thus users have different network bandwidth.

congestion game is *symmetric* if all the players have the same strategy set, i.e., $\forall i, j \in N, A_i = A_j$; otherwise it is *asymmetric*. A congestion game is *weighted* if each player i is specified a weight w_i. The cost of player i for the strategy profile $\mathbf{a}$ is given by $c_i(\mathbf{a}) = f_{a_i}(\mathbf{a}, w_i)$.

The goal of each player in congestion games is to minimize her own cost without trying to optimize the global situation. That is, all players will try to lower their own cost by changing their strategies individually. A *pure Nash equilibrium* is defined as a steady state in which no players have an incentive to change their strategies.

Definition 1 (Pure Nash equilibrium [65]). *A strategy profile $\mathbf{a} \in \mathcal{A}$ is said to be a pure Nash equilibrium of G if for all players $i \in N$ and each alternate strategy $a'_i \in A_i$,*

$$c_i(a_i, \mathbf{a}_{-i}) \leq c_i(a'_i, \mathbf{a}_{-i}), \tag{5.1}$$

where $\mathbf{a}_{-i} = (a_j)_{j \in N \setminus \{i\}}$ denotes the list of strategies of the strategy profile $\mathbf{a}$ for all players except i.

Congestion games have a fundamental property that a pure Nash equilibrium always exists in a congestion game [66]. To analyze the convergence property of congestion games (i.e., the time from any state to a pure Nash equilibrium), the definition of potential function for congestion games is introduced.

Definition 2 (Potential function [46, 49]). *A function $\Phi : \mathcal{A} \to R$ is a potential for game G if $\forall \mathbf{a} \in \mathcal{A}$, $\forall a_i, a_j \in A_i$,*

$$c_i(a_i, \mathbf{a}_{-i}) \leq c_i(a_j, \mathbf{a}_{-i}) \Rightarrow \Phi(a_i, \mathbf{a}_{-i}) \leq \Phi(a_j, \mathbf{a}_{-i}). \tag{5.2}$$

Therefore, $\mathbf{a}^*$ is a Nash equilibrium if and only if

$$\mathbf{a}^* = \arg\min_{\mathbf{a} \in \mathcal{A}} \Phi(\mathbf{a}). \tag{5.3}$$

An ϵ-Nash equilibrium is an approximate Nash equilibrium, which is defined as a state in which no player can reduce her cost by a multiplicative factor of less than $1 - \epsilon$ by changing her strategy.

5.2 Unweighted Congestion Game Formulation

In the UCG formulation, The abstraction layer is responsible for the audio stream disseminating. The abstraction layer consists of *helpers* which are decomposed from the participating nodes. Every speaker v_i will send her audio stream to the abstraction layer, which then forwards the stream to $N(v_i)$ (i.e., the neighbors of v_i). Each helper in the abstraction layer is assumed to be atomic and can only *serve* one speaker at a time. However, a helper may be *selected* by several users concurrently. As soon as one of those users becomes a speaker, that helper will be *occupied* until the speaker releases it (i.e., if there are multiple speakers selecting the same helper, conflicts will occur such that only one speaker can utilize the bandwidth of the helper.). Afterwards, another user may use it, and so on. Hence, the *load* of a helper is defined as the number of nodes that select it. Note that the node with a high upload capacity which is serving multiple speakers concurrently can be regarded as a collection of atomic helpers, each of which is occupied by one of the speakers. All the helpers have the same capacity, denoted by c ($c \geq 2$), which is determined by the helper decomposition algorithm described in Section 5.2.3.

Fig. 5.1 shows an illustrative example of audio dissemination in the UCG formulation. Node v_0 selects two helpers h_3 and h_5, which are decomposed by v_1 and v_2, respectively. Note that all the helpers are virtual nodes in the abstraction layer, hence the nodes that decompose the helpers will fulfill the tasks of their helpers, i.e., forwarding audio streams. Therefore, once v_0 starts to speak, v_1 and v_2 will forward the stream to $N(v_0)$ on behalf of v_0.

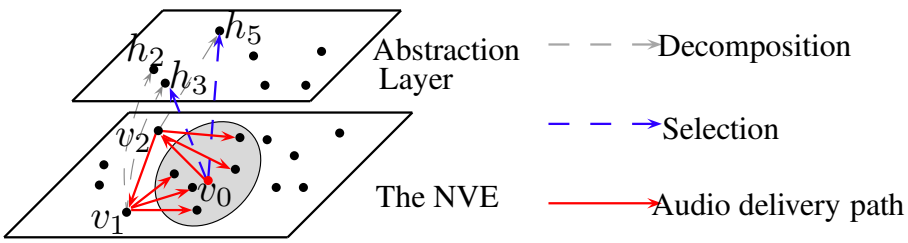


Figure 5.1: Audio dissemination in the UCG formulation.

5.2.1 System Reachability in the UCG Formulation

The number of helpers that a node requires to select from the abstraction layer depends on the number of its neighbors and the helper capacity c. For each node $v_i \in V$, the number of helpers it needs to select is $|h(v_i)| = \lfloor \frac{|N(v_i)|-1}{c-1} \rfloor$. Hence the number of receivers is directly proportional to the number of helpers that are used by the speakers, given that the capacity of each helper is c. The expected system reachability, denoted by $\mathrm{E}(R)$, can be approximated by

$$\mathrm{E}[R] = \frac{\mathrm{E}\left[|h(S)|\right]}{\mathrm{E}\left[\sum_{s_i \in S} |h(s_i)|\right]}, \tag{5.4}$$

where $S \subset V$ is the set of concurrent speakers in the system, and $h(S) = \cup_{s_i \in S} h(s_i)$ is the set of helpers selected by all speakers in S. Therefore, $\mathrm{E}(|h(S)|)$ approximates the expected number of successful receivers, and $\mathrm{E}\left[\sum_{s_i \in S} |h(s_i)|\right]$ approximates the expected number of total receivers.

5.2.2 Schur Concavity of System Reachability

Since speakers rely on their helpers to delivery their audio streams, the loads of non-occupied helpers in the abstraction layer will have significant impact on the system reachability. It is worth noting that multiple nodes can select the same non-occupied helpers, incurring loads on this helper.

Theorem 2. *Let V and H be the set of nodes and the set of non-occupied helpers, respectively. Let $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^{|H|}$ be any two vectors of loads of non-occupied helpers in H, where $\|\mathbf{x}\|_1 = \|\mathbf{y}\|_1$[1]. Define the function $R : \mathbb{Z}^{|H|} \to \mathbb{R}$ as the system reachability. Then we have $\mathbf{x} \preceq \mathbf{y} \Rightarrow R(\mathbf{x}) \geq R(\mathbf{y})$.*

Proof. The concepts of majorization, Schur-convexity that will be used in the proof are provided in the Appendix. Let $|V| = n$ and $|H| = m$. Let $\mathbf{x} = \{x_1, ..., x_m\}$ be the vector of loads of helpers, where x_i is the load of helper h_i. Assume there are k nodes that are speaking in the system, i.e., $|S| = k$. For each helper $h_i \in H$, the probability that h_i is selected by at least one speaker is

[1] $\|\mathbf{x}\|_1 = \sum_{i=1}^{m} |x_i|$ is the L^1-norm of the vector $\mathbf{x} = [x_1, ..., x_m]$.

$\Pr(h_i \in h(S)) = 1 - (1 - \frac{k}{n})^{x_i} = 1 - q^{x_i}$, where $q = (1 - k/n)$. Then the expected number of helpers that will be used by speakers (i.e., nodes in S) is

$$\mathrm{E}\left[|h(S)|\right] = \sum_{h_i \in H} \Pr(h_i \in h(S)) \tag{5.5}$$

$$= m - \sum_{i=1}^{m} q^{x_i}. \tag{5.6}$$

As shown in Corollary 15 in the Appendix, the function $\sum_{i=1}^{m} q^{x_i}$ is Schur-convex on $\mathbb{Z}^m$, hence $\mathrm{E}\left[|h(S)|\right]$ is Schur-concave on $\mathbb{Z}^m$. The expected number of helpers that are required for k speakers is:

$$\mathrm{E}\left[\sum_{s_i \in S} |h(s_i)|\right] = \sum_{h_i \in H} \mathrm{E}\left[|h^{-1}(h_i)|\right] \tag{5.7}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{k} j C_{x_i}^{j} (\frac{k}{n})^{j} q^{x_i - j}$$

$$= \sum_{j=1}^{k} j \left(\frac{k}{n-k}\right)^{j} \sum_{i=1}^{m} C_{x_i}^{j} q^{x_i} \tag{5.8}$$

$$= g(f_1(\mathbf{x}), ..., f_k(\mathbf{x})) \tag{5.9}$$

where $g : \mathbb{Z}^k \to \mathbb{R}$, and $f_1, ..., f_k : \mathbb{Z}^m \to \mathbb{R}$ are Schur-convex functions, which are proved by Corollary 16 in the Appendix. Since g is a non-decreasing function in each argument, $g(f_1(\mathbf{x}), ..., f_k(\mathbf{x}))$ is Schur-convex on $\mathbf{x} \in \mathbb{Z}^m$ [67]. By Proposition G.1 in [67], it suffices to show that the expected system reachability, denoted by $\mathrm{E}[R]$ (see Eq. 5.4), is Schur-concave on $\mathbf{x} \in \mathbb{Z}^m$ given any number of speakers. □

Theorem 2 shows that the system reachability is Schur-concave on the loads of non-occupied helpers, hence the system reachability is maximized if the loads of non-occupied helpers are perfectly balanced. The goal is to balance the loads on the helpers in a distributed manner. I formulate the distributed load balancing problem as a *symmetric congestion game*, where nodes in the game can change their strategies (i.e., helpers) individually and concurrently. I define the potential function of the game as $\Phi(\mathbf{x}) = \sum_{i=1}^{m} (x_i - \overline{x})^2$, where $\mathbf{x} = \{x_1, ..., x_m\}$ is a load vector of the set of non-occupied helpers, and $\overline{x} = \left\lceil \frac{1}{m} \sum_{i=1}^{m} x_i \right\rceil$ is the average load. A *Nash equilibrium*, denoted by $\mathbf{x}_{\text{Nash}}$, is reached when all the nodes have

no incentives to change their strategies, i.e., $\Phi(\mathbf{x}_{\text{Nash}}) = 0$. Theorem 2 implies that the system reachability is maximized when a Nash equilibrium is reached. An ϵ-*Nash equilibrium* is defined as a state in which no node can reduce its cost more than ϵ by changing its strategy.

By Karamata's inequality, the potential function $\Phi(\mathbf{x})$ is Schur-convex on $\mathbb{Z}^m$. Hence the potential function has the property that if node v_i switches from overloaded helpers (those helpers with loads larger than $\overline{x}$) to underloaded helpers (those helpers with loads smaller than $\overline{x}$), $\Phi(\mathbf{x})$ will be decreased accordingly.

5.2.3 Abstraction Layer Construction

Without loss of generality, I assume that all nodes have heterogeneous upload capacities, which can be approximated with high accuracy by a log-normal distribution. Let the random variable $X \sim \text{Log-}\mathcal{N}(\mu, \sigma^2)$ be the upload capacity (measured in upload units) of nodes that is allocated for audio transmissions. The expected number of helpers that a node needs to select is

$$\mathrm{E}\left[|h(v_i)|\right] = \lfloor \frac{\mathrm{E}\left[N(v_i)\right] - 1}{c - 1} \rfloor, \tag{5.10}$$

where $c \in \mathbb{Z}_+$ is the capacity of each helper. Let $Y = X/c \sim \text{Log-}\mathcal{N}(\mu - \ln c, \sigma^2)$, then we have

$$\begin{aligned} \mathrm{E}\left[\frac{X-1}{c} \middle| X > c\right] &= \mathrm{E}\left[Y - \frac{1}{c} \;\middle|\; Y > 1\right] \\ &= \mathrm{E}\left[Z - 1 - \frac{1}{c}\right] \qquad (5.11) \\ &= \frac{\mathrm{E}[X] - 1}{c} - 1. \qquad (5.12) \end{aligned}$$

where $Z = Y + 1$ has a shifted log-normal distribution, hence $\mathrm{E}[Z] = \mathrm{E}[Y] + 1$. Note that the upload units of each node are an integer quantity, and the number of helpers that a node could decompose is also an integer. The expected number

of helpers that can be abstracted from a node is

$$\mathrm{E}\left[\left\lfloor\frac{X-1}{c}\right\rfloor\middle| X>c\right] = \mathrm{E}\left[\left\lfloor Z-1-\frac{1}{c}\right\rfloor\right]$$
$$\approx \frac{\mathrm{E}[X]}{c}-\frac{1}{2} \tag{5.13}$$
$$= \mathrm{E}\left[Z-\frac{3}{2}\right] \tag{5.14}$$
$$= \frac{\mathrm{E}[X]}{c}-\frac{1}{2}, \tag{5.15}$$

where the following approximation is used:

$$\lfloor Z\rfloor = Z-\frac{1}{2}+\frac{1}{\pi}\sum_{k=1}^{\infty}\frac{\sin(2\pi kZ)}{k} \approx Z-\frac{1}{2}. \tag{5.16}$$

By Theorem 2, the expected load of helpers, defined by $\frac{\mathrm{E}[|h(v_i)|]}{\mathrm{E}\left[\lfloor\frac{X-1}{c}\rfloor|X>c\right]}$, should be minimized to maximize the system reachability. That is,

$$\text{Minimize:} \quad \frac{\mathrm{E}\left[|h(v_i)|\right]}{\mathrm{E}\left[\left\lfloor\frac{X-1}{c}\right\rfloor\middle| X>c\right]}$$
$$\text{Subject to:} \quad c\geq 2.$$

Therefore, the optimal helper capacity is

$$c = \max\left(\left[\sqrt{2\mathrm{E}[X]}\right], 2\right), \tag{5.17}$$

where $\left[\sqrt{2\mathrm{E}[X]}\right]$ denotes the integer closest to $\sqrt{2\mathrm{E}[X]}$, and $\mathrm{E}(X) = e^{u+\frac{1}{2}\sigma^2}$. Since the server will maintain the upload capacity information of each node, it will calculate the helper capacity and construct the abstraction layer.

5.2.4 Load Balancing in the UCG Formulation

To rapidly balance the loads of non-occupied helpers, all the nodes should select their helpers concurrently [46] [49]. However, It can be observed that if all nodes behave greedily at each step (i.e., they select those helpers with minimum load deterministically), the system may not converge to a steady state [49]. To guarantee that the system will converge rapidly, some probability rules need to be imposed for all nodes when they decide to change their strategies at each step.

I propose a water-fitting algorithm to enable each node to change its helpers at each step. As shown in Fig. 5.2, all the nodes only consider to switch from *overloaded* helpers (whose loads are larger than $\overline{x}$) to *underloaded* helpers (whose loads are smaller than $\overline{x}$), where $\overline{x} = \lceil \frac{1}{m} \sum_{i=1}^{m} x_i \rceil$ is defined as the average load of all helpers. Moreover, a set of probability distribution functions is assigned to each node, as shown in Eq. 5.18. Let $\mathbf{x} = \{x_1, ..., x_a, x_{a+1}, ..., x_m\}$ be the load vector of non-occupied helpers at the current step, where $x_1 \leq \cdots \leq x_m$, $l = \sum_{i=1}^{a} x_i$ and $\overline{x} \in (x_a, x_{a+1})$. The probability that a node switches from h_k to h_j, denoted by $p_{k,j}$ is

$$p_{k,j} = \begin{cases} (1 - \frac{\overline{x}}{x_k})(\frac{\overline{x} - x_j}{a\overline{x} - l}) & \text{if } 1 \leq j \leq a < k, \\ \frac{\overline{x}}{x_k} & \text{if } a < j = k, \end{cases} \tag{5.18}$$

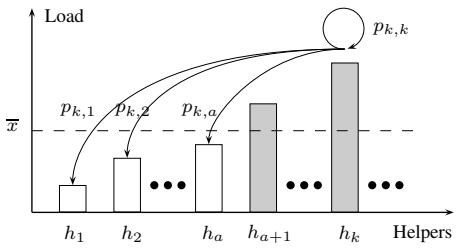


Figure 5.2: Water-fitting illustration: nodes only consider to switch from overloaded helpers in $[h_{a+1}, h_k]$ to underloaded helpers in $[h_1, h_a]$.

5.2.5 Convergence Time Analysis of the UCG Formulation

An important criteria to evaluate distributed algorithms is the convergence time, which is a measure of how fast a system reaches a steady state. In dynamic systems rapid convergence is especially critical, as otherwise churn may prevent the system from closely tracking the optimal solution. Let H be the set of non-occupied helpers, and $|H| = m$. Let $\mathbf{x}_t$ and $\Phi(\mathbf{x}_t)$ be the load vector of H and the corresponding potential function at the step t ($t \geq 0$). Let $\mathbf{x}_t^i = x_i$ be the load of helper $h_i \in H$ at step t. Denoted by $\Phi(\mathbf{x}_{\text{Nash}})$ is the potential function of a Nash equilibrium. Similarly, denoted by $\Phi(\mathbf{x}_{\epsilon\text{-Nash}})$ is the potential function

of an ϵ-Nash equilibrium. It can be observed that $\max(\Phi(\mathbf{x}_{\epsilon\text{-Nash}})) = \frac{\epsilon^2 m}{4}$ (i.e., in case when there are $m/2$ non-occupied helpers with load $\overline{x} + \epsilon/2$ and $m/2$ non-occupied helpers with load $\overline{x} - \epsilon/2$) and $\min(\Phi(\mathbf{x}_{\epsilon\text{-Nash}})) = 0$ (when all the m non-occupied helpers have the same load $\overline{x}$). Therefore, the system is said to be *ϵ-balanced* if $\Phi(\mathbf{x}) \leq \epsilon m$.

The water-fitting algorithm guarantees that the system, from any state, will reach an ϵ-balanced state after a number of steps, denoted by λ. I prove (in Theorem 3) that the expected number of steps, denoted by $\mathrm{E}[\lambda]$, is $O(\log \log n)$, where n is the number of nodes. I also show (in Theorem 6) that the potential function $\Phi(\mathbf{x})$ is a *supermartingale* (i.e., $\mathbb{E}[\Phi(\mathbf{x}(t+1)) \mid \mathbf{x}(t)] \leq \Phi(\mathbf{x}(t))$, for any $t \geq 0$) which guarantees that the water-fitting algorithm will lead the system to a pure Nash equilibrium.

Theorem 3. *Let n and m be the number of nodes and number of non-occupied helpers, respectively. From any state of the system, the expected number of steps required, denoted by $\mathrm{E}[\lambda]$, to reach an ϵ-balanced state is $O(\log \log n)$.*

Proof. To better structure the following proof, I introduce two supporting lemmas (Lemma 4 and Lemma 5). Given any step t, it can be observed from Lemma 4 that the expected load of any non-occupied helper at step $t+1$ is the optimal load $\overline{x}$. Lemma 5 provides an upper bound on the variance of loads of non-occupied helpers at step $t+1$. Consider $\mathrm{E}\left[\Phi(\mathbf{x}_{t+1})\right]$, which is the expected value of the potential function at the next step. Since all nodes update their helpers individually at each step, we have $\mathrm{E}\left[\Phi(\mathbf{x}_{t+1})\right] = \mathrm{E}\left[\sum_{i=1}^{m} \left(\mathbf{x}_{t+1}^i - \overline{x}\right)^2\right]$, and it can be simplified (by Lemmas 4 and 5) to

$$\mathrm{E}\left[\Phi(\mathbf{x}_{t+1})\right] = \sum_{i=1}^{m} \mathrm{Var}\left[\mathbf{x}_{t+1}^i\right] \tag{5.19}$$

$$\leq \left(m\Phi(\mathrm{E}[\mathbf{x}_t])\right)^{\frac{1}{2}}. \tag{5.20}$$

Note that the square-root function is a concave function, hence $\mathrm{E}\left[\Phi(\mathbf{x}_{t+1})\right] \leq (m\mathrm{E}[\Phi(\mathbf{x}_t)])^{\frac{1}{2}}$ (by Jensens inequality). Since $\mathrm{E}\left[\Phi(\mathbf{x}_t)\right] > 0$ for any $t \geq 0$, I define a new function $f(t) = \log(\mathrm{E}\left[\Phi(\mathbf{x}_t)\right])$. Hence $f(t+1) \leq \frac{1}{2}\log m + \frac{1}{2}f(t)$. After λ

steps from t,

$$f(t+\lambda) \leq \left(1 - \frac{1}{2^\lambda}\right) \log m + \frac{f(t)}{2^\lambda} \tag{5.21}$$

$$\leq \log m + \frac{f(t)}{2^\lambda}. \tag{5.22}$$

Therefore, $\mathrm{E}\left[\Phi(\mathbf{x}_{t+\lambda})\right] \leq m\Phi(\mathbf{x}_t)^{2^{-\lambda}}$. The maximum value of $\Phi(\mathbf{x})$ is $\frac{n^4}{2}$, which occurs in case when $N(v_i) = H$ for each node $v_i \in V$, and every node v_i select the same helper $h_j \in H$. That is, $\exists v_j \in V, \forall v_i \in V, h(v_i) = \{h_j\}$. Hence, the maximum value of $f(t)$ is $\log \frac{n^4}{2}$. By E.q. 5.21, we have

$$\begin{aligned} \inf\{\lambda : f(t+\lambda) \leq \log 2m\} &= \inf\{\lambda : f(t) \leq 2^\lambda\} \\ &= \log(4 \log n - 1). \end{aligned} \tag{5.23}$$

After $\lambda = \log(4 \log n - 1)$ steps from any $t \geq 0$, $\mathrm{E}\left[\Phi(\mathbf{x}_{t+\lambda})\right] \leq 2m$. By Markov's inequality, we have that

$$\Pr(\Phi(\mathbf{x}_{t+\lambda}) \geq 4m) \leq \frac{\mathrm{E}\left[\Phi(\mathbf{x}_{t+\lambda})\right]}{4m} = 0.5. \tag{5.24}$$

Let τ be the first time $\Phi(\mathbf{x}_{t+\tau}) \leq 4m$ occurs from any step t. Flip a coin after each of λ steps until $\Phi(\mathbf{x}_{t+\tau}) \leq 4m$:

$$\Pr(\tau = i\lambda) \begin{cases} \geq p & \text{if } i = 1, \\ \leq p(1-p)^{i-1} & \text{if } i \geq 2, \end{cases} \tag{5.25}$$

where $p = \Pr(\Phi(\mathbf{x}_{t+\lambda}) \leq 4m)$, thus $0.5 \leq p < 1$. Therefore,

$$\mathrm{E}[\tau] = \sum_{i=1}^{\infty} i\lambda \Pr(\tau = i\lambda) \tag{5.26}$$

$$\leq \lambda + \frac{1}{4}\lambda \sum_{i=2}^{\infty} (1-p)^{i-2} \tag{5.27}$$

$$= \left(1 + \frac{1}{4p}\right)\lambda \tag{5.28}$$

$$\leq 1.5\lambda. \tag{5.29}$$

By Markov's inequality, $\Pr(\tau \geq 60\lambda) \leq \frac{\mathrm{E}[\tau]}{60\lambda} = 0.025$. Hence, $\Pr(\Phi(\mathbf{x}_{t+60\lambda}) \leq 4m) \geq 0.975$. As a result, from any state, the system can rapidly (within $O(\log \log n)$ steps) converge to an ϵ-balanced state ($\epsilon = 4$) with high probability. □

Lemma 4. $\mathrm{E}\left[\mathbf{x}_{t+1}^i\right] = \overline{x}$, *where* $t \geq 0$ *and* $1 \leq i \leq m$.

Proof. For overloaded helper h_i where $a+1 \leq i \leq m$, we have $\mathrm{E}\left[\mathbf{x}_{t+1}^i\right] = x_i p_{i,i} = \overline{x}$. For underloaded helper h_i where $1 \leq i \leq a$, $\mathrm{E}\left[\mathbf{x}_{t+1}^i\right] = x_i + \sum_{j=a+1}^{m} x_j p_{j,i} = \overline{x}$. □

Lemma 5. $\sum_{i=1}^{m} \mathrm{Var}\left[\mathbf{x}_{t+1}^i\right] \leq (m\Phi(\mathrm{E}[\mathbf{x}_t]))^{\frac{1}{2}}$.

Proof. Let $\mathbf{x}(t) = \{x_1, ..., x_a, x_{a+1}, ..., x_m\}$ where $x_a < \overline{x} < x_{a+1}$. Let $l = \sum_{i=1}^{a} x_i$ and $r = \sum_{i=a+1}^{m} x_i$. For overloaded helper h_i where $a+1 \leq i \leq m$, we have that

$$\sum_{i=a+1}^{m} \mathrm{Var}[\mathbf{x}_{t+1}^i] = \sum_{i=a+1}^{m} x_i p_{i,i}(1 - p_{i,i}) \tag{5.30}$$

$$= (m-a)\overline{x} - \sum_{i=a+1}^{m} \frac{\overline{x}^2}{x_i}. \tag{5.31}$$

For underloaded helper h_i where $1 \leq i \leq a$, we have

$$\sum_{i=1}^{a} \mathrm{Var}[\mathbf{x}_{t+1}^i] = \sum_{i=1}^{a} \sum_{j=a+1}^{m} x_j p_{j,i}(1 - p_{j,i}) \tag{5.32}$$

$$\leq a\overline{x} - l - \eta + (m-a)\overline{x} - \sum_{i=a+1}^{m} \frac{\overline{x}^2}{x_i} \tag{5.33}$$

$$= a\overline{x} - l - \eta + \beta \left((m-a)\overline{x} - \sum_{i=a+1}^{m} \frac{\overline{x}^2}{x_i} \right), \tag{5.34}$$

where $\beta = \sum_{i=1}^{a} (\frac{\overline{x} - x_i}{a\overline{x} - l})^2$, and $\eta = \frac{\sum_{i=1}^{a} (\overline{x} - x_i)^2}{a\overline{x} - l}$. It can be easily proved that, $(m-a)\overline{x} < r$ and $a\overline{x} - l = r - (m-a)\overline{x}$. In addition, $\eta = \beta(a\overline{x} - l)$ and $2(a\overline{x} - l) = \sum_{i=1}^{m} |x_i - \overline{x}|$. By Jensen's inequality, $\sum_{i=a+1}^{m} \frac{1}{x_i} \geq \frac{(m-a)^2}{r}$. As a result,

$$\sum_{i=1}^{m} \mathrm{Var}[\mathbf{x}_{t+1}^i] = r + \beta(m-a)\overline{x} - (\beta+1) \sum_{i=a+1}^{m} \frac{\overline{x}^2}{x_i} - \eta \tag{5.35}$$

$$= (\beta+1) \left[r - \frac{(m-a)^2 \overline{x}^2}{r} \right] - 2\beta(a\overline{x} - l) \tag{5.36}$$

$$= \frac{(\beta+1)(r + (m-a)\overline{x})(a\overline{x} - l)}{r} - 2\beta(a\overline{x} - l) \tag{5.37}$$

$$\leq 2(a\overline{x} - l) = \sum_{i=1}^{m} |x_i - \overline{x}|. \tag{5.38}$$

By Cauchy-Schwarz inequality, we have

$$\sum_{i=1}^{m} |x_i - \overline{x}| \leq (m \sum_{i=1}^{m} (x_i - \overline{x})^2)^{\frac{1}{2}} \tag{5.39}$$

$$= (m\Phi(\mathrm{E}[\mathbf{x}_t]))^{\frac{1}{2}} . \tag{5.40}$$

It suffices to show that $\sum_{i=1}^{m} \mathrm{Var}\left[\mathbf{x}_{t+1}^{i}\right] \leq (m\Phi(\mathrm{E}[\mathbf{x}_t]))^{\frac{1}{2}}$. □

Note that λ will be extremely small in practice (e.g. $\lambda = 7$ when n and θ are set to be the world population.), hence the system can reach an ϵ-balanced state rapidly from any state. Now I will show that the potential function $\Phi(\mathbf{x})$ is a *supermartingale*, which guarantees (by Doob's martingale convergence theorem) that the system is stochastically stable, and the water-fitting algorithm will lead the system to converge to a pure Nash equilibrium, where the loads of non-occupied helpers in the abstraction layer are perfectly balanced.

Theorem 6. *$\Phi(\mathbf{x}_t)$ is a supermartingale, such that for any $t \leq 0$, $\mathrm{E}[\Phi(\mathbf{x}_{t+1}) \mid \mathbf{x}_t] \leq \Phi(\mathbf{x}_t)$.*

Proof. By 5.38, we have

$$\mathrm{E}\left[\Phi(\mathbf{x}_{t+1}) \mid \mathbf{x}_t\right] \leq \sum_{i=1}^{m} |x_i - \overline{x}|. \tag{5.41}$$

Since $x_i, \overline{x} \in \mathbb{Z}$, it suffices to show that $|x_i - \overline{x}| \leq (x_i - \overline{x})^2$ for any $x_i, \overline{x} \in \mathbb{Z}$. Therefore, $\mathrm{E}[\Phi(\mathbf{x}_{t+1}) \mid \mathbf{x}_t] \leq \Phi(\mathbf{x}_t)$ for any $t \leq 0$. □

5.3 Weighted Congestion Game Formulation

Recall that in the UCG formulation, the abstraction layer is constructed to utilize the heterogeneous upload bandwidth of peers. The abstraction layer consists of helpers that are decomposed from peers with heterogeneous upload bandwidth. This formulation is "unweighted" because all the helpers have the same capacity. As a result, peers with high upload capacities can decompose more helpers than peers with low upload capacities do. Intuitively, when the loads of helpers are balanced, the loads of corresponding peers are balanced proportionally to their

upload capacities. Furthermore, the helper capacity in UCG formulation, which has significant impact on the system reachability, is determined by distribution of peers' upload bandwidth. Since the bandwidth distribution of peers is hardly to be known and it is time-varying due to churn and fluctuation, UCG approach may result in sub-optimal system performance and high communication overhead. Moreover, the audio delivery path in UCG formulation may have 3 or more hops, which incurs higher latency comparing to C/S scheme. In this section, the weighted congestion game (WCG) formulation is introduced, which can achieve a near-optimal system reachability and satisfy the tight latency constraints.

In the WCG formulation, every node is treated as a helper. Each node v_i to select a set of helpers, denoted by $h(v_i)$, which consists of $u_i = |h(v_i)|$ nodes selected by v_i. For v_i, a helper $v_j \in h(v_i)$ is *active* if v_j is forwarding the audio stream of v_i and v_j is *inactive* otherwise. It is worth to note that the every audio delivery path is at most 2 hops, such that a helper of a speaker cannot forward the audio stream to another helper of the speaker.

Every node v_i needs to send a *bid price* to each helper. The bid price, denoted by b_i, is defined as the maximum amount of upload units that v_i attempts to utilize at each helper. I assume that every node v_i will send the same bid price to all its helpers, and $b_i = \max\{\lceil \frac{N(v_i)}{u_i} \rceil, 2\}$. The *load* of v_i, denoted by l_i, is defined as the sum of bids at v_i, and $l_i = \sum_{v_j \in h^{-1}(v_i)} b_j$, where $h^{-1}(v_i)$ denotes the set of nodes that select v_i as a helper. Let $r_i = l_i/u_i$ denote the *load ratio* of v_i.

It should be noted that for both formulations, i.e., UCG and WCG, each node can relay audio streams for multiple speakers. The difference of UCG and WCG in terms of streaming forwarding is that: 1). in UCG, each node can be regarded as to multiple helpers, each of which has the same upload capacity and can only serve one speaker at any time, but 2). in WCG, each node can only be a single helper, which can serve multiple speakers simultaneously.

All the terms used in this section are summarized in Table 5.1.

Helper Selection and Update. After joining the system, every node v_i will select u_i helpers uniformly at random, and then invokes a periodic helper update process to choose better helpers every m seconds (the update interval). At each step, node v_i randomly selects u_i nodes, defined as *helper candidates*, in the system. Then v_i will potentially replace the current helpers with the helper

Table 5.1: List of terms used in WCG.

Term	Definition
V	Set of nodes. $\|V\| = n$.
$N(v_i)$	Set of neighbors of $v_i \in V$.
$h(v_i)$	Set of helpers selected by $v_i \in V$.
$h^{-1}(v_i)$	Set of nodes that select $v_i \in V$ as a helper.
u_i	Upload capacity of $v_i \in V$.
b_i	Bid price of $v_i \in V$.
l_i	Load of $v_i \in V$.
r_i	Load ratio of $v_i \in V$, $r_i = \frac{l_i}{u_i}$.
u	Total amount of upload capacities in V, $u = \sum_{i=1}^{n} u_i$
l	Total amount of loads in V, $l = \sum_{i=1}^{n} l_i$
$\overline{r}$	Optimal load ratio, $\overline{r} = \frac{l}{u}$.
m	Helper update interval.
$\mathbf{r}$	Load ratio vector, $\mathbf{r} = [r_1, ..., r_n]$.
$\Phi(\mathbf{r})$	Potential function, $\Phi(\mathbf{r}) = \sum_{i=1}^{n}(r_i - \overline{r})^2$.

candidates using Algorithm 7. It is worth noting that every node will incur a certain load on its helpers when it select these helpers, but it will not attempt to utilize the upload bandwidth of its helpers until it starts to speak.

Audio Dissemination. When node v_i starts to speak, it sends the audio stream to its helpers, each of whom then forwards the audio stream to at most b_i receivers (depth-2 path). A *collision* occurs when two or more speakers attempt to utilize the upload bandwidth of the same helper at the same time, while the helper cannot satisfy all these speakers due to the limited upload capacity.

If some helpers of v_i are unable to forward the audio stream caused by collisions, v_i will send the audio stream directly to its receivers as much as possible (depth-1 path). Therefore, the length of every audio delivery path between a speaker and its receivers is at most 2 hops.

5.3.1 System Reachability in the WCG Formulation

Since every node is a helper in the WCG formulation, the loads of nodes will reflect the system reachability. In this section, I investigate the problem of maximizing the system reachability in the WCG formulation.

Theorem 7. *Let l and u be the sum of the loads and the capacities of nodes in V, respectively. Let $S = \{s_1, ..., s_m\} \subset V$ be any set of speaking nodes. Let R be the system reachability. Then, the expected system reachability R given S, denoted by $\mathrm{E}(R|S)$ is maximized if for any $r_j \in \mathbf{r}$, $r_j = \overline{r} = \frac{l}{u}$.*

Proof. For each speaker $s_i \in S$, let α_i and β_i denote the number of upload units it requires and actually obtains from its helpers, respectively. Clearly we have $\beta_i \leq \alpha_i \leq b_i \cdot u_i$. The *shortfall* of s_i, denoted by $\delta_i = \alpha_i - \beta_i$, results from the collisions with other speaking nodes. Hence the system reachability is

$$R = \frac{\sum_{i=1}^{m} \beta_i}{\sum_{i=1}^{m} \alpha_i} \tag{5.42}$$

$$= 1 - \frac{\sum_{i=1}^{m} \delta_i}{\alpha} \tag{5.43}$$

$$= 1 - \frac{\delta}{\alpha}, \tag{5.44}$$

where $\delta = \sum_{i=1}^{m} \delta_i$ and $\alpha = \sum_{i=1}^{m} \alpha_i$ is the total amount of the shortfall and the required upload units, respectively.

Recall that S can be any subset of V, hence any upload unit of node $v_i \in V$ can be the required upload unit of S. Therefore, the equivalent problem of maximizing the system reachability is minimizing the number of collisions in a "*Balls-into-Bins*" game: throwing α balls (i.e., the total amount of the required upload units) into u *independent* bins (i.e., the total amount of the upload units in V).

Let x_k^i be the bin corresponding to an upload unit of v_i, where $k \in [1, u]$. Its equivalent notation is x_k. The landing probability of bin x_k^i, denoted by p_k^i (equivalent with p_k), can be deduced from the load ratio of v_i:

$$\left(\sum_{i=1}^{n} l_i\right) \cdot p_k^i \cdot u_i = l_i \quad \Rightarrow \quad p_k^i = \frac{r_i}{\sum_{i=1}^{n} l_i} = \frac{r_i}{l}. \tag{5.45}$$

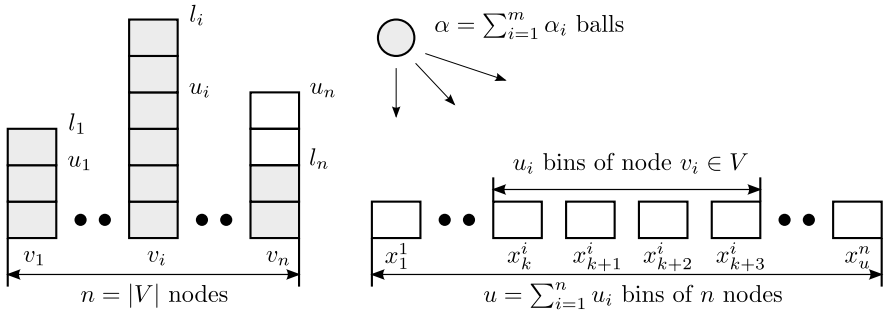


Figure 5.3: The equivalent problem: minimizing the number of collisions in a Balls-into-Bins game.

Let $\mathcal{C}$ denote the event that any two balls collide at a bin, then the probability of event $\mathcal{C}$ is

$$\Pr(\mathcal{C}) = \sum_{k=1}^{u} p_k^2 \tag{5.46}$$

$$= \frac{\sum_{i=1}^{n}(u_i r_i^2)}{l^2} \tag{5.47}$$

$$= \frac{\sum_{i=1}^{n}(\frac{l_i^2}{u_i})}{l^2}. \tag{5.48}$$

Let $\mathcal{C}_{ij}$ denote the event that ball i and ball j collide, and let I_{ij} denote the indicator variable for $\mathcal{C}_{ij}$. That is, $I_{ij} = 1$ if $\mathcal{C}_{ij}$ occurs and $I_{jk} = 0$ otherwise. I define I as the total number of collisions, where $I = \sum_{i \neq j} I_{ij},\ \forall i, j \in [1, \alpha]$. Then the expected number of collisions given S is

$$\mathrm{E}(I|S) = \sum_{i \neq j} \mathrm{E}(I_{ij}) = \sum_{i \neq j} \Pr(\mathcal{C}_{ij}) \tag{5.49}$$

$$= \binom{\alpha}{2} \sum_{k=1}^{u} p_k^2 \tag{5.50}$$

$$= \binom{\alpha}{2} \frac{\sum_{i=1}^{n}(\frac{l_i^2}{u_i})}{l^2}. \tag{5.51}$$

It should be noted that the number of collisions is counted over distinct and unordered pairs (e.g., if three balls fall into the same bin, three collisions are counted). Hence we have $\mathrm{E}(\delta) = \frac{\mathrm{E}(I|S)}{2}$, where $\mathrm{E}(\delta)$ is the expected number of

shortfalls. Then the expected system reachability given S is

$$\mathrm{E}(R|S) = 1 - \frac{\mathrm{E}(\delta)}{\alpha} \tag{5.52}$$

$$= 1 - \frac{(\alpha - 1)\sum_{i=1}^{n}(\frac{l_i^2}{u_i})}{4l^2}, \tag{5.53}$$

where α is a constant given S. Therefore, the problem of maximizing $\mathrm{E}(R|S)$ is formulated as follows:

$$\text{Minimize} \quad f(l_1, ..., l_n) = \frac{\sum_{i=1}^{n}(\frac{l_i^2}{u_i})}{l^2}$$

$$\text{Subject to} \quad \sum_{i=1}^{n} l_i = l.$$

By the method of Lagrange multipliers, $f(l_1, ..., l_n)$ is minimized if $\frac{l_i}{u_i} = \frac{l_j}{u_j}$ for any two nodes $v_i, v_j \in V$, i.e., $\forall v_i, v_j \in V,\ r_i = r_j = \overline{r}$, and this completes the proof. □

Eq. 5.53 shows that the expected system reachability given any set of speaker S, i.e., $\mathrm{E}(R|S)$, is a inner product of two convex functions:

$$\mathrm{E}(R|S) = 1 - \frac{(\alpha - 1)\sum_{i=1}^{n}(\frac{l_i^2}{u_i})}{4l^2} \tag{5.54}$$

$$= f(\mathbf{r}) \times \mathbf{u}^T, \tag{5.55}$$

where $\mathbf{r} = \{r_1, ..., r_n\}$, and $\mathbf{u} = \{u_1, ..., u_n\}$ denote the vector of load ratios of nodes and the vector of upload capacities of nodes, respectively. Thus, $\mathrm{E}(R|S)$ is a Schur-convex function on $\mathbf{r}$.

As a result, to maximize the system reachability, the loads on the nodes must be balanced proportionally to their capacities, and that all the nodes in the system should have the the same load ratio. To achieve a scalable system, the goal is to distribute and balance the loads on the nodes proportionally to their upload capacities in a fully distributed manner. I formulate the distributed load balancing problem as a *symmetric congestion game*, where each player (i.e., each node) in the game can change her strategy (i.e., helpers) individually and concurrently.

More specifically, the problem of maximizing system reachability can be formulated as a symmetric congestion game, denoted by a tuple $\langle V, (A_i), (c_i) \rangle$, which consists of a finite set of players $V = \{v_1, ..., v_n\}$. Throughout the discussion I use the term player and node interchangeably. Each node $v_i \in V$ is assigned a finite set of strategies A_i and a cost function $c_i : A_1 \times \cdots \times A_n \to \mathbb{R}$ that it wishes to minimize. The strategy set of node v_i, denoted by $A_i \in \mathcal{P}(V)$, is an arbitrary collection of subsets of V, and it corresponds to the set of helpers of v_i in our setting. Hence the strategy of each node, denoted by a_i, is a subset of V.

The cost of node v_i is $c_i = \sum_{v_j \in a_i} r_j$, which is the load ratio of the node. Inspired by [46] and [49], I define a *potential function* $\Phi(\mathbf{r}) = \sum_{i=1}^{n} (r_i - \overline{r})^2$ to associate the system reachability with the distribution of load ratios, where $\mathbf{r} = [r_1, ..., r_n]$ is a load ratio vector and $\overline{r} = \frac{l}{u}$ is the optimal load ratio. The potential function has the property that if node v_i switches from helpers with high load ratios to helpers with low load ratios, $\Phi(\mathbf{r})$ will decrease accordingly. A Nash equilibrium, denoted by $\mathbf{r}_{\text{Nash}}$, is reached when all the nodes have no incentives to change their strategies, i.e., $\Phi(\mathbf{r}_{\text{Nash}}) = 0$. Theorem 7 implies that the system reachability is maximized when a Nash equilibrium is reached. An *ϵ-Nash equilibrium* is an approximate Nash equilibrium, and is defined as a state in which no node can reduce its cost by a multiplicative factor of less than $1 - \epsilon$ by changing its strategy.

Every node is selfish and will change its strategy to lower its own cost at each step. This will result in *Nash dynamics* – i,e., $\Phi(\mathbf{r})$ will fluctuate over time. To guarantee that the system converges to a steady state rapidly, a set of probability distributions over A_i (i.e., strategy set) needs to be assigned to each node $v_i \in V$. That is, all the nodes' strategies are nondeterministic and are regulated by a probabilistic rule.

5.3.2 Proportional Load Balancing in the WCG Formulation

Now I present the proportional load balancing (PLB) algorithm in the WCG formulation. The objective of PLB is to distribute the loads of the nodes proportionally to their upload capacities. Intuitively, the nodes with high upload

capacities will have more loads than the nodes with low upload capacities. Hence it would be more appropriate to assign the probability of a node to be contacted proportional to its capacity.

Our solution is to maintain an *identifier space*, which can be managed either by a distributed scheme (e.g., using DHT techniques [68]), or by a centralized scheme (i.e., using a server to store all the identifiers). Each identifier consists of a node ID (i.e., its name or IP address) and a randomly generated hash key. To let a node to be chosen probabilistically proportional to its upload capacity, each node v_i will maintain u_i identifiers, each of which has the same node ID (i.e., node ID of v_i) but a different key. At each step of the helper update process, node v_i will choose an identifier from the identifier space uniformly at random, hence, the probability of node v_j being contacted by v_i, denoted by $\Pr(j)$, is proportional to its upload capacity u_j, i.e., $\Pr(j) = \frac{u_j}{\sum_{k=1}^{n} u_k} = \frac{u_j}{u}$.

Algorithm 7: Proportional load balancing (PLB)

```
1 ∀v_i ∈ V, do the following;
2 foreach v_j ∈ h(v_i) and v_j is inactive do
3     Select v_k at random from the identifier space;
4     Let r_j and r_k be the load ratios of v_j and v_k, respectively;
5     if r_j > r_k then
6         Replace v_j with v_k with probability (1 − r_k/r_j);
```

PLB enables each node to update its helpers as follows (shown in Algorithm 7). At each step, node v_i will contact a helper candidate v_k for each helper v_j by selecting a identifier from the identifier space uniformly at random. At the same time it finds the load and the capacity of v_k (i.e., the helper candidate). Let r_j and r_k be the load ratios of v_j and v_k, respectively. If $r_j > r_k$, then replace v_j with v_k with a probability of $1 - \frac{r_k}{r_j}$. Recall that the probability of the helper candidate v_k being selected from the identifier space is $\Pr(k) = \frac{u_k}{u}$, hence the helper $v_j \in h(v_i)$ will be replaced by any node $v_k \in V$ with probability:

$$p_{jk} = \begin{cases} \frac{u_k}{u}\left(1 - \frac{r_k}{r_j}\right) & \text{if } j \neq k \text{ and } r_j > r_k, \\ 1 - \sum_{j \neq k} p_{jk} & \text{if } j = k. \end{cases} \tag{5.56}$$

It is worth noting that any helpers that are currently active will not be replaced at the helper update step (lines 3-4 in Algorithm 7). The quick convergence of the PLB algorithm is an important property which I will further investigate in Section 5.3.4.

5.3.3 Low Latency Audio Dissemination

When a node starts to speak, it constructs a source-specific dissemination tree on-the-fly, using its dedicated helpers. Each dissemination tree consists of two-types of sub-trees: (a) depth-1 trees and (b) depth-2 trees, both rooted at the speaking node. A depth-2 tree provides the speaker with an indirect relay mechanism, whereby the speaker sends its audio stream to a helper, piggybacking the addresses[1] of some neighbors, and then the helper forwards the audio stream to the neighbors.

As described in Algorithm 8, speaker v_i first establishes direct connections to the helpers within the AoI of the speaker that have available upload units (lines 4-8), and then constructs depth-2 trees by randomly assigning neighbors to the helpers closer to the speaker iteratively (lines 9-18). Finally, the remaining neighbors are connected to v_i directly if v_i still has available upload units, and this forms the depth-1 trees (lines 19-22).

The WCG formualtion significantly reduces the impact of system dynamics (churn and avatar mobility) on the system reachability. Obviously, the depth-1 trees are unaffected by system dynamics. For each depth-2 tree, all the intermediate nodes are the helpers, which forward the audio stream of the speaker to its receivers no matter where these helpers are in the virtual world. As a result, depth-2 trees are unaffected by avatar mobility. When a node leaves, it may impact the speakers who selected it as a helper. However, only a small number of nodes will be affected by the leaving node, because the loads of nodes are balanced proportionally to their capacities. In addition, each node has more than 2 helpers and this would further reduce the impact of churn on the system reachability.

[1]To protect direct identity exposure, they can be replaced by node identifiers.

Algorithm 8: Dissemination tree construction

```
1  Let M be the dissemination tree of v_i;
2  M ← v_i; N ← N(v_i);
3  foreach h ∈ h(v_i) ∩ N(v_i) do
4      u_h ← min(available upload units of h, b_i);
5      if u_h ≥ 1 then
6          M ← add node h and edge (v_i, h);
7          N ← N − {h};
8  H ← build min-heap by delay(v_i, h(v_i));
9  while h ← pop(H) ≠ ∅ do
10     if N = ∅ then
11         return;
12     u_h ← min(available upload units of h, b_i);
13     I ← choose min(u_h, |N|) neighbors randomly;
14     if |I| ≥ 2 then
15         M ← add nodes h, I and edges (v_i, h), (h, I);
16         N ← N − I;
17     H ← H − {h};
18 if N ≠ ∅ then
19     u_i^a ← available upload bandwidth of v_i;
20     I ← choose min(u_i^a, |N|) neighbors randomly;
21     M ← add nodes I and edges (v_i, I);
```

5.3.4 Convergence Time Analysis of the WCG Formulation

An important criteria to evaluate distributed algorithms is the convergence time, which is a measure of how fast a system reaches a steady state. In dynamic systems rapid convergence is especially critical, as otherwise churn may prevent the system from closely tracking the optimal solution. I analyze the upper bound of the convergence time of PLB, where all the nodes behave simultaneously at each step.

Let $\mathbf{r}_t$ and $\mathbf{r}_{t+1}$ be the load ratio vectors of the current step t and the next step $t+1$, respectively. Let $\mathbf{r}_t^i = \frac{l_i}{u_i}$ denote the load ratio of node v_i at time t. Since I consider a highly dynamic and heterogeneous environment where nodes with different upload capacities can join, leave and speak at will, I am especially interested to show how fast PLB can lead the system to converge to an ϵ-Nash equilibrium. It can be observed that $\max \Phi_{\epsilon\text{-Nash}} = \frac{\epsilon^2 n}{4}$, where n is the number of the nodes. Hence an ϵ-Nash equilibrium is reached at time t if $\Phi(\mathbf{r}_t) = O(n)$.

Theorem 8. *Let n be the number of nodes in the system, and let $\Phi(\mathbf{r}_t)$ be the potential function at any step t. Then, the upper bound on the number of steps required to reach an ϵ-Nash equilibrium is $O(\log \log n)$.*

To better structure the following proof of Theorem 8, two supporting lemmas are introduced as follows. Given any step t, it can be observed from Lemma 9 that the expected load ratio of any node at step $t+1$ is the optimal load ratio $\overline{r}$. Lemma 10 provides an upper bound of the variance of load ratios at step $t+1$.

Definition 3. *Let $J := \{j \mid r_j < r_i\}$ be the set of nodes that have a smaller load ratio than v_i. Similarly let $H := \{h \mid r_h > r_i\}$ be the set of nodes that have a larger load ratio than v_i.*

Lemma 9. *Let $\mathbf{r}_t$ be the load ratio vector at time t. Then, the expected load ratio of any node v_i at time $t+1$ is the optimal load vector $\overline{r}$, i.e., $\forall r_i \in \mathbf{r}$, $\mathrm{E}\left[\mathbf{r}_{t+1}^i \mid \mathbf{r}_t\right] = \overline{r}$.*

Proof. For any node $v_k \in V$, the nodes that choose v_k as a helper at time t will

choose v_i as a helper with probability p_{ki} (see Eq. 5.56), hence we have

$$\mathrm{E}[\mathbf{r}_{t+1}^i \mid \mathbf{r}_t] = \frac{1}{u_i}\left(\sum_{k=1}^{n} l_k p_{k,i}\right) \tag{5.57}$$

$$= \frac{1}{u_i}\left(\sum_{h\in H} l_h p_{h,i} + l_i p_{i,i}\right) \tag{5.58}$$

$$= \frac{1}{u_i}\left(\sum_{h\in H} l_h p_{h,i} + l_i - \sum_{j\in J} l_i p_{i,j}\right) \tag{5.59}$$

$$= \frac{1}{u_i}\left(\frac{u_i}{u}\sum_{k\in H\cup J} l_k + l_i - \frac{l_i}{u}\sum_{k\in H\cup J} u_k\right) \tag{5.60}$$

$$= \frac{1}{u_i}\left(\frac{u_i}{u}l - \frac{u_i}{u}\sum_{k\notin H\cup J} l_k + \frac{l_i}{u}\sum_{k\notin H\cup J} u_k\right). \tag{5.61}$$

For each node $k \notin H \cup J$, $r_k = r_i$. Hence $\frac{u_i}{u}\sum_{k\notin H\cup J} l_k - \frac{l_i}{u}\sum_{k\notin H\cup J} u_k = 0$. As a result, $\mathrm{E}\left[\mathbf{r}_{t+1}^i \mid \mathbf{r}_t\right] = \frac{l}{u} = \overline{r}$. □

Lemma 10. $\sum_{i=1}^{n} \mathrm{Var}\left[\mathbf{r}_{t+1}^i \mid \mathbf{r}_t\right] \le (n\Phi(\mathbf{r}_t))^{\frac{1}{2}}$

Proof. For each node v_i, the variance of the load ratio of v_i at time $t+1$ is

$$\mathrm{Var}[\mathbf{r}_{t+1}^i \mid \mathbf{r}_t] = \frac{1}{u_i^2}\left(\sum_{k=1}^{n} l_k p_{k,i}(1-p_{k,i})\right) \tag{5.62}$$

$$\le \frac{1}{u_i^2}\left(\sum_{h\in H} l_h p_{h,i} + l_i(1-p_{i,i})\right) \tag{5.63}$$

$$= \frac{1}{u_i^2}\left(\sum_{l\in L} x_l \Pr(l,i) + \sum_{j\in J} l_i \Pr(i,j)\right) \tag{5.64}$$

$$= \frac{1}{u_i^2}\left(\sum_{h\in H} l_h \frac{u_i}{u}(1-\frac{r_i}{r_h}) + \sum_{j\in J} l_i \frac{u_j}{u}(1-\frac{r_j}{r_i})\right) \tag{5.65}$$

$$= \frac{1}{u_i u}\left(\sum_{h\in H}(l_h - r_i u_h) + \sum_{j\in J}(r_i u_j - l_j)\right) \tag{5.66}$$

$$= \frac{1}{u_i u}\sum_{k=1}^{n}|l_k - r_i u_k| = \frac{1}{u}\sum_{k=1}^{n}\frac{u_k}{u_i}|r_k - r_i|, \tag{5.67}$$

where u_i is the upload capacity of node v_i, and we assume $u_i \geq 2$. Therefore, we sum up the variance of load ratio at each node, hence we have

$$\frac{1}{u}\sum_{k=1}^{n}\frac{u_k}{u_i}|r_k - r_i| \leq \frac{1}{u}\sum_{i=1}^{n}\sum_{k=1}^{n}\frac{u_k}{u_i}|r_k - r_i|. \tag{5.68}$$

For all $r_k, r_i \in \mathbf{r}$, $|r_k - r_i| \leq |r_k - \overline{r}| + |r_i - \overline{r}|$. Using this, we can simplify the above inequality:

$$\frac{1}{u}\sum_{i=1}^{n}\sum_{k=1}^{n}\frac{u_k}{u_i}|r_k - r_i| \leq \frac{1}{u}\sum_{i=1}^{n}\sum_{k=1}^{n}\frac{u_k}{u_i}\left(|r_k - \overline{r}| + |r_i - \overline{r}|\right) \tag{5.69}$$

$$= \sum_{i=1}^{n}\frac{1}{u_i}|r_i - \overline{r}| + \frac{1}{u}\sum_{k=1}^{n}\sum_{i=1}^{n}\frac{u_k}{u_i}|r_k - \overline{r}|. \tag{5.70}$$

Note that the multivariable function $f(u_1, ..., u_n) = \sum_{i=1}^{n}\frac{1}{u_i}$ is a Schur-concave [67] function, and the maximum of f, denoted by $\sup f$, is is achieved if and only if $\forall i, j \in [1, n]$, $u_i = u_j$ (by Karamata's inequality [67]). That is $\sup f = f(\frac{u}{n}, ..., \frac{u}{n})$. Hence we have

$$\frac{1}{u}\sum_{k=1}^{n}\sum_{i=1}^{n}\frac{u_k}{u_i}|r_k - \overline{r}| \leq \frac{n}{u^2}\sum_{k=1}^{n}u_k|r_k - \overline{r}|. \tag{5.71}$$

Since $u_i \geq 2$ for each $v_i \in V$, then we have $\sum_{i=1}^{n}\frac{1}{u_i}|r_i - \overline{r}| \leq \frac{1}{2}|r_i - \overline{r}|$. In addition, $2nu_k \leq u^2$ for any $u_k \geq 2$ and $n \geq 2$. Combine all the inequations above, we have

$$\sum_{i=1}^{n}\mathrm{Var}[\mathbf{r}_{t+1}^{i} \mid \mathbf{r}_t] \leq \sum_{i=1}^{n}|r_i - \overline{r}| \tag{5.72}$$

Then, by Cauchy-Schwarz inequality, we have that

$$\sum_{i=1}^{n}\mathrm{Var}[\mathbf{r}_{t+1}^{i} \mid \mathbf{r}_t] \leq \left(n\sum_{i=1}^{n}|\overline{r} - r_i|^2\right)^{\frac{1}{2}} = (n\Phi(\mathbf{r}_t))^{\frac{1}{2}} \tag{5.73}$$

□

Now I proceed with the proof of Theorem 8.

Proof of Theorem 8. Consider $\mathrm{E}\left[\Phi(\mathbf{r}_{t+1})\right]$, which is the expected value of potential function at step $t+1$:

$$\mathrm{E}\left[\Phi(\mathbf{r}_{t+1})\right] = \mathrm{E}\left[\sum_{i=1}^{n}\left(\mathbf{r}_{t+1}^{i} - \overline{r}\right)^2\right]. \tag{5.74}$$

All the nodes behave independently at each step, hence $\mathbf{r}_{t+1}$ is independent of $\mathbf{r}_t$. Hence we have

$$\mathrm{E}\left[\Phi(\mathbf{r}_{t+1})\right] = \sum_{i=1}^{n}\mathrm{E}\left[\left(\mathbf{r}_{t+1}^{i}\right)^2\right] - 2\overline{r}\sum_{i=1}^{n}\mathrm{E}\left[\mathbf{r}_{t+1}^{i}\right] + n\overline{r}. \tag{5.75}$$

Lemma 9 indicates that the expectation of each node v_i, $\mathrm{E}\left[\mathbf{r}_{t+1}^{i}\right]$, is $\overline{r}$ at each step, hence,

$$\begin{aligned}\mathrm{E}\left[\Phi(\mathbf{r}_{t+1})\right] &= \sum_{i=1}^{n}\mathrm{E}\left[\left(\mathbf{r}_{t+1}^{i}\right)^2\right] - n\overline{r}^2 &(5.76)\\ &= \sum_{i=1}^{n}\mathrm{Var}\left[\mathbf{r}_{t+1}^{i}\right] &(5.77)\\ &\leq \left(n\Phi(\mathbf{r}_t)\right)^{\frac{1}{2}}. &(5.78)\end{aligned}$$

It is easy to observe that the square-root function is a concave function. Therefore, by Jensen's inequality, the expected value of the potential function at step $t+1$ is

$$\begin{aligned}\mathrm{E}\left[\Phi(\mathbf{r}_{t+1})\right] &\leq \mathrm{E}\left[\left(n\Phi(\mathbf{r}_t)\right)^{\frac{1}{2}}\right] &(5.79)\\ &\leq \left(n\mathrm{E}\left[\Phi(\mathbf{r}_t)\right]\right)^{\frac{1}{2}}. &(5.80)\end{aligned}$$

Since $\mathrm{E}\left[\Phi(\mathbf{r}_t)\right] > 0$ for any step $t \geq 0$, I define a function $f(t) = \log(\mathrm{E}\left[\Phi(\mathbf{r}_t)\right])$. Hence we have

$$f(t+1) \leq \frac{\log n}{2} + \frac{f(t)}{2}. \tag{5.81}$$

After $\lambda \in \mathbb{Z}$ steps from any step $t \geq 0$,

$$\begin{aligned}f(t+\lambda) &\leq \left(1 - \frac{1}{2^\lambda}\right)\log n + \frac{1}{2^\lambda}f(t) &(5.82)\\ &\leq \log n + \frac{1}{2^\lambda}f(t). &(5.83)\end{aligned}$$

Therefore, the expected value of the potential function after λ steps from any step $t \geq 0$ is

$$\mathrm{E}\left[\Phi(\mathbf{r}_{t+\lambda})\right] \leq n\mathrm{E}\left[\Phi(\mathbf{r}_t)\right]^{2^{-\lambda}}. \tag{5.84}$$

The upper bound of $\Phi(\mathbf{r})$ is $O(n^6)$, which occurs in case when $N(v_i) = V$ for each node $v_i \in V$, and every node v_i select one specific node $v_j \in V$ as the helper. That is, $\exists v_j \in V, \forall v_i \in V, h(v_i) = \{v_j\}$. Hence, the upper bound of $f(t)$ is $O(\log n)$, i.e., there exists a integer $a \in \mathbb{Z}$ such that $\sup f(t) \leq a \log n$. Therefore, it suffices to show that there exists a $\lambda = \lceil \log(a \log n) \rceil \sim O(\log \log n)$, such that $\frac{f(\lambda)}{2^\lambda} \leq 1$. Hence, for any $t \geq 0$, it suffices to show that

$$\mathrm{E}\left[\Phi(\mathbf{r}_{t+\lambda})\right] \leq 2n, \tag{5.85}$$

where $\lambda = O(\log \log n)$. By Markov's inequality, we have

$$\Pr(\Phi(\mathbf{r}_{t+\lambda}) \geq 4n) \leq \frac{\mathrm{E}\left[\Phi(\mathbf{r}_{t+\lambda})\right]}{4n} = 0.5. \tag{5.86}$$

Let τ be the first time such that $\Phi(\mathbf{r}_{t+\tau}) \leq 4n$ occurs from any step t. Flip a coin after each of λ steps until $\Phi(\mathbf{r}_{t+\tau}) \leq 4n$:

$$\Pr(\tau = i\lambda) \begin{cases} \geq p & \text{if } i = 1, \\ \leq p(1-p)^{i-1} & \text{if } i \geq 2, \end{cases} \tag{5.87}$$

where $p = \Pr(\Phi(\mathbf{r}_{t+\lambda}) \leq 4n)$, thus $0.5 \leq p < 1$. Therefore,

$$\mathrm{E}[\tau] = \sum_{i=1}^{\infty} i\lambda \Pr(\tau = i\lambda) \tag{5.88}$$

$$\leq \lambda + \frac{1}{4}\lambda \sum_{i=2}^{\infty} (1-p)^{i-2} \tag{5.89}$$

$$= \left(1 + \frac{1}{4p}\right)\lambda \leq 1.5\lambda. \tag{5.90}$$

By Markov's inequality, $\Pr(\tau \geq 60\lambda) \leq \frac{\mathrm{E}[\tau]}{60\lambda} = 0.025$. Hence, $P(\Phi(\mathbf{r}_{t+60\lambda}) \leq 4n) \geq 0.975$. As a result, from any state, the system can rapidly (within $O(\log \log n)$ steps) converge to an ϵ-Nash equilibrium, where the load ratios of the nodes are approximately balanced. □

5.4 Conclusion

In this chapter, I proposed two game-theoretic approaches (UCG and WCG) for P2P 3D audio streaming aiming to achieve a near-optimal system reachability, while at the same time, achieve a similar end-to-end latency as the client-server scheme, respecting the churn, bandwidth limits of users, and avatar mobility.

The idea of the two game-theoretic approaches is to efficiently utilize the bandwidth of participating users in a system-wide manner (to achieve a high system reachability), and restrict the length of the audio delivery path to small hops (to achieve a good end-to-end latency performance).

WCG outperforms UCG in terms of system reachability and the end-to-end latency. While UCG achieve a faster convergence due to the uniform setting of the system model. In general WCG would be a better solution for P2P 3D audio streaming considering the heterogenous user bandwidth, churn, and avatar mobility.

5. GAME-THEORETIC APPROACHES FOR P2P-BASED 3D AUDIO STREAMING

6

Evaluation of Approaches for 3D Audio Streaming over NVEs

To evaluate the feasibility and performance of the Intra-AoI approach (with cross-tree adjustment) and the game-theoretic approaches (UCG and WCG formulation), I have implemented a comprehensive simulation program based on C++. The metrics that I use to evaluate the performance of the proposed approaches are system reachability, end-to-end latency, and system convergence and stability.

6.1 Evaluation Settings

I initially deploy $n = 500$ nodes in a 500m $\times$ 500m simulated NVE, where the AoI radius of each node is chosen to be 30 m and 40 m (similar to the normal text chat range of 20 m in Second Life). Considering that any P2P application will almost certainly perform NAT traversal to establish direct connections between nodes, I assume that NAT hole punching techniques [69] are used. For each pair of nodes, the HPSR (hole punching success ratio) is defined as the fraction of times that the NAT hole punching process succeeds, thus reflecting the probability of a successful connection. According to the experiments of [70], I choose three HPSRs (i.e., 0.8, 0.85, 0.9, and 0.95) to evaluate the performance of all the algorithms.

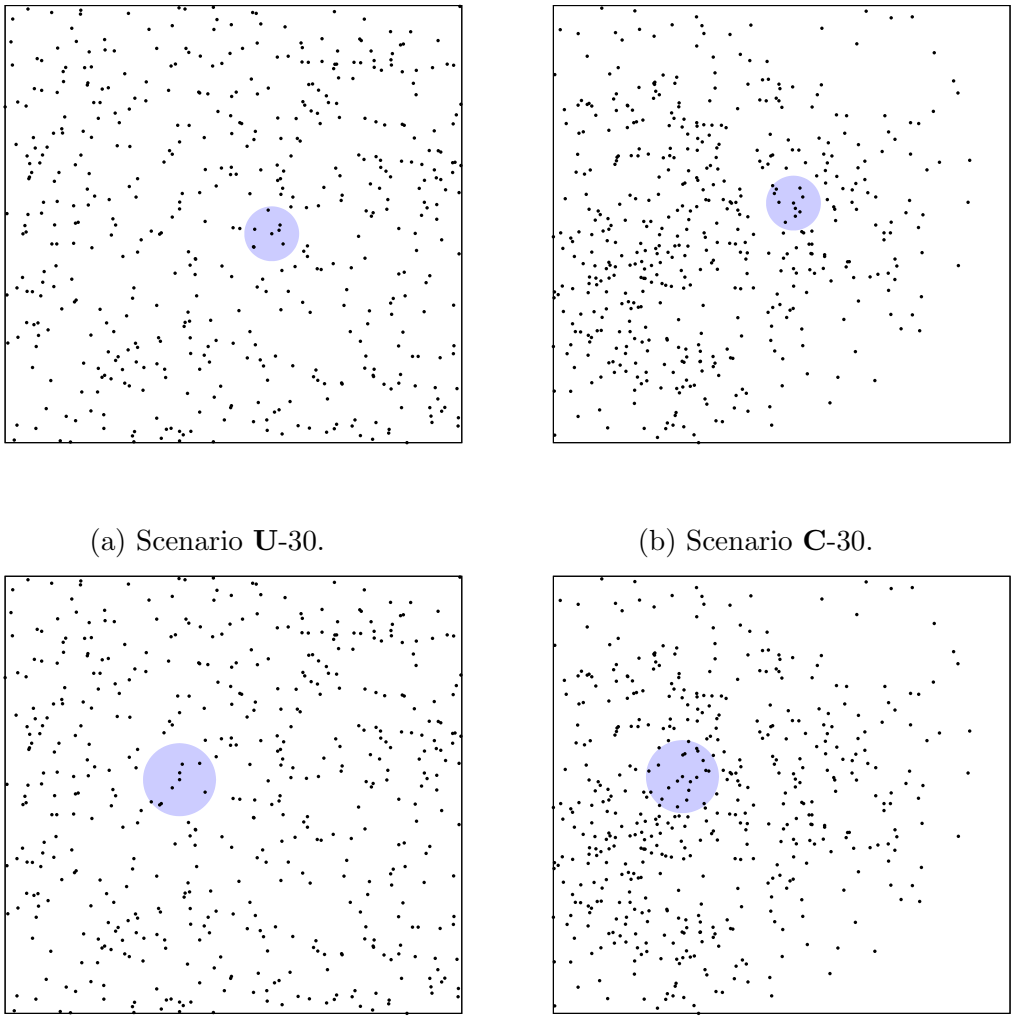
Without loss of generality, two patterns of population densities are considered in the simulation: nodes are uniformly distributed (denoted by **U**) or clustered (denoted by **C**) in the virtual world, respectively. Therefore, I consider four

Table 6.1: List of parameters used in the simulation.

Parameter	Units
Virtual world (m^2)	500^2
Radius of AoI (m)	30, 40
Initial number of nodes	500
Mean of speaking duration (sec)	300
Range of node speed (m/sec)	[0,12]
Fraction of concurrent speakers ($s/n = \|S\|/\|V\|$)	[0, 0.5]
Churn rate (number of nodes *per* min)	[0,30]
Helper update interval (sec)	10, 30

scenarios in the evaluation, as shown in Figure 6.1. The average number of neighbors are shown in Figure 6.2.

Based on a realistic upload capacity dataset collected from dslreports.com [63], I evaluate all the proposed approaches in a heterogeneous environment by assigning an upload capacity randomly selected from the dataset to each node. In addition, only 25% of the upload capacity of each node is dedicated to audio channels, each of which has a 25 kbps delivery rate (e.g., the maximum bandwidth of any audio stream in TeamSpeak 2). The remaining bandwidth is reserved for NVE or game control related traffic. Figure 6.3(b) shows the distribution of the audio-dedicated download and upload units. To simulate avatar mobility, I use the well-known random walk model to evaluate the performance of the proposed algorithms in a worse case because an avatar is unlikely to move randomly in a networked virtual environment. In addition, I assume that the duration of each conversation session conforms to an exponential distribution. A Poisson-based model is used to emulate churn, where the churn rate denotes the number of nodes joining or leaving the system per minute. Table 6.1 shows the parameters I used in the simulations.



(a) Scenario **U**-30.

(b) Scenario **C**-30.

(c) Scenario **U**-40.

(d) Scenario **C**-40.

Figure 6.1: Four scenarios used in the evaluation.

6.2 System Reachability

First, the system reachability of different approaches: SPT, the Intra-AoI approach [21, 22], game-theoretic approach (UCG and WCG) and OPT are evaluated. OPT denotes an optimal solution that achieves the maximum system reachability respecting the limit of the upload capacity of each node. Since it is a NP-hard problem to find OPT in a P2P manner [41], I emulate OPT using

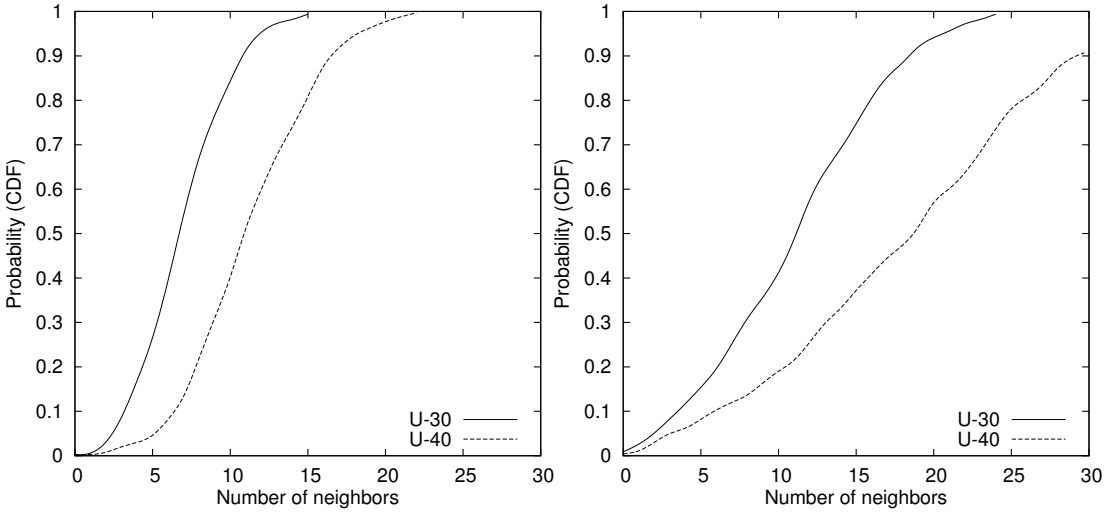


Figure 6.2: Distribution of number of neighbors.

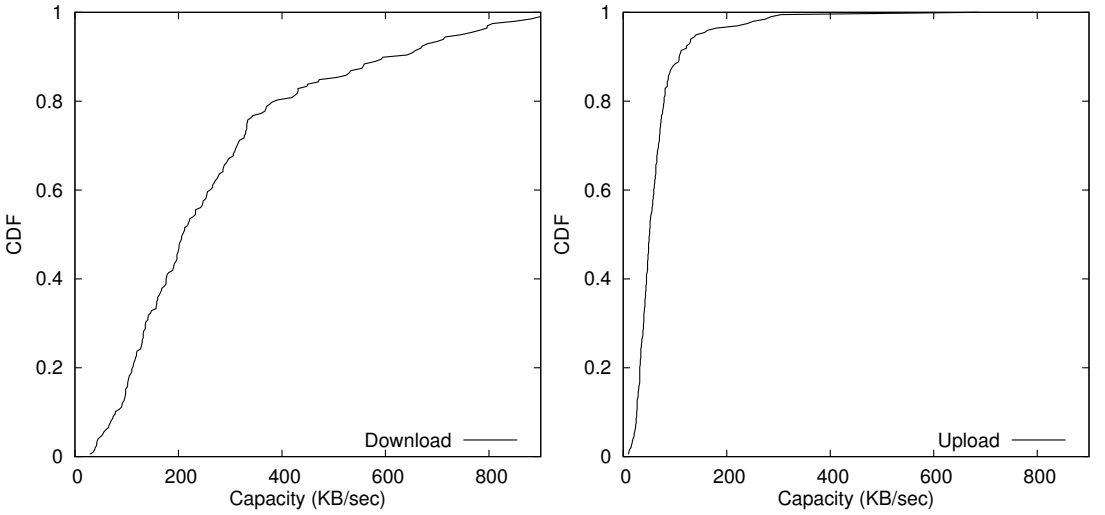


Figure 6.3: Distribution of download and upload bandwidth.

a C/S solution: a server with an upload capacity $\sum_{i=1}^{n} u_i - n$ forwards every audio stream. SPT denotes the solution which construct the multicast tree using the traditional shortest path tree algorithm. Based on the observation that the default keep-alive period is 120 s in BitTorrent, for example, and considering that NVEs are highly dynamic environments, I evaluate two different helper update intervals of 10 s and 30 s (denoted by m), respectively. For the Intra-AoI

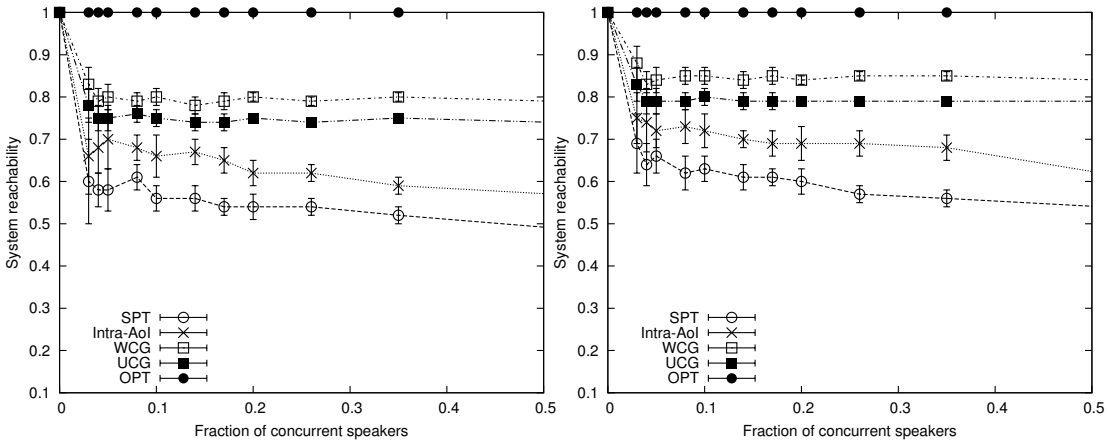
approach, a cross-tree adjustment (CTA) [22] technique is used to improve the system reachability.

To facilitate the presentation, I use the term *spt-sr*, *intra-aoi-sr*, *ucg-sr*, *wcg-sr* and *opt-sr* to denote the system reachability of SPT, the Intra-AoI approach, UCG, WCG and OPT, respectively. In addition, $\frac{s}{n}$ is used to denote the fraction of concurrent speakers, where s is the number of concurrent speakers. The speaking activity is categorized into low ($\frac{s}{n} \in [0, 0.1)$), medium ($\frac{s}{n} \in [0.1, 0.3)$) and high ($\frac{s}{n} \in [0, 3.0.5]$).

Figures 6.4, 6.5, 6.6 and 6.7 shows *spt-sr*, *intra-aoi-sr*, *ucg-sr*, *wcg-sr* and *opt-sr* in the four scenarios (e.g., **U**-30, **U**-40, **C**-30 and **C**-40). The average number of the neighbors of each speaker in the four scenarios is shown in Figure 6.2.

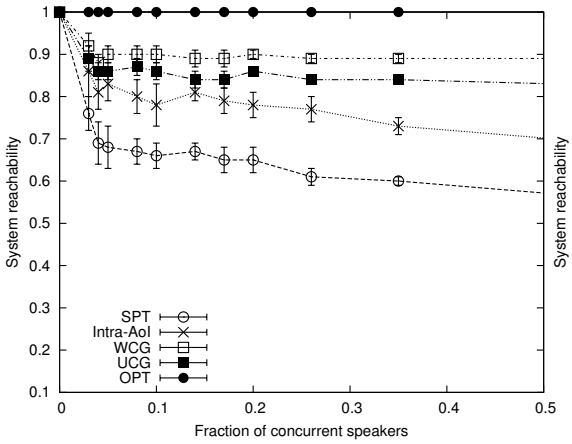
In the scenarios that the positions of avatars are uniformly distributed in the NVE (i.e., **U**-30 and **U**-40), as shown in Figures 6.4 and 6.5, WCG can achieve a near-optimal system reachability (around 95% shown in Figures 6.4(d) and 6.5(d)) when the HPSR is high. If the HPSR is low (0.8), *wcg-sr* can achieve 80%. However, *spt-sr* as well as *intra-aoi-sr* decreases dramatically as HPSR decreases. For example, *intra-aoi-sr* is less than 82% when HPSR is high (0.95), and it decreases to 65% when HPSR is low (0.8). On average, WCG achieves a 15% higher system reachability than the Intra-AoI approach with scenario **U**. This is attributed to the different dissemination structures of WCG and the Intra-AoI approach. Each audio delivery path in WCG is at most 2 hops, while the length of the audio delivery path in Intra-AoI trees is usually larger than 2 hops, especially when the number of neighbors is large. In addition, audio streams can be successfully delivered to the first hop in WCG, since each node will select the helpers with which it can establish connections (using NAT hole punching). This further reduces the impact of HPSR to the system reachability of WCG. Compared to WCG, UCG achieves a lower system reachability. Note that helpers in UCG have the same capacity and cannot serve multiple speakers at the same time, and this decrease the utilization of upload bandwidth of peers.

In the scenarios that the positions of avatars are clustered in the NVE (i.e., **C**-30 and **C**-40), as illustrated in Figures 6.6 and 6.7, *spt-sr* and *intra-aoi-sr* declines dramatically as the value of $\frac{s}{n}$ increases from 0 to 0.2, because the number of neighbors increases greatly in the scenario **C**-30 and **C**-40. We can see that WCG

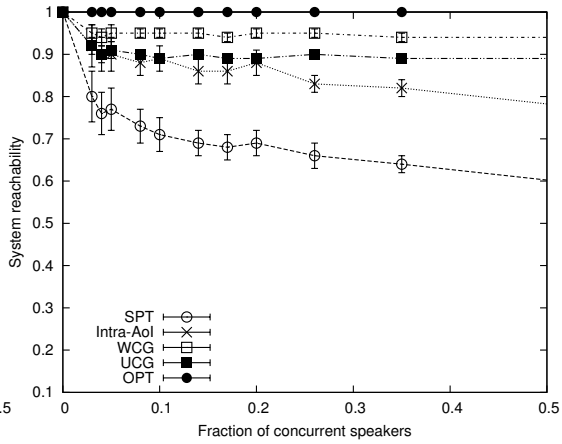


(a) HPSR = 0.8.

(b) HPSR = 0.85.



(c) HPSR = 0.9.

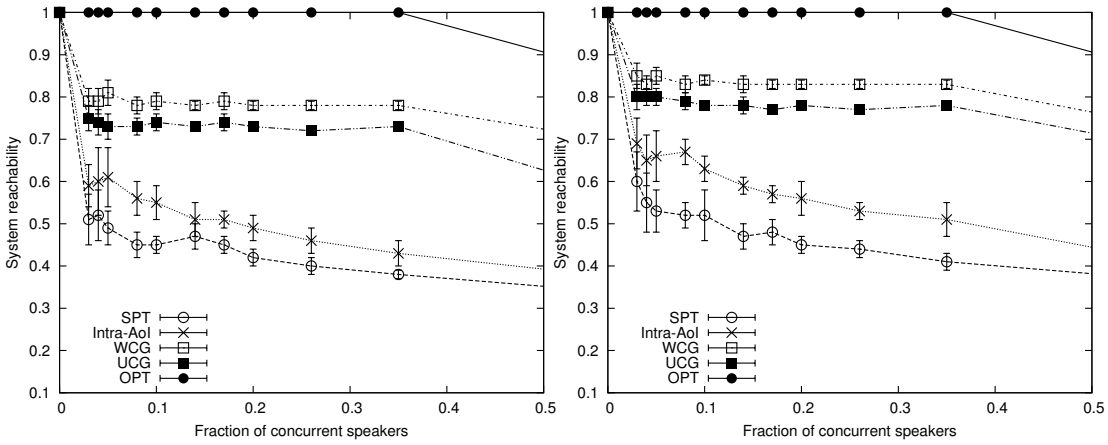


(d) HPSR = 0.95.

Figure 6.4: System reachability with different HPSRs in Scenario **U-30**, where nodes are uniformly distributed in the NVE. The radius of each node is 30 m.

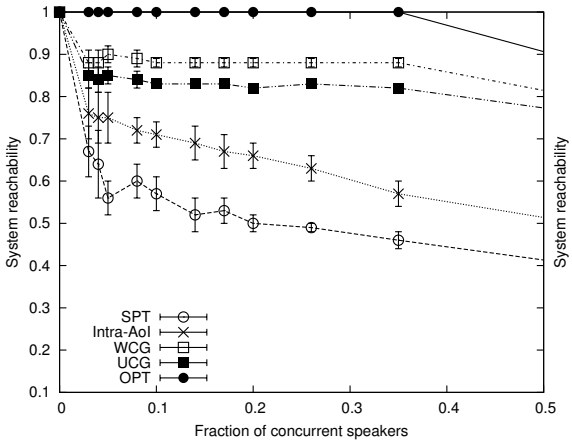
outperforms Intra-AoI approach by 20% on average. Note that the increase of neighbors will increase the length of the audio delivery path in Intra-AoI approach (the upload capacity constraints of nodes remain the same as in scenario **U**). This degrades the system reachability both due to the upload capacity constraints and the HPSR.

It can be observed from Figures 6.4, 6.5, 6.6 and 6.7 that game-theoretic approaches (e.g., WCG and UCG) will outperform Intra-AoI approach and SPT under all conditions, especially when the speaking activity is high and nodes are clustered in the NVE. This is attributed to the different upload bandwidth uti-

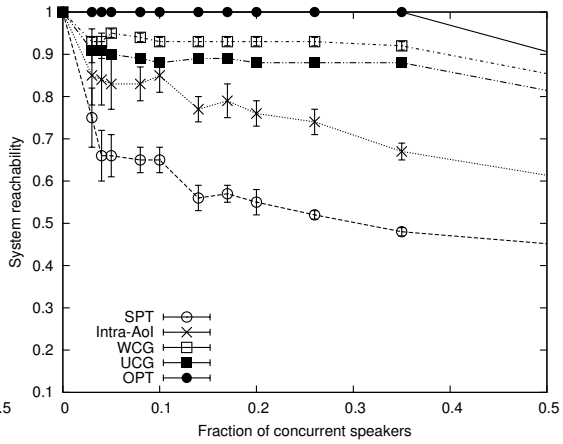


(a) HPSR = 0.8.

(b) HPSR = 0.85.



(c) HPSR = 0.9.

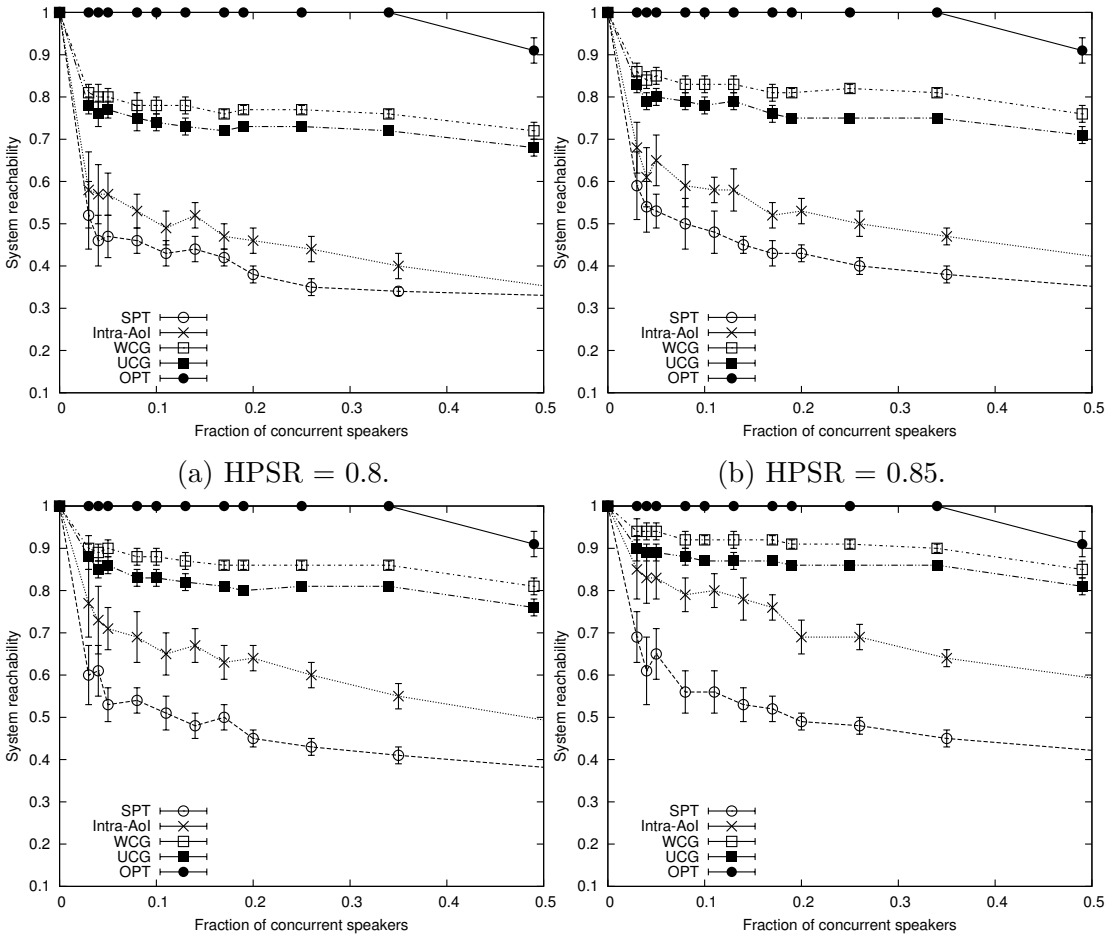


(d) HPSR = 0.95.

Figure 6.5: System reachability with different HPSRs in Scenario **U-40**, where nodes are uniformly distributed in the NVE. The radius of each node is 40 m.

lization schemes of WCG/UCG, Intra-AoI approach and SPT. Specifically, every speaker in SPT cannot efficiently utilize the upload bandwidth of its neighbors (i.e., within the AoI). For MLT, each speaker can only utilize the upload bandwidth of its neighbors. However each speaker with IDM can utilize the upload bandwidth of the nodes in a system-wide manner.

The helper update interval in WCG/UCG has a limited impact on the system reachability. Specifically, if the value of $\frac{s}{n}$ is less than 0.2, the system reachability of IDM with both helper update intervals is close and near-optimal. This is due to the fast convergence of PLB, which can lead the system to converge to an



(a) HPSR = 0.8.

(b) HPSR = 0.85.

(c) HPSR = 0.9.

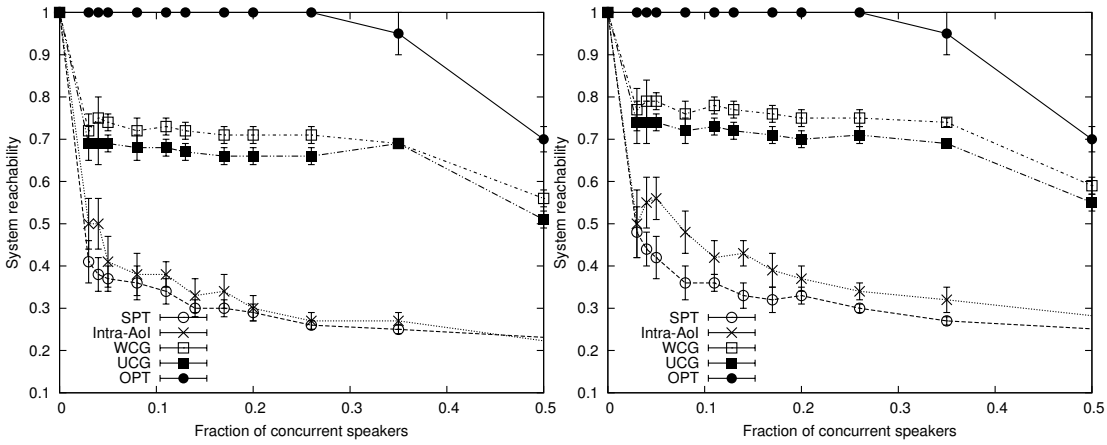
(d) HPSR = 0.95.

Figure 6.6: System reachability with different HPSRs in Scenario **C-30**, where nodes are clustered in the NVE. The radius of each node is 30 m.

approximately balanced state rapidly.

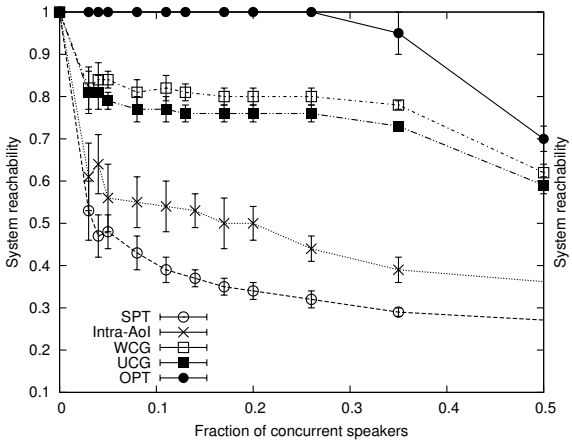
6.3 End-to-end Latency

Next, I present the end-to-end latency performance of the SPT, Intra-AoI, UCG, WCG and C/S schemes. To emulate the C/S scheme, I randomly choose a node in the system as the server which forwards the audio streams sent by every node. The Meridian latency matrix [71] is used to emulate the end-to-end latency between any two nodes in the system.

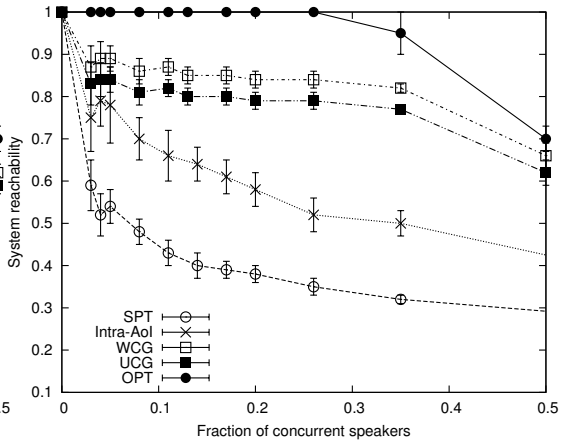


(a) HPSR = 0.8.

(b) HPSR = 0.85.



(c) HPSR = 0.9.

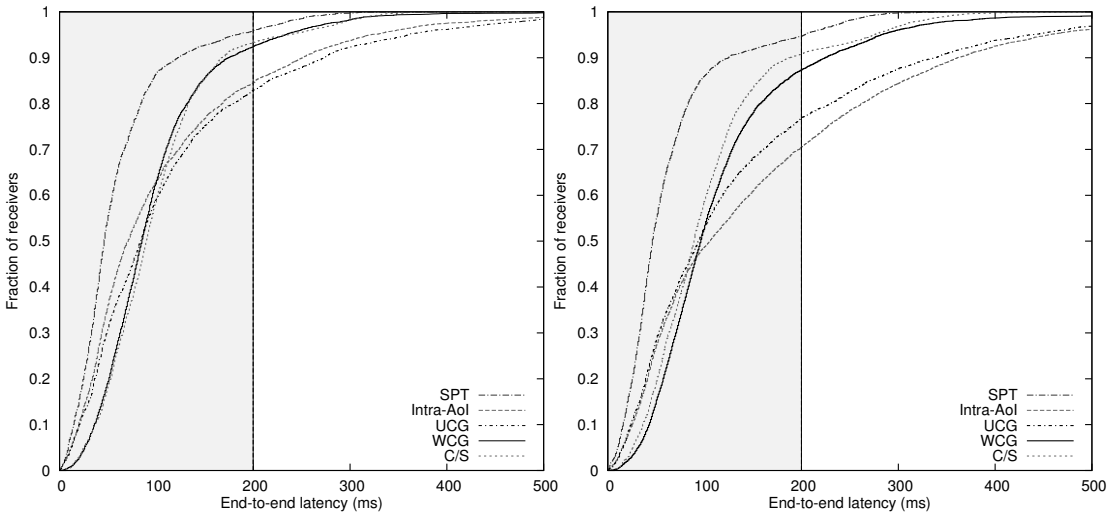


(d) HPSR = 0.95.

Figure 6.7: System reachability with different HPSRs in Scenario **C-40**, where nodes are clustered in the NVE. The radius of each node is 40 m.

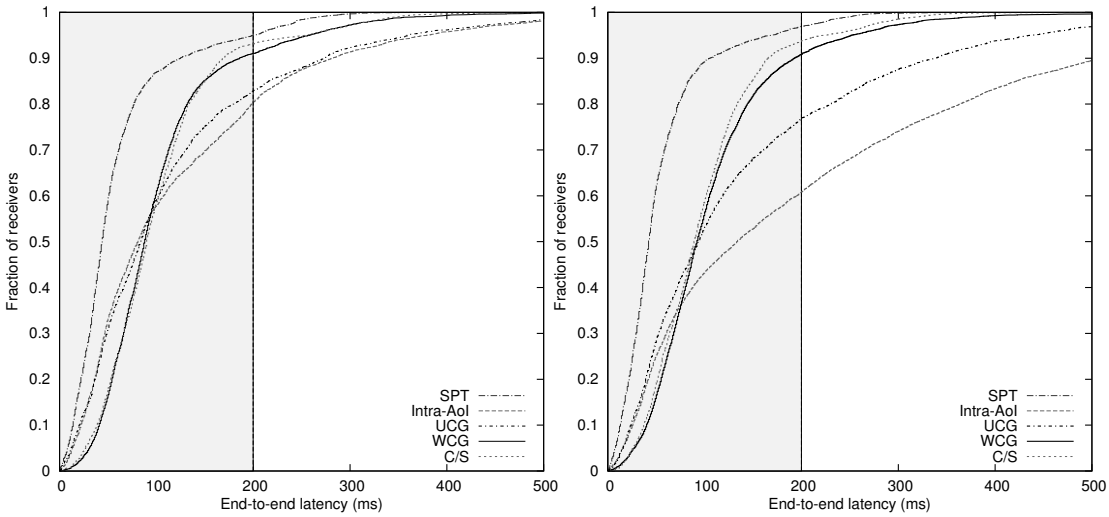
Figures 6.8 and 6.9 illustrate the distribution of end-to-end latencies from the speakers to their receivers with different approaches in the four scenarios. It can be seen that SPT achieves the best latency performance since the multicast trees are constructed based on shortest path trees. Comparing to the Intra-AoI approach, we can see that WCG significantly reduces the end-to-end latency that the receivers experience.

As shown in Figures 6.8(a) and 6.9(a), If the radius of AoI is 30 m, nearly 90% of the receivers with WCG can receive their audio streams within 200 ms, which is the maximum acceptable one-way latency in an interactive audio ap-



(a) Scenario **U**-30. (b) Scenario **U**-40.

Figure 6.8: End-to-end latency performance in the scenario where the positions of avatars are uniformly distributed in the NVE.



(a) Scenario **C**-30. (b) Scenario **C**-40.

Figure 6.9: End-to-end latency performance in the scenario where the positions of avatars are clustered in the NVE.

plication [16]. However, only 80% of receivers with the Intra-AoI approach can satisfy the latency constraint. If the radius of AoI is set to 40 m, the number of neighbors of each speaker will increase, and the latency performance with Intra-AoI approach will degrade significantly, as illustrated in Figure 6.8(b) and 6.9(b). Specifically, less than 70% and 60% of receivers with the Intra-AoI approach can satisfy the latency constraint in scenario **U**-40 and **C**-40, respectively, as shown in Figures 6.8(b) and 6.9(b).

Moreover, the node location distribution in the virtual world has no impact on the end-to-end latencies of receivers with WCG. This is because the length of every audio delivery path with WCG is at most 2 links. Hence the latency performance of WCG is similar or better than the optimal latency performance of the client-server solutions. However, UCG will be affected by the node location distribution, since the length of audio delivery path with UCG depends on the number of neighbors of speakers. As the number of neighbors increases, the length of audio delivery path would increase since helpers with UCG have the same capacity.

6.4 System Convergence and Stability

The final simulation is performed to demonstrate that game-theoretic approaches (i.e., UCG and WCG) can lead the system to converge to a steady state very quickly and that it can adapt to churn.

Figure 6.10(a) shows the worst-case convergence time of load balancing algorithms with UCG and WCG, where all the nodes update their helpers simultaneously at each step. Moreover, all nodes will select the *same node* as their helper initially, i.e., the system starts from the state where $\exists v_i \in V, \forall v_j \in V,\ \{v_i\} = h(v_j)$. We see from Figure 6.10(a) that UCG and WCG can lead the system to converge to a steady state within $O(\log \log n)$ steps regardless of the node position distributions, where n is the number of nodes in the system.

I collected the total number of concurrent users [1] in Second Life during a period of 24 hours, as shown in Figure 6.11.

[1] http://secondlife.com/xmlhttp/secondlife.php

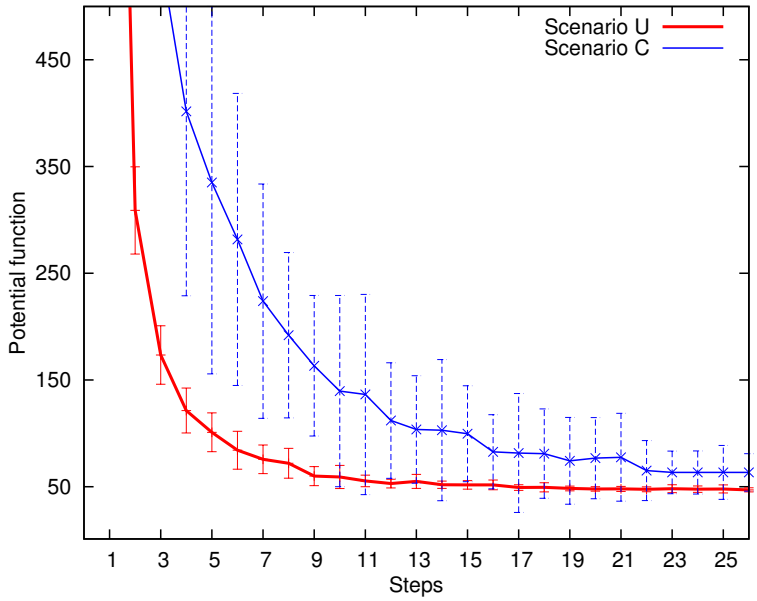


Figure 6.10: Worst-case convergence rate of UCG and WCG.

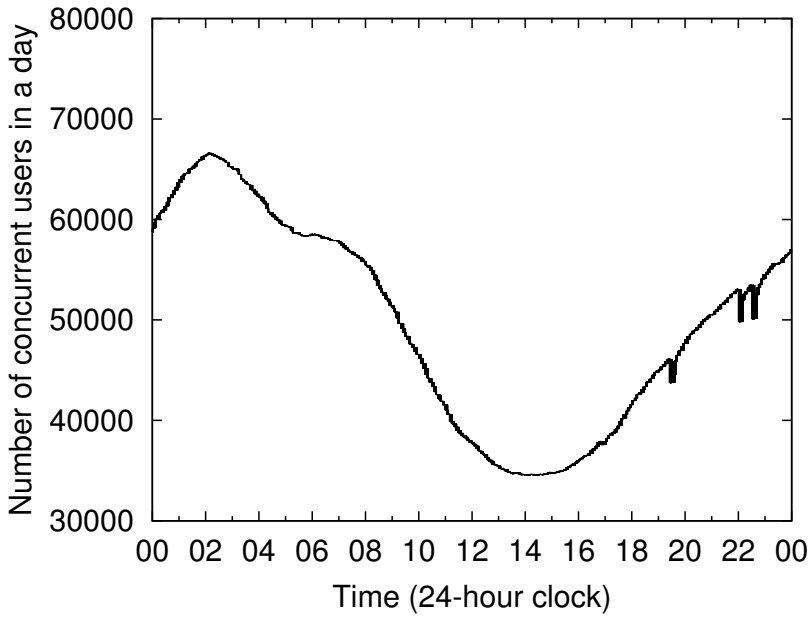


Figure 6.11: Number of online users in Secondlife during a day.

The value of the potential function of UCG and WCG reflects the distribution

of loads on the nodes, and a lower value of the potential function results in a higher system reachability (see Theorems 2 and 7, and Eq.5.55). Figure 6.12 shows the value of the potential function of the system during 24 hours. It can be observed that the value of the potential function varies frequently when churn is high (e.g., during the time intervals 08:00-14:00 and 14:00-00:00). Despite the frequent spikes (resulting from churn) in the potential function, the system always quickly returns to a stable state (i.e., ϵ-Nash equilibrium), which is attributed to the fast convergence rate of UCG and WCG. Therefore, the system performance is stable such that the impact of system dynamics on the reachability is mitigated.

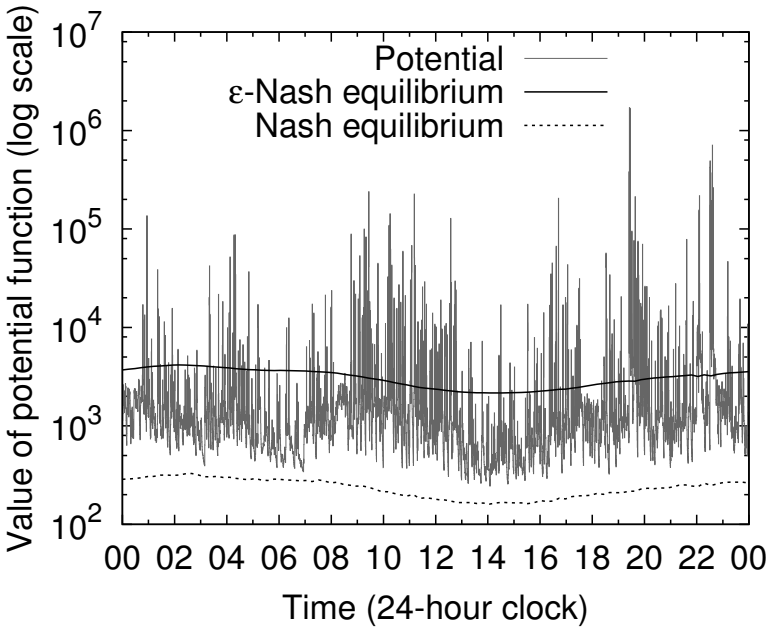


Figure 6.12: Value of potential function over time.

6.5 Overhead

Figure 6.13 depicts the communication overhead incurred by UCG and WCG during a day. The communication overhead are incurred by the helper updating process at nodes, because of their need to establish new connections with newly selected helpers. The communications overhead rises proportionally with the

fraction of nodes that update their helpers. It can observed that, even during periods of high churn (e.g., 08:00-14:00 and 14:00-00:00), only a small portion of nodes (less than 2.5%) will change their helpers. When the churn is low (e.g., 06:00-08:00), less than 1% of nodes will change their helpers. Therefore, I believe the communication overhead of UCG and WCG is reasonable in practice.

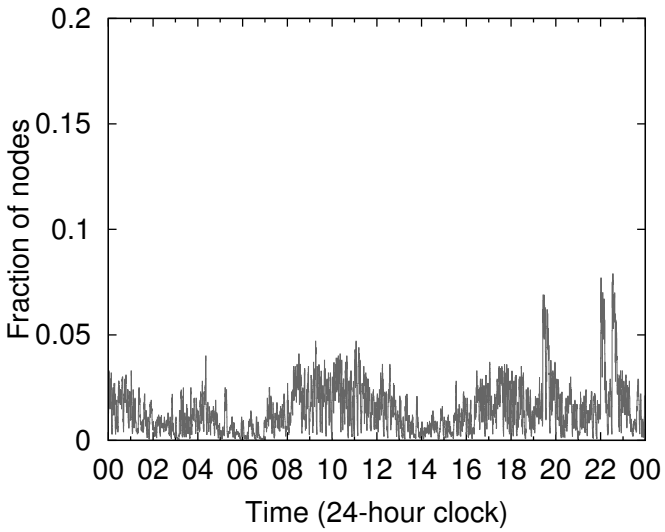


Figure 6.13: Fraction of nodes updating their helpers over time.

To investigate the processing overhead of nodes incurred by UCG and WCG, I compute the time interval between two successive helper update operations, as shown in Figure 6.14. It can be observed that 50% of time intervals between two successive helper update operations are larger than 100 seconds. On average, each node establishes connections with newly selected helpers (i.e., change its helpers) every 420 seconds. As a result, the processing overhead of nodes is relatively low.

6.6 Angular Errors Introduced by Audio Mixing

Audio mixing may result in some spatial information loss because the position information of two or more audio sources are merged and averaged. Figure 6.15

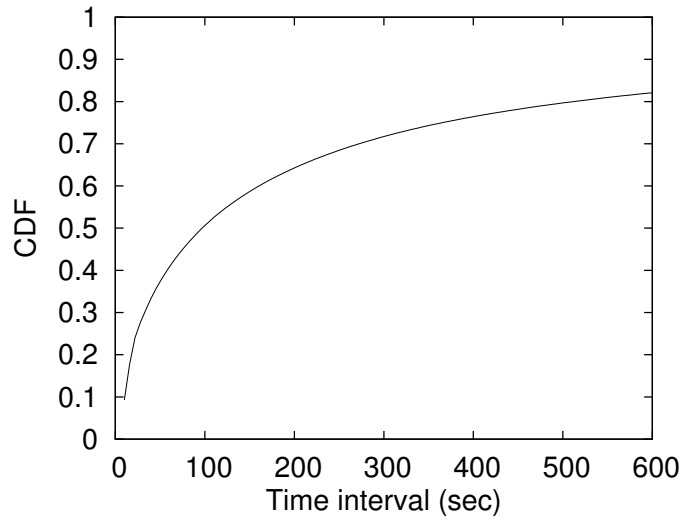
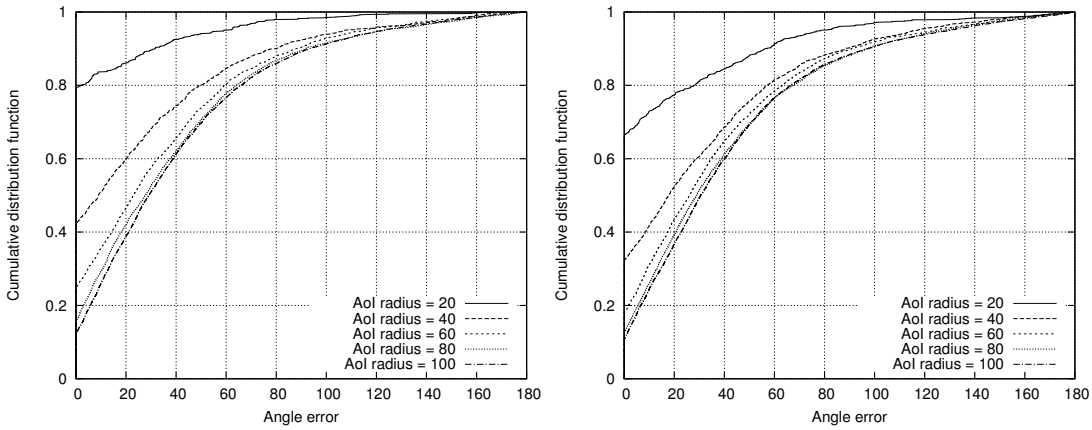


Figure 6.14: Distribution of the time interval between two successive helper updating operations .

depicts the angular error introduced through this merging process. We can observe from Figure 6.15 that the angular errors introduced by audio mixing are significantly affected by avatar mobility, and the number of neighbors (note that a larger AoI radius results in a larger number of neighbors in general). Since the system reachability may be further increased by mixing audio streams on the helpers scarifying users' perception of the 3D audio, the audio mixing techniques is worth further exploring in the future work.



(a) Scenario **U**.

(b) Scenario **C**.

Figure 6.15: CDF of angular error when mixing audio streams with a fixed talking probability of 50% in the two scenarios.

7

Game-Theoretic Approach for Peer-assisted Texture Streaming

In this chapter, I propose a light-weight, decentralized and effective peer selection strategy for peer-assisted texture systems which can minimize the server bandwidth consumption while respecting the latency constraints of textures.

7.1 System Model

The most popular metaverse today is Second Life [12], where the virtual world consists of non-overlapping and independent regions. Each region is managed by a server that maintains the states of all objects and avatars within the region. To simply the discussion, I restrict my focus to a single region in Second Life. Once a peer logs in, it contacts the server which ensures the persistency of the region by continually sending relevant updates of objects and avatars to the peer through the network. In addition, the server determines the relevant subset of data that is required by each peer according to its location and visibility.

Each peer can freely roam around within the region and interact with other peers and objects within its *area-of-interest* (AoI), which is defined as a circular space around the peer with a given radius. The peers in the AoI of a peer are called AoI neighbors. Each peer needs to download the textures within its AoI that are not stored in its *cache* as it is moving in the region. After receiving the demanded textures, it will store them in its cache, from where a view is rendered

according to the spatial information (e.g., location and orientation) of the objects that the textures apply to. The size of the cache is determined by the metaverse provider or peers themselves. I assume that the default cache size of each peer is 128 MB (default cache size of users in Second Life). A simple but effective cache replacement algorithm is used in our setting: the cached textures that are far way from the avatar will be replaced with the newly downloaded textures when the cache is full.

To cut the server bandwidth cost, every peer is allowed to download the required textures from other peers by sending requests to others. If a peer receives requests from other peers, it is regarded as a content *provider* of those peers, and I assume every peer can be a provider to others. These requests are processed by content providers in a FIFO (First In First Out) manner. To maintain a good user satisfaction, all the requested textures must be delivered within a certain time limit, defined as the *latency constraint* of textures. If a peer fails to obtain the requested textures from its content providers within the latency constraint (due to the upload constraints of its content providers), it will send the requests to the server, thus consuming server bandwidth.

Let P and T denote the set of online peers and the set of textures in a region, respectively. The size of texture $i \in T$, denoted by w_i, is measured in units of kilobytes (i.e., KB). Let r_i^j denote the request sent by peer j for texture i, and the *weight* of the request r_i^j be the size of the requested texture (i.e., w_i). The latency constraint for each texture is denoted by d, measured in seconds. Without loss of generality, I assume online peers are heterogeneous and have different upload capacities. Let u_i denote the upload capacity of peer i, measured in kilobytes per second, i.e., KB/s. It is critical to note that the upload capacity is assumed to be the only bottleneck in the system, since the bottleneck caused by the constrained download capacities of peers can be eliminated by reducing the number of requests accordingly.

Let $R_i = \cup_{j \in P}\{r_i^j\}$ be the set of requests sent by all peers for texture i. Since every texture is unique, we have $R_i \cap R_j = \emptyset$ for any two textures i and j. In order to avoid receiving duplicate textures which would significantly increase bandwidth overhead and server bandwidth cost, every request is sent to only one of the peers. In other words, every request has only one provider. Similarly, let

Term	Definition
P	Set of online peers.
T	Set of textures.
u_i	Upload capacity of peer $i \in P$.
w_i	Size of texture $i \in T$.
r_k^i	Request sent by peer i for texture k.
R_k	Set of requests for texture k. $R_k = \cup_{i \in P}\{r_k^i\}$
S_k	Set of peers that have texture k in their caches.
l_i	Load of peer $i \in P$.
$h_k^{i,j}$	Height of request r_k^i at its provider, peer j.
d	Latency constraint of textures.

Table 7.1: List of notations and their definitions.

$S_i \subseteq P$ be the set of peers that have texture i in their caches. The information of S_i can be piggybacked into the updates sent by the server through the network. The *load* of peer i, denoted by l_i, is defined as the total weights of requests sent to peer i. The *height* of request r_k^i (sent by peer i) at peer j, denoted by $h_k^{i,j}$ (since every request is unique and has only one provider), is defined as the load of peer j directly after request r_k^i arrives at peer j. The notations that will be used throughout this chapter are summarized in Table 7.1.

7.2 Problem Formulation

Since the amount of data that can be delivered by peer i within the latency constraint is $d \cdot u_i$, the server will disseminate the rest of the amount of data, if any, to the corresponding peers that sent their requests to peer i. Therefore, the server will send $\max\{l_i - d \cdot u_i, 0\}$ of data due to the inadequate upload capacity of peer i. Our goal is to minimize the bandwidth consumption at the server while respecting the upload capacities of heterogeneous peers, which can be formally

stated as follows:

$$(\mathbf{F1})\ \min \quad \sum_{i \in P} \max\{ \sum_{R_k \neq \emptyset} \sum_{i \in S_k} x_k^{j,i} w_k - d \cdot u_i, 0\} \tag{7.1}$$

$$\text{s.t.} \quad \sum_{i \in S_k} x_k^{j,i} = 1,\ \forall r_k^j \in R_k \tag{7.2}$$

$$x_k^{j,i} \in \{0,1\},\ \forall r_k^j \in R_k. \tag{7.3}$$

As shown in Eq. 7.3, $x_k^{j,i}$ is an indicator variable indicating whether the request r_k^j is sent to peer i. Specifically, if r_k^j is sent to peer i, $x_k^{j,i} = 1$; otherwise $x_k^{j,i} = 0$. Eq. 7.2 shows that every request has only one provider. The load of peer i is

$$l_i = \sum_{R_k \neq \emptyset} \sum_{i \in S_k} x_k^{j,i} w_k. \tag{7.4}$$

Note that the equivalent problem of minimizing the bandwidth consumption at the server is to maximize the total amount of data that is transmitted by peers within the latency constraint of textures (i.e., d seconds), which can be formulated as follows:

$$(\mathbf{F2})\ \max \quad \sum_{j \in P} \sum_{k \in T} w_k x_k^{j,i} \tag{7.5}$$

$$\text{s.t.} \quad \sum_{i \in S_k} \sum_{k \in T} w_k x_k^{j,i} \leq d \cdot u_i,\ \forall r_k^j \in R_k, \tag{7.6}$$

$$x_k^{j,i} \in \{0,1\},\ \forall r_k^j \in R_k. \tag{7.7}$$

It can be observed that the alternative formulation (i.e., **F2**) is a multidimensional knapsack problem [72], which is NP-complete [73] in general. Many approaches have been proposed to solve it, such as LP-relaxation [74], the primal-dual method [75] and dynamic programming [76].

I would like to point out that centralized approaches are not desirable for the design of peer selection strategies in peer-assisted texture streaming systems since a global connectivity information and the request information is required to obtain a feasible solution. Therefore, the computation to solve **F1** (Eq. 7.1) or **F2** (Eq. 7.5) will be invoked upon churn (i.e., peer's leaving or joining) and avatar mobility that significantly changes the set of requests. As the number of

peers increases, the overhead of communication and processing is not tolerable in practice.

Although there exist decentralized [77] and online [75] [78] algorithms that compute approximate solutions within poly-logarithmic communication rounds, they are based on the assumption that each peer has the same capacity and is aware of global information, i.e., it knows about loads and upload capacities of peers, and the connectivity information between any two peers. However, I consider the scenario where peers have different upload capacities and each requested texture needs to be downloaded within the latency constraint. As a result, existing centralized or decentralized algorithms which aiming to solve Eq. 7.5 in the formulation **F2** are not suitable for peer-assisted texture streaming.

By combining Eq. 7.1 and Eq. 7.4, the objective function of **F1** can be simplified to

$$\min \quad \sum_{i \in P} \max\{l_i - d \cdot u_i, 0\}, \tag{7.8}$$

where $\max\{l_i - d \cdot u_i, 0\}$ is the amount of bandwidth consumed by the server due to the inadequate upload capacity of peer i. Therefore, I transform **F1** into an asymmetric load balancing problem: how to distribute the loads on *overloaded* peers (i.e., $l_i > d \cdot u_i$) to *underloaded* peers (i.e., $l_i < d \cdot u_i$).

The objective of this chapter is to design a light-weight peer selection strategy that can be implemented in a decentralized and online fashion. Considering that a practical peer selection strategy should be performed in a decentralized manner, I model the peer selection strategy as a *congestion game* and thus inherit its practical and decentralized nature, which has been successfully used to model load balancing problems in P2P networks as well as many other real world applications due to its conceptual simplicity.

7.3 Minimizing Server Bandwidth Cost

The server bandwidth minimization problem can be formulated as an asymmetric weighted congestion game, denoted by G, which is defined as

$$G = (N, M, (A_i)_{i \in N}, (f_e)_{e \in M}), \tag{7.9}$$

where $N = \cup_{k \in W} R_k$ and $M = \cup_{k \in W} S_k$ correspond to the set of requests and peers (i.e., providers) in our scenario, respectively. Each request $i \in N$ has a weight w_i (i.e., the size of the requested texture), and each peer $i \in M$ has an upload capacity u_i. The strategy set of request i, denoted by A_i, is the set of peers M_i that have stored the requested texture in their caches. Since every request is sent to only one of the peers at a time, G is a *singleton* congestion game, i.e., $\forall i \in N$, $a_i \in A_i$.

The strategy profile of G, denoted by $\mathbf{a} = (a_i)_{i \in N}$ corresponds to the peer selection profile, and $\mathcal{A} = \times_{i \in N} A_i$ corresponds to the set of all possible peer selection profiles. The cost of request i given a peer selection profile $\mathbf{a}$ is defined as

$$c_i(\mathbf{a}) = \min\{\max\{h_i - d \cdot u_j, 0\}, w_i\}, \tag{7.10}$$

where $j = a_i$ is the peer selected by request i and h_i is the *height* of request i at peer j. Clearly, for each peer $j \in M$, the sum of the cost of requests that select peer j is

$$\sum_{a_i = j} c_i(\mathbf{a}) = \sum_{a_i = j} \min\{\max\{h_i - d \cdot u_j, 0\}, w_i\} \tag{7.11}$$

$$= \max\{l_j - d \cdot u_j, 0\}, \quad \forall a_i \in \mathbf{a}. \tag{7.12}$$

Recall that $\max\{l_j - d \cdot u_j, 0\}$ is the amount of data that the server will disseminate to the corresponding peers due to the upload capacity limitation of peer j. I define the potential function of G is as the sum of the cost of all requests:

$$\Phi(\mathbf{a}) = \sum_{j \in M} \sum_{a_i = j} c_i(\mathbf{a})$$

$$= \sum_{j \in M} \max\{l_j - d \cdot u_j, 0\}. \tag{7.13}$$

Therefore, the problem of minimizing the server bandwidth cost can be solved by minimizing the potential function of the corresponding congestion game. The peer selection profile, i.e., $\mathbf{a}$, has a significant impact on the server bandwidth cost, which is minimized if $\mathbf{a}$ is a pure Nash equilibrium (i.e., $\mathbf{a} = \mathbf{a}^*$ as shown in Eq. 5.3).

7.4 Peer Selection Strategy

Based on the established connection between the congestion game G (see Eq. 7.9) and the server bandwidth cost minimization problem, an optimal peer selection strategy can be thought of as an optimal strategy profile, i.e., a pure Nash equilibrium of G. It has been shown that finding a pure Nash equilibrium in a congestion game is PLS-complete [66], hence the number of changes of the strategy profile required from one state to any pure Nash equilibrium is exponentially large. Therefore, our goal is to quickly reach an *approximate* Nash equilibrium from any state by enabling each peer to change its strategies for its requests according to a *bounded jump rule* (see lines 14 and 19 in Algorithm 9), which will be discussed later in this section.

It is worth highlighting that the offline peer selection strategies are undesirable in practice due to both the additional delay they incur to find appropriate content providers for all peers, and the inefficient utilization of peer bandwidth (since the peer bandwidth cannot be used to delivery texture data during the process of finding a good peer selection strategy). Furthermore, metaverses are highly dynamic environments such that churn, avatar mobility, and time-varying peer capacities will significantly degrade the feasibility and efficiency of offline peer selection strategies.

I introduce a light-weight online peer selection strategy, shown in Algorithm 9, which reduces the server bandwidth cost by enabling each peer to repeatedly *update* its strategy for each texture that it requests. More specifically, all requests are sent by peers in a repeated fashion, i.e., the strategy of each request can be changed if (1) the current content provider of the request is overloaded, and (2) the corresponding peer finds an underloaded provider within the latency constraint.

Our approach is an online algorithm such that each peer performs the peer selection process and the texture transmission process in parallel. Each request is sent to a peer once it is generated (line 10 in Algorithm 9). Each peer i can change its strategy for each request in a repeated fashion if the bounded jump rule is satisfied. Also, our algorithm is light-weight such that each peer i needs to contact u.a.r. (i.e., uniformly at random) only one other peer in $S_k(d')$ for each request r_k^i

Algorithm 9: Peer selection for each peer $i \in P$.

1 Let t be the local time of peer i;
2 Let r_k^i be the request sent by peer i for texture $k \in T$;
3 Let t_k be the time when r_k^i is generated;
4 Let a_i^k be the provider of the request r_k^i;
5 **foreach** $r_k^i \in R_k$ **do**
6 Contact a peer j from $S_k(d)$ u.a.r. ;
7 **if** *timeout* **then**
8 Go to line 6;
9 $a_i^k \leftarrow j$;
10 Send r_k^i to peer a_i^k;
11 Let h_k^i be height of r_k^i at peer j;
12 **if** $h_k^i \leq d \cdot u_j$ **then**
13 Stop contacting other peers;
14 **while** $t \leq t_k + d$ *and* $h_k^i - (d - t + t_k)u_{a_i^k} \geq w_k$ **do**
15 Contact a peer j' from $S_k(d - t + t_k)$ u.a.r. ;
16 **if** *timeout* **then**
17 Go to line 14;
18 Let $l_{j'}$ be the load of peer j';
19 **if** $l_{j'} \leq (d - t + t_k)u_{j'}$ **then**
20 $a_i^k \leftarrow j'$;
21 Send r_k^i to peer a_i^k;
22 **else**
23 Go to line 14;
24 Let h_k^i be height of r_k^i at peer j';
25 **if** $h_k^i < w_k + (d - t + t_k) \cdot u'_j$ **then**
26 Stop contacting other peers;
27 Send the request for texture k directly to the server;

with latency constraint d' at one time, where $S_k(d') = \{i \in S_k : l_i < d'u_i\}$. The load and capacity information (i.e., l_i and u_i) of S_k is maintained by the server

and can be piggybacked onto the update packets in practical implementations. The current latency constraint (i.e., $d' = d - t + t_k$) of the request is calculated at the requesting peer, where t and t_k are the current time and the time when the request is generated, respectively. It is worth noting that every peer in S_k has the same probability being selected, although some peers have failed to satisfy the bounded jump rule in previous rounds. Because both requests and caches of peers are highly dynamic due to the avatar mobility.

The key idea of the bounded jump rule is to restrict the change of strategies such that the potential of the system decreases monotonically. Since all requests are processed by their content providers in a FIFO fashion, the peers that have the requests which can be processed within their latency constraints will not consider changing their current strategies (see lines 12 and 25 in Algorithm 9). From the perspective of congestion games, those requests (i.e., players in games) have no incentives to change their strategies since their costs cannot be reduced (for pure Nash equilibrium) or not significantly reduced (for ϵ-Nash equilibrium).

More formally, for a given request r_k^i (sent by peer i for texture k) at peer a_i^k with a weight w_k and a height $h_k^i(t)$, by Eq. 7.10, the cost of r_k^i at time t is

$$c_{r_k^i}(\mathbf{a}(t)) = \min\{\max\{h_k^i(t) - (d - t + t_k)u_{a_i^k}, 0\}, w_k\},$$

where $\mathbf{a}(t)$ is the strategy profile of all peers at t, and $(d - t + t_k)$ is the latency constraint of texture k at t. This implies

$$c_{r_k^i}(\mathbf{a}(t)) = w_k \iff h_k^i(t) - (d - t + t_k)u_{a_i^k} \geq w_k. \tag{7.14}$$

Therefore, a peer will consider to change its strategy for its requests only when the costs of requests equal their weights, otherwise it will stop contacting other peers (line 26 in Algorithm 9). If the requesting peer of request r_k^i (i.e., peer i) finds another peer j' at t (line 15 in Algorithm 9) during the repeated update process (the loop from line 14 to line 26 in Algorithm 9) such that $l_{j'}(t) \leq (d - t + t_k)u_{j'}$, it will send the request r_k^i to peer j'. It is worth noting that multiple peers may find peer j' and send requests to peer j' at the same time. In the worst case, only the first request can be processed by peer j' since peers process requests in a FIFO manner. Finally, the request will be sent to the server if $c_{r_k^i}(\mathbf{a}(t)) = w_k$ for any $t \in [t_k, t_k + d]$.

With the bounded jump rule, it is easy to observe that the potential function of game G is monotonically decreasing such that for a given set of requests and $\forall t_1, t_2$, we have

$$t_1 \leq t_2 \implies \Phi(\mathbf{a}(t_1)) \leq \Phi(\mathbf{a}(t_2)). \tag{7.15}$$

Furthermore, our algorithm applies to a fully distributed and concurrent setting such that all peers can change their strategies for their requests at the same time without a centralized coordination.

7.5 Convergence Analysis

Recall that the proposed peer selection strategy follows principles and mechanisms of congestion games, and thus it is carried out in a repeated fashion (see Algorithm 9). More specifically, every peer i will repeatedly contact a provider j in S_k for each texture k that it requests (i.e., r_k^i), and it will change its strategy for the request (i.e., a_i^k) if the bounded jump rule (see lines 14 and 19 in Algorithm 9) is satisfied. In addition, peers perform this process individually without centralized coordination.

I formulate the problem of minimizing the server bandwidth cost as a congestion game, and I define the potential of the congestion game as the server bandwidth cost (see Eq. 7.13), which is minimized when a Nash equilibrium (a steady state) is reached. With the proposed bounded jump rule, a steady state is a ϵ-Nash equilibrium of the congestion game. Therefore, a critical question to investigate is *how fast does the system reach a ϵ-Nash equilibrium from any state?* In what follows, I seek to investigate the upper bound of the convergence time of our peer selection strategy.

Recall that the potential function of the game G with the bounded jump rule is monotonically decreasing in the presence of concurrent changes of peers' strategies (peers update their content providers (i.e., peers in M) individually), because peers only consider to change the strategies of the requests that are sent to overloaded peers. More specifically, peer i will change the content provider of request r_k^i, if and only if $h_k^{i,j} - (d - t + t_k)u_j \geq w_k$ (i.e., Eq. 7.14 and Line 14 in Algorithm 9), where t and t_k are the local time and the time when the request is

generated, respectively. The equivalent condition is $h_k^{i,j} + (t - t_k)u_j - w_k \geq d \cdot u_j$. Therefore, I can analyze the convergence rate of the proposed peer selection strategy via analyzing the convergence rate of the corresponding congestion game.

Theorem 11. *Given a game $G = (N, M, (A_i)_{i \in N}, (f_e)_{e \in M})$ that satisfies the bounded jump rule, where $N = \cup_{k \in W} R_k$ and $M = \cup_{k \in W} S_k$ is the set of requests and content providers, respectively. Then a ϵ-Nash equilibrium can be reached within $O(\log n)$ rounds, where n is the number of online peers.*

Proof. Let $\Phi(t)$ be the potential of the game at time t. According to Eq. 7.13, we have $\Phi(t) = \sum_{k \in R(t)} w_k$. Similarly, let $S(t) = \cup_{k \in R(t)} S_k(d)$ be the set of underloaded peers at time t, where $S_k(d) = \{i \in S_k : l_i < d \cdot u_i\}$ denotes the set of underloaded peers that have texture k in their caches. Let $n = |P|$ and $m_t = |S(t)|$ be the number of online peers and the number of underloaded content providers at time t, respectively.

Note that a pure equilibrium can be reached at time t' if (1) $\Phi(t') = 0$ (all requests can be delivered by peers within the latency constraint), or (2) $m = 0$ (no more underloaded content providers). Therefore, I introduce a non-increasing function $\Psi(t) = m_t \Phi(t)$ to analyze the convergence rate of proposed peer selection strategy.

At time $t + 1$, let $\mathbf{x} = \{x_1, x_2, ..., x_{m_t}\}$ be a vector, where x_i denote the total size of requests that migrate to peer $i \in S(t)$. Obviously, $\Phi(t) = \sum_{i=1}^{m_t} x_i$. By applying CauchySchwarz inequality to $\mathrm{E}[\Psi(t + 1)]$, which is the expected value of $\Psi(t + 1)$, we have the following:

$$\mathrm{E}[\Psi(t+1)] = \mathrm{E}[m_{t+1}\Phi(t+1)] \tag{7.16}$$

$$\leq \left(\mathrm{E}[m_{t+1}^2]\mathrm{E}[\Phi^2(t+1)]\right)^{\frac{1}{2}}. \tag{7.17}$$

Noting that the square root function $f : x \to \sqrt{x}$ and the function $g : x \to x^2$ are convex functions, I apply Jensen's inequality to Eq. 7.17 and get

$$\mathrm{E}[\Psi(t+1)] \leq \mathrm{E}[m_{t+1}]\mathrm{E}[\Phi(t+1)] \tag{7.18}$$

$$= \sum_{m=1}^{m_t} m \Pr(m_{t+1} = m)\mathrm{E}[\Phi(t+1)|m_{t+1} = m], \tag{7.19}$$

where m is the number of underloaded content providers in $S(t+1)$ at time $t+1$. The expected value of the potential function given m is

$$\mathrm{E}[\Phi(t+1)|m_{t+1}=m] = \Phi(t) - \frac{\binom{m-1}{m_t-1}}{\binom{m}{m_t}} \sum_{i=1}^{m_t} x_i \tag{7.20}$$

$$= \left(1 - \frac{m}{m_t}\right) \Phi(t). \tag{7.21}$$

Combining Eq. 7.21 and Eq. 7.19, we have the following:

$$\mathrm{E}[\Psi(t+1)] \leq \sum_{m=1}^{m_t} m \left(1 - \frac{m}{m_t}\right) \Phi(t) \Pr(m_{t+1}=m) \tag{7.22}$$

$$\leq \frac{m_t}{4} \Phi(t) \sum_{m=1}^{m_t} \Pr(m_{t+1}=m) \tag{7.23}$$

$$= \frac{1}{4} \Psi(t). \tag{7.24}$$

Let τ be the number of rounds that a ϵ-Nash equilibrium (i.e., $\Psi(t+\tau) = O(1)$) is reached from time t. By Lemma 12 (see right column), we have

$$\mathrm{E}[\tau|\Psi(t)] \leq \log\left(\frac{\Psi(t)}{\Psi(t+\tau)}\right) \tag{7.25}$$

$$= \log\left(\Psi(t)\right). \tag{7.26}$$

Note that the maximum value of of $\Psi(t)$ is $|T|n^2$ ($|T|$ is the total number of textures), which occurs in the case when every peer requests all the textures in the region. Therefore, $\mathrm{E}[\tau] \leq 2\log(|T|n)$. By Markov's inequality, we have that

$$\Pr(\tau \geq 40\log(|T|n)) \leq \frac{\mathrm{E}[\tau]}{40\log(|T|n)} = 0.05. \tag{7.27}$$

This implies that $\Pr(\tau \leq 40\log(|T|n)) \geq 0.95$. As a result, with a high probability, a ϵ-Nash equilibrium can be reached from any state within $O(\log n)$ rounds. □

Lemma 12. *Let $X_1, X_2, \ldots$ denote a sequence of non-negative random variables and assume that for all $i \geq 0$*

$$\mathrm{E}[X_i|X_{i-1}=x_{i-1}] \leq \alpha \cdot x_{i-1}$$

for some constant $\alpha \in (0,1)$*. Furthermore, fix some constant* $x^* \in (0, x_0]$ *and let* τ *be the random variable that describes the smallest* t *such that* $X_t \leq x^*$*. Then,*

$$\mathrm{E}[\tau | X_0 = x_0] \geq \frac{2}{\log(1/\alpha)} \cdot \log\left(\frac{x_0}{x^*}\right). \tag{7.28}$$

Proof. The complete proof is shown in [79]. □

Theorem 11 indicates that the proposed peer selection strategy can rapidly reach a ϵ-Nash equilibrium within $O(\log n)$ communication rounds from any state, where n is the total number of online peers in the region.

7.6 Evaluation

In this section, the proposed peer-assisted texture streaming system is evaluated using a comprehensive simulation program with input from realistic traces collected from the most popular metaverse application - Second Life. I choose Second Life to conduct our evaluation, as the Second Life Viewer[1] and OpenMetaverse[2] are open source and provide functionalities to collect the texture information and location information of avatars from Second Life servers.

For comparisons, I have also simulated the following three peer selection algorithms using the same implementation settings with the proposed algorithm: (1) C/S scheme, where all the requests are solely served by the centralized server; (2) FLoD [42], where each peer can send its requests to AoI neighbors; if AoI neighbors fail to respond, it will send the requests to the server; (3) BAPS [44], which allocates the upload capacities of peers into channels and each peer will select the provider that in the currently connected channels; if the provider is overloaded, it will pick a peer from AoI neighbors as the provider; if that fails, it will randomly select a peer beyond of the AoI; eventually the request will go to the server if no appropriate provider is found.

[1] http://wiki.secondlife.com/wiki/Source_downloads
[2] http://openmetaverse.org/

Region	Number/Size of Textures	User Hours
Freebies	2940 / 0.71 GB	201.1
Waterhead	407 / 0.21 GB	580.1
Orientation Island	87 / 0.56 GB	604.7
Mauve	621 / 0.57 GB	270.7
Japan Resort	551 / 0.41 GB	364.5

Table 7.2: Total number and size of textures and user hours of different regions during a day.

7.6.1 Data Collection and Simulation Setup

I collect the realistic texture information of 5 regions (i.e., Freebies, Waterhead, Orientation Island, Mauve, and Japan Resort) from Second Life. The 2D positions of textures, and the distributions of texture sizes of different regions are shown in the left column of Figures 7.1 and 7.2, respectively. The total size of textures and user hours (i.e., the sum of the combined length of all user sessions) during a day in these regions are summarized in Table 7.2. Before logging into the Freebies region, I clear the cache of our Second Life Viewer client to store all the textures after I log in. After logging into each region, I move the avatar within the region step by step. Specifically, I stop the avatar at each step and wait for all textures within the AoI to be downloaded and rendered before moving to the next step until the whole region is covered and all textures within the region have been downloaded.

It is worth noting that I only collect the textures of objects instead of avatars, since the positions of textures on avatars may change over time due to the mobility and behavior of avatars. Since a texture can be applied to multiple objects that exist in different locations, textures can have multiple positions in our simulations.

I collect the one-day avatar mobility traces of the 5 regions from Second Life. As shown in the right column of Figure 7.1, the number of concurrent users changes over time. I assume that every peer joins the system with a cleared cache, hence peers will request all the textures in their AoIs immediately after they join the system. As avatars move around in the virtual world, they will request the textures within their AoI that are not stored in their caches. The

Region	Server Bandwidth Cost (GB)				Fraction of Saved Bandwidth		
	C/S	FLoD	BAPS	Proposed	FLoD	BAPS	Proposed
Freebies	215.6	201.1	183.5	166.1	6.7 %	14.8 %	**23.0** %
Waterhead	17.5	8.4	6.6	2.8	52.2 %	62.2 %	**84.1** %
Orientation	20.3	11.1	10.5	5.4	45.2 %	48.1 %	**73.4** %
Mauve	21.6	17.6	16.4	13.7	18.3 %	23.8 %	**36.4** %
Japan Resort	32.6	25.4	24.7	19.4	22.1 %	24.2 %	**40.5** %

Table 7.3: Server bandwidth consumption (smaller is better) and the fraction of saved bandwidth (larger is better) of different algorithms during a day, when the latency constraint of textures is set to 5 seconds.

default cache size is set to 128 MB for each peer. I use a simple but efficient cache replacement algorithm such that the cached textures that are far way from the avatar will be replaced with the newly downloaded textures when the cache is full.

To evaluate the proposed algorithm in a heterogeneous environment, every peer is assigned an upload capacity and a download capacity randomly selected from a realistic data set collected from dslreports.com [63], as shown in Figure 7.3. Note that peers (especially peers with wireless access) may have bandwidth (both upload and download) fluctuations, I assume that each peer can utilize 80% of their assigned capacities most of the time. The AoI radius of each peer is set to be 64 m, which is the default value in Second Life.

7.6.2 Server Bandwidth Consumption

Recall that our main objective is to reduce the server bandwidth consumption via utilizing the network capacities of participating peers, hence the main performance metric in our simulation is the average server bandwidth consumption in one day. Three different algorithms are evaluated using collected avatar mobility traces during a day. I sum up the total amount of textures required and total amount of texture data that is delivered by the server.

Table 7.3 shows the efficiency of different schemes in terms of server bandwidth consumption when the latency constraint of all textures is set to 5 seconds,

Region	Server Bandwidth Cost (GB)				Fraction of Saved Bandwidth		
	C/S	FLoD	BAPS	Proposed	FLoD	BAPS	Proposed
Freebies	215.6	189.9	161.0	131.4	11.9 %	25.3 %	**39.0** %
Waterhead	17.5	5.1	3.0	1.3	70.9 %	82.8 %	**92.2** %
Orientation	20.3	7.7	6.5	3.2	61.9 %	68.2 %	**84.0** %
Mauve	21.6	14.9	12.5	9.3	30.5 %	41.8 %	**56.5** %
Japan Resort	32.6	21.3	20.3	12.6	34.6 %	37.9 %	**61.3** %

Table 7.4: Server bandwidth consumption (smaller is better) and the fraction of saved bandwidth (larger is better) of different algorithms during a day, when the latency constraint of textures is set to 10 seconds.

i.e., all the required textures should be delivered (from peers or the server) to the destinations within 5 seconds. It can be observed that all the algorithms except the C/S scheme can reduce the server bandwidth consumption in all the regions via utilizing the bandwidth of participating peers. It can be observed that the proposed peer selection strategy outperforms the other alternatives by a substantial margin in any region.

Table 7.4 shows the server bandwidth consumption when the latency constraint of all textures is set to 10 seconds. Since relaxing the latency constraint of textures increases the number of requests that peers can satisfy, the server bandwidth consumption is reduced significantly. It is worth noting that relaxing the latency constraint of textures inevitably degrades the user satisfaction.

In the case when the number of textures is large (e.g., the Freebies region with 2940 textures), the number of requests is large. Therefore the server bandwidth consumption is high due to the inadequate capacities of peers which cannot satisfy all the demand. Nevertheless, the proposed algorithm can save 23% and 39% of server bandwidth consumed by the C/S scheme in the Freebies region when the latency constraints of textures are 5 seconds and 10 seconds, respectively. In the case when the number of textures is relatively small (i.e., less than 1000), peer-assisted schemes can save a large amount of server bandwidth. Moreover, it can be seen in Tables 7.3 and 7.4 that the proposed peer selection strategy significantly outperforms existing algorithms since it can utilize the peer capacities more efficiently than FLoD and BAPS.

Region	FloD	BAPS	Proposed
Freebies	76.3 %	66.3 %	**50.5** %
Waterhead	25.1 %	20.1 %	**10.2** %
Orientation Island	20.1 %	15.9 %	**7.5** %
Mauve	47.4 %	42.1 %	**31.8** %
Japan Resort	45.4 %	42.7 %	**30.2** %

Table 7.5: Expected server request ratio (smaller is better) in different regions, when the latency constraint of textures is 5 seconds.

Region	FloD	BAPS	Proposed
Freebies	66.3 %	49.6 %	**35.8** %
Waterhead	13.4 %	10.2 %	**6.6** %
Orientation Island	12.6 %	8.3 %	**5.7** %
Mauve	37.4 %	30.1 %	**20.1** %
Japan Resort	35.6 %	32.8 %	**19.4** %

Table 7.6: Expected server request ratio (smaller is better) in different regions, when the latency constraint of textures is 10 seconds.

7.6.3 Server Request Ratio

Another performance metric I use is the server request ratio, which is defined as the percentage of requests in the region that are successfully served by the server. Intuitively, a higher server request ratio results in both a higher server bandwidth consumption and a higher server load. Figure 7.4 show the distribution of server request ratio of different algorithms when the latency constraints of textures are 5 seconds and 10 seconds, respectively.

It can be observed in Tables 7.5 and 7.6 that our peer selection strategy achieves a lower expected server request ratio than other existing algorithms in all regions and under all implementation settings. This is because the requests with the proposed peer selection strategy can be sent to underloaded peers rapidly within the latency constraint of textures, so that the peers' bandwidth is efficiently utilized. Therefore, the server load and bandwidth consumption can be reduced (since fewer requests will be processed). Benefitting from this, the system scalability can be improved.

Region	Communication Overhead	
	$d = 5$ Seconds	$d = 10$ Seconds
Freebies	2.36 KB/s	2.16 KB/s
Waterhead	0.26 KB/s	0.12 KB/s
Orientation Island	0.08 KB/s	0.06 KB/s
Mauve	1.02 KB/s	0.86 KB/s
Japan Resort	1.04 KB/s	0.88 KB/s

Table 7.7: Averaged communication overhead per peer under different latency constraints (i.e., d) of textures in different regions.

7.6.4 Overhead

Since the protocol overhead incurred is naturally an important metric for every distributed algorithm, I investigate the communication overhead incurred by the proposed peer selection strategy in the last group of simulations. I compute the average overhead per peer as shown in Table 7.7. It is easy to observe that the communication overhead of our algorithm is very low, with a consumption of less than 3 KB/s per peer in all regions. This is because each peer contacts only one peer for each request at each round, and the sizes of request packets and their corresponding acknowledge packets are usually small (less than 100 bytes per packet). Considering that the requests are processed in a FIFO fashion at each peer, I believe that the processing overhead is reasonable in practice.

(a) Freebies

(b) Waterhead

(c) Orientation Island

(d) Mauve

(e) Japan Resort

Figure 7.1: 2D positions of textures (left) and the number of concurrent users (right) in different regions, each of which is 256×256 meters.

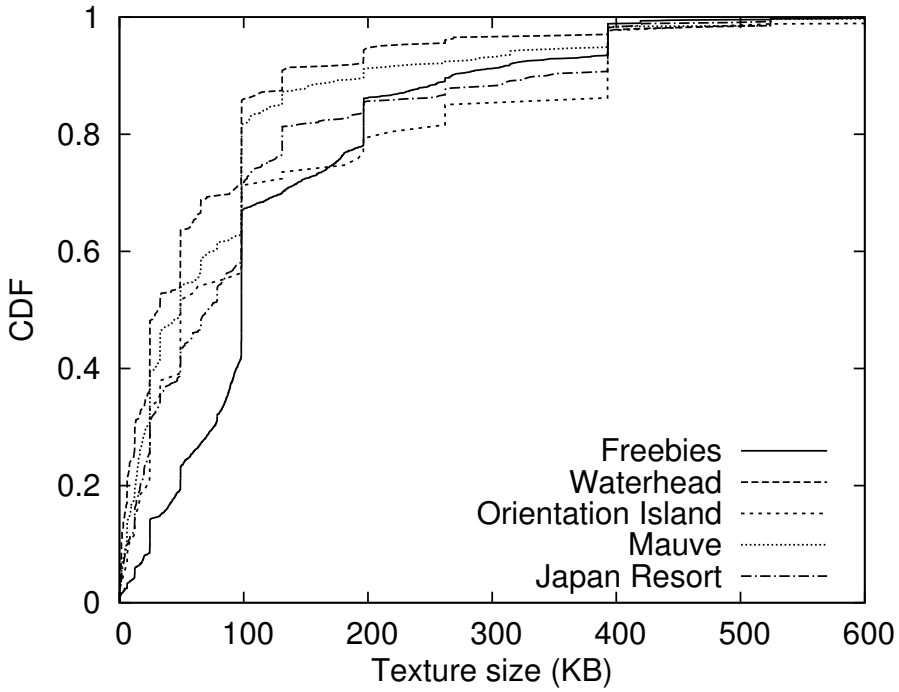


Figure 7.2: Distribution of textures sizes.

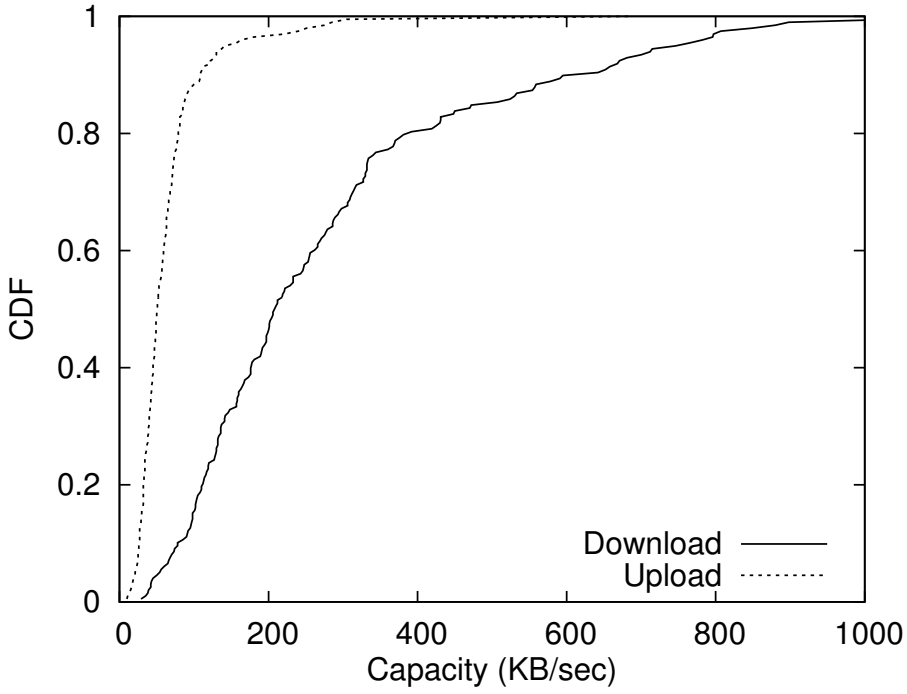


Figure 7.3: Distribution of peer bandwidth.

5 seconds (a) Freebies 10 seconds

5 seconds (b) Waterhead 10 seconds

5 seconds (c) Orientation Island 10 seconds

5 seconds (d) Mauve 10 seconds

5 seconds (e) Japan Resort 10 seconds

Figure 7.4: Distribution of server request ratio (smaller is better) in different regions during a day, when the latency constraint of textures is set to 5 seconds and 10 seconds.

7. GAME-THEORETIC APPROACH FOR PEER-ASSISTED TEXTURE STREAMING

8

Conclusion and future work

In this thesis, I have introduced the design of peer-to-peer based approaches for the distribution of interactive 3D media (i.e., 3D audio and 3D texture) in networked virtual environments. The main motivation of this work is not only the desire to create a 3D audio experience that is more congruent with the three-dimensional visual display of virtual worlds while at the same time avoiding large, centralized server resources, but reduce the server bandwidth consumption for 3D content dissemination in metaverses (NVEs). The results indicate that indeed, such peer-to-peer based approaches are not only feasible but can achieve good performance.

For 3D audio streaming in NVEs, I first introduced a heuristic Intra-AoI approach which constructs multicast trees for spatial audio delivery in NVEs. The main motivation of the Intra-AoI approach is to quickly deploy multicast trees for the speakers and achieve high system reachability, while at the same time keeping the end-to-end latency as low as possible. In particular, the Intra-tree approach constructs a audio dissemination tree for each speaker in two stages. Firstly, a speaker constructs a MLT (i.e., minimum latency tree) without considering the bandwidth limits of the neighbors, trying to minimize the end-to-end latency. Secondly, the MLT is pruned according to the bandwidth limits and weights of neighbors. The tree pruning process in this stage will result in orphan nodes, which will be attached to the MLT by performing a breadth-first-search on the pruned tree.

With cross-tree adjustment, the audio dissemination trees can be incrementally constructed and adjusted, Moreover, the system reachability can be further improved via optimizing the bandwidth allocation at the conflicting nodes incurred by multiple concurrent speakers. Besides, audio mixing techniques can be applied to increase the system reachability with the price of sacrificing users' perception of the 3D audio.

Considering that the Intra-AoI approach can only utilize the upload bandwidth of users within the speakers' AoI, the performance that it achieves is sub-optimal and is vulnerable to the system dynamics. The two game-theoretic approaches, termed UCG and WCG, are proposed in this thesis to provide solutions for the following challenges at the same time: (1) maximizing the system reachability, (2) satisfying tight latency constraints, and (3) adapting to system dynamics. With UCG and WCG, users can efficiently utilize the upload capacities of all the participating users in a system wide manner without global information of the network. I proved that the reachability of the system with UCG or WCG is maximized when the loads of users are balanced proportionally to their capacities. I then proposed a water-fitting algorithm for UCG and a proportional load balancing (PLB) algorithm for WCG to balance the loads of users in a fully distributed manner. Both algorithms can lead the system to converge to a ϵ-Nash equilibrium in expected $O(\log \log n)$ time, where n is the number of users in the system. The simulation results showed that the proposed algorithms are efficient and desirable in practical compared to the Intra-AoI approach.

In general, WCG will outperform UCG since WCG can utilize the bandwidth of users more efficiently with the non-uniform model for users' bandwidth which is not uniformly distributed in practical. Furthermore, every audio delivery path with WCG has at most 2 hops such that it can achieve a similar end-to-end performance as the client-server scheme.

Several additional interesting aspects should be explored in future as well. For example, Boulstead et al. [39] have introduced a foreground versus background distinction that places more importance on the accurate sound rendering from close audio sources. This concept can also be incorporated into our design to further increase the system reachability via mixing audio streaming for those less important receivers which are far way from the speakers. Additionally, bandwidth

allocation algorithms on the helpers (for UCG and WCG) in the game-theoretic approaches can be improved to increase the system reachability, since the bid price can be changed due to the mobility of avatars such that the number of neighbors can be changed. Last but not least, helper selection strategies that considering the incentive mechanism of users, malicious attack or cheating from users may be performed to make the P2P 3D audio streaming solution more robust in practical.

For texture streaming in metaverses, I introduced a peer-assisted texture streaming system in this thesis to minimize the server bandwidth cost without degrading the end-user satisfaction in metaverses. I formulated the problem of server bandwidth minimization problem as a multidimensional knapsack problem which is NP-complete in general. To propose a approximate solution that can achieve a near-optimal performance, I formulate the problem as a congestion game, and use the concept of congestion games to design a peer selection strategy. The proposed peer selection algorithm is light-weight, and it can efficiently utilize the bandwidth of heterogeneous peers in a decentralized manner by enabling each peer to repeatedly update its content providers independently and concurrently. The algorithm was evaluated through an extensive comparison study based on simulations using realistic texture information and avatar mobility traces collected from Second Life. As shown by the simulation results, the proposed algorithm can effectively reduce the server bandwidth cost and increase the scalability of metaverses.

There are a lot of open issues in the area of 3D content streaming in metaverses. Firstly, statistic peer selection strategy may be explored. The idea of statistic peer selection strategy is to select those peers as content providers which have similar visibility and close to each others. Secondly, caching and prefetching techniques can be tailed for 3D streaming to further reduce the bandwidth consumption at the server.

It is worth noting that the two related problems are addressed in this thesis, which are 3D audio streaming and 3D texture streaming in networked virtual environments. Both applications are of the types of real-time many-to-many dissemination applications which usually require a large amount of bandwidth to deliver the audio or texture data. A common solution that targets on both the

two problems would be very challenging since there are two individual objectives (maximizing the system reachability and minimizing the server bandwidth consumption) that need to be optimized respecting a large number of constraints, such as the bandwidth limits of users, churn, avatar mobility, texture distribution, limited size of caches, and etc.

At last, encouraged by the conclusions in this thesis, it is worth to work on a real-world deployment of the proposal as standalone applications in the near future to provide 3D audio and texture streaming service for users.

9

Appendix

The following lemmas that used in this paper can be found in many references such as [67] and [80]. We define $I^n = I \times \cdots \times I$ (n copies), where I is an open interval.

Definition 4. *The vector* $\mathbf{x} = \{x_1, ...x_n\}$ *is called majorized by the vector* $\mathbf{y} = \{y_1, ..., y_n\}$, *denoted by* $\mathbf{x} \preceq \mathbf{y}$, *if* $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$, *and* $\sum_{i=1}^{k} x_{(i)} \leq \sum_{i=1}^{k} y_{(i)}, \quad k \in [1, n-1]$, *where* $x_{(i)}$ *and* $y_{(i)}$ *denote the* i*-largest elements in* $\mathbf{x}$ *and* $\mathbf{y}$, *respectively.*

Definition 5. *A function* $f : I^n \to \mathbb{R}$ *is called Schur-convex if for any two vectors* $\mathbf{x}, \mathbf{y} \in I^n$, $\mathbf{x} \preceq \mathbf{y} \Rightarrow f(\mathbf{x}) \leq f(\mathbf{y})$. *The function* f *is said to be* Schur-concave *if the inequality is reversed.*

Lemma 13. *[67]. Let* $f(x) = f(x_1, ..., x_n)$ *be symmetric and have a continuous partial derivative on* I^n. *Then* $f : I^n \to \mathbb{R}$ *is Schur-convex if and only if* $(x_i - x_j)\left(\frac{\partial f}{\partial x_i} - \frac{\partial f}{\partial x_j}\right) \geq 0$ *on* I^n. *The function* f *is said to be Schur-concave if the inequality is reversed.*

Lemma 14. *[67]. If* f_i *is Schur-convex (concave),* $i = 1, 2, ..., k$, *and* $f_i(x) \leq 0$ *for all* i *and* x. *Then* $f : I^n \to \mathbb{R}$ *is Schur-convex if and only if* $g(x) = \prod_{i=1}^{k} f_i(x)$ *is said to be Schur-convex (concave).*

Corollary 15. *Let* $q, m \in \mathbb{R}$, *and* $0 < q < 1$. *Let* $\mathbf{x} = \{x_1, ..., x_m\}$, *where* $x_i \in \mathbb{Z}$. *Then* $\sum_{i=1}^{m} q^{x_i}$ *is Schur-convex on* $\mathbb{Z}^m$.

Proof. Consider $f(\mathbf{y}) = \sum_{i=1}^{m} q^{y_i}$, where $\mathbf{y} = \{y_1, ..., y_m\}$ is a vector on $\mathbb{R}^m$. Since the function $f(\mathbf{y})$ is symmetric and has continuous partial derivatives on $\mathbb{R}^m$.

Since $(y_i - y_j)\left(\frac{\partial f}{\partial y_i} - \frac{\partial f}{\partial y_j}\right) = \ln q \cdot (y_i - y_j) \cdot (q^{y_i} - q^{y_j})$ is positive for any $y_i, y_j \in \mathbf{y}$, hence $f(\mathbf{y})$ is Schur-convex on $\mathbb{R}^m$ (Theorem A.4. in [67]). Since $\mathbb{Z}^m \in \mathbb{R}^m$, then $\sum_{i=1}^m q^{x_i}$ is Schur-convex on $\mathbb{Z}^m$. □

Corollary 16. *Let $m, n, j, k \in \mathbb{Z}$, where $1 \leq j \leq k \leq n$ and $1 \leq m \leq n$. Let $q = 1 - k/n$. $\mathbf{x} = \{x_1, x_2, ..., x_m\}$ is a vector on $\mathbb{Z}^m$. Then $\sum_{i=1}^m C_{x_i}^j q^{x_i}$ is Schur-convex on $\mathbb{Z}^m$.*

Proof. Let $a \in \mathbb{Z}$, and $1 \leq j < a < n-2$, it is easy to prove that $C_{a+2}^j - C_{a+1}^j \geq C_{a+1}^j - C_a^j$. Hence, $f_1(x) = C_x^j, x \geq j$ is a convex function. Then $f_1(\mathbf{x}) = \sum_{i=1}^m C_{x_i}^j$ is Schur-convex on $\mathbb{Z}^m$. By Corollary 15 in the Appendix, $f_2(\mathbf{x}) = \sum_{i=1}^m q^{x_i}$ is Schur-convex on $\mathbb{Z}^m$. By Proposition E.1 in [67], $g(\mathbf{x}) = f_1(\mathbf{x}) f_2(\mathbf{x})$ is Schur-convex on $\mathbb{Z}^m$. □

References

[1] A. E. Rhalibi and M. Merabti. **Interest Management and Scalability Issues in P2P MMOG**. In *CCNC '06: Proc. of the 3rd IEEE Consumer Communications and Networking Conference*, pages 1188–1192, 2006. 4, 34

[2] Anthony (Peiqun) Yu and Son T. Vuong. **Mopar: a Mobile Peer-to-Peer Overlay Architecture for Interest Management of Massively Multiplayer Online Games**. In *NOSSDAV '05: International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 99–104, 2005. 4, 23, 34

[3] Roger Zimmermann, Beomjoo Seo, Leslie S. Liu, Rahul S.Hampole, and Brent Nash. **AudioPeer: A Collaborative Distributed Audio Chat System**. In *Distributed Multimedia Systems*, 2004. 4, 24

[4] Xiaohui Gu, Zhen Wen, Philip S. Yu, and Zon-Yin Shae. **PeerTalk: A Peer-to-Peer Multiparty Voice-over-IP System**. In *IEEE Transactions on Parallel and Distributed Systems*, **19**(4):515–528, 2008. 4, 5, 24

[5] F Baumgarte and Christof Faller. **Binaural Cue Coding - Part I: Psychoacoustic Fundamentals and Design Principles**. in *IEEE Transactions on Speech and Audio Processing*, **11**(6):509–519, 2003. 4

[6] Gale Huang. **Keynote: Experiences with PPLive**. In *Peer-to-Peer Streaming and IP-TV Workshop*, page 18, 2007. 5

[7] Kunwadee Sripanidkulchai, Aditya Ganjam, Bruce Maggs, and Hui Zhang. **The Feasibility of Supporting Large-scale Live Streaming Applications with Dynamic Application End-points**. In *SIGCOMM '04*, pages 107–120. 2004. 5

[8] Yang Chu, Sanjay Rao, Srinivasan Seshan, and Hui Zhang. **Enabling Conferencing Applications on the Internet Using an Overlay Muilticast Architecture**. In *SIGCOMM Computer Communication Review*, **31**(4):55–67, 2001. 5

[9] Miguel Castro, Michael B. Jones, Anne-Marie Kermarrec, Antony Rowstron, Marvin Theimer, Helen Wang, and Alec Wolman. **An Evaluation of Scalable Application-Level Multicast Built Using Peer-to-Peer Overlays**. In *INFOCOM '03*, 2003. 5, 18, 20, 24

[10] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. **Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications**. In *INFOCOM '03*, **2**:1521–1531, 2003. 5, 30

[11] Iain A. Oliver, Alan H.D. Miller, and Colin Allison. **Virtual Worlds, Real Traffic: Interaction and Adaptation**. In *MMSys '10: the 1st Annual ACM SIGMM Conference on Multimedia Systems*, 2010. 6

[12] Second Life. **Online: http://secondlife.com**. 6, 103

[13] ActiveWorlds. **Online: http://www.activeworlds.com**. 6

[14] The Second Life Economy. **Online: http://blogs.secondlife.com/tags/quarterly_economic_report**. 7

[15] Richard T. B. Ma, Sam C. M. Lee, John C. S. Lui, and David K. Y. Yau. **A Game Theoretic Approach to Provide Incentive and Service Differentiation in P2P Networks**. In *SIGMETRICS '04: the International Conference on Measurement and Modeling of Computer Systems*, pages 189–198, 2004. 8

[16] ITU-T Recommendation. **G.114 One-way Transmission Time**, 2003. 9, 97

[17] Shao Liu, Rui Zhang-Shen, Wenjie Jiang, Jennifer Rexford, and Mung Chiang. **Performance Bounds for Peer-Assisted Live Streaming**. In *SIGMETRICS '08: the International Conference on Measurement and Modeling of Computer Systems*, pages 313–324, 2008. 10, 18

[18] N. Magharei and R. Rejaie. **PRIME: Peer-to-Peer Receiver-drIven MEsh-Based Streaming**. In *INFOCOM '07*, pages 1415 –1423, 2007. 10, 18

[19] Yan Huang, Tom Z.J. Fu, Dah-Ming Chiu, John C.S. Lui, and Cheng Huang. **Challenges, Design and Analysis of a Large-scale P2P-VOD System**. In *SIGCOMM '08*, 2008. 10, 18

[20] Jiajun Wang, Cheng Huang, and Jin Li. **On ISP-friendly Rate Allocation for Peer-assisted VoD**. In *MM '08: the 16th ACM International Conference on Multimedia*, pages 279–288, 2008. 10, 18

[21] Roger Zimmermann and Ke Liang. **Spatialized Audio Streaming for Networked Virtual Environments**. In *MM '08: the 16th ACM International Conference on Multimedia*, pages 299–308, 2008. 11, 26, 30, 47, 57, 89

[22] Ke Liang and Roger Zimmermann. **Cross-Tree Adjustment for Spatialized Audio Streaming over Networked Virtual Environments**. In *NOSSDAV '09: the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 73–78, 2009. 11, 30, 57, 89, 91

[23] The Gnutella v0.6 Protocol. **Online: http://rfc-gnutella.sourceforge.net/src/rfc0_6draft.html**. 16

[24] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. **Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications**. In *SIGCOMM Computer Communication Review*, **31**:149–160, 2001. 17

REFERENCES

[25] Rowstron, Antony I. T. and Druschel, Peter. **Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems**. In *Middleware '01: the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350, 2001. 17, 18, 20

[26] Gummadi, P. Krishna and Saroiu, Stefan and Gribble, Steven D. **A Measurement Study of Napster and Gnutella as Examples of Peer-to-Peer File Sharing Systems**. In *SIGCOMM Computer Communication Review*, **32**:82–82, 2002. 18

[27] Castro, M. and Druschel, P. and Kermarrec, A.-M. and Rowstron, A.I.T. **Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure**. In *IEEE Journal on Selected Areas in Communications*, **20**(8):1489 – 1499, oct 2002. 19

[28] Kostic, D. and Rodriguez, A. and Albrecht, J. and Vahdat, A. **Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh**. In *SOSP'03: the* 19^{th} *ACM Symposium on Operating System Principles*, 2003. 22

[29] Kostic, D. and Rodriguez, A. and Albrecht, J. and Abhijeet, B. and Vahdat, A. **Using Random Subsets to Build Scalable Network Services**. In *USITS'03: the* 4^{th} *USENIX Symposium on Internet Technologies and Systems (USITS)*, 2003. 22

[30] Marios Assiotis and Velin Tzanov. **A Distributed Architecture for MMORPG**. In *NetGames '06: the* 5^{th} *ACM SIGCOMM Workshop on Network and System Support for Games*, page 4, 2006. 23

[31] Fábio Reis Cecin, Rafael de Oliveira Jannone, Cláudio Fernando Resin Geyer, Márcio Garcia Martins, and Jorge Luis Victoria Barbosa. **FreeMMG: a Hybrid Peer-to-Peer and Client-Server Model for Massively Multiplayer Games**. In *NetGames '04: the* 3^{rd} *ACM SIGCOMM Workshop on Network and System Support for Games*, pages 172–172, 2004. 23

[32] Alvin Chen and Richard R. Muntz. **Peer Clustering: a Hybrid Approach to Distributed Virtual Environments**. In *NetGames '06: the* 5^{th} *ACM SIGCOMM Workshop on Network and System Support for Games*, page 11, 2006. 23

[33] Thorsten Hampel, Thomas Bopp, and Robert Hinn. **A Peer-to-Peer Architecture for Massive Multiplayer Online Games**. In *NetGames '06: the* 5^{th} *ACM SIGCOMM Workshop on Network and System Support for Games*, page 48, 2006. 23

[34] Shun-Yun Hu and Guan-Ming Liao. **Scalable Peer-to-Peer Networked Virtual Environment**. In *NetGames '04: the* 3^{rd} *ACM SIGCOMM Workshop on Network and System Support for Games*, pages 129–133, 2004. 23

[35] Leslie Liu and Roger Zimmermann. **Adaptive Low-latency Peer-to-Peer Streaming and its Application**. In *Multimedia Systems*, **11**:497–512, 2006. 23

[36] Shinya Yamamoto, Yoshihiro Murata, Keiichi Yasumoto, and Minoru Ito. **A Distributed Event Delivery Method With Load Balancing for MMORPG**. In *NetGames '05: the* 4^{th} *ACM SIGCOMM Workshop on Network and System Support for Games*, pages 1–8, 2005. 23

[37] Paul Boustead, Farzad Safaei, and Mehran Dowlatshahi. **DICE: Internet Delivery of Immersive Voice Communication for Crowded Virtual Spaces**. In *VR '05: the IEEE Virtual Reality*, pages 35–41, 2005. 24, 25

[38] Leslie S. Liu, Rahul Hampole, Beomjoo Seo, and Roger Zimmermann. **ACTIVE: A Low Latency P2P Live Streaming Architecture**. In *MMCN '05: the SPIE/IS&T Conference on Multimedia Computing and Networking* , 2005. 24

[39] Paul Boustead and Farzad Safaei. **Comparison of Delivery Architectures for Immersive Audio in Crowded Networked Games**. In *NOSSDAV '04: the* 14^{th} *ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 22–27, 2004. 25, 26, 126

[40] C.D. Nguyen, F. Safaei, and P. Boustead. **Performance Evaluation of a Proxy System for Providing Immersive Audio Communication to Massively Multi-Player Games**. In *Globecom '04: the IEEE Global Telecommunications Conference Workshops*, pages 192–199, 2004. 25

[41] John R. Douceur, Jay R. Lorch, and Thomas Moscibroda. **Maximizing Total Upload in Latency-Sensitive P2P Applications**. In *SPAA '07: the 19th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 270–279, 2007. 26, 57, 89

[42] Shun-Yun Hu, Ting-Hao Huang, Shao-Chen Chang, Wei-Lun Sung, Jehn-Ruey Jiang, and Bing-Yu Chen. **FLoD: A Framework for Peer-to-Peer 3D Streaming**. In *INFOCOM '08*, pages 1373 –1381, 2008. 27, 115

[43] Matteo Varvello, Christophe Diot, and Ernst Biersack. **P2P Second Life: Experimental Validation Using Kad**. In *INFOCOM '09*, pages 1161–1169, 2009. 27

[44] Chien-Hao Chien, Shun-Yun Hu, and Jehn-Ruey Jiang. **Bandwidth-Aware Peer-to-Peer 3D Streaming**. In *NETGAMES '09: the* 8^{th} *Annual Workshop on Network and Systems Support for Games*, 2009. 27, 115

[45] Petar Maymounkov and David Mazières. **Kademlia: A Peer-to-Peer Information System Based on the XOR Metric**. In *IPTPS '01: the* 1^{st} *International Workshop on Peer-to-Peer Systems*, pages 53–65, 2002. 27

[46] Eyal Even-Dar and Yishay Mansour. **Fast Convergence of Selfish Rerouting**. In *SODA '05: the* 16^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 772–781, 2005. 28, 60, 65, 76

[47] Steve Chien and Alistair Sinclair. **Convergence to Approximate Nash Equilibria in Congestion Games**. In *SODA '07: the* 18^{th} *Annual ACM-SIAM Symposium on Discrete algorithms*, pages 169–178, 2007. 28, 59

[48] Paul W. Goldberg. **Bounds for the Convergence Rate of Randomized Local Search in a Multiplayer Load-balancing Game**. In *PODC '04: the* 23^{rd} *Annual ACM Symposium on Principles of Distributed Computing*, pages 131–140, 2004. 28, 59

[49] Petra Berenbrink, Tom Friedetzky, Leslie Ann Goldberg, Paul Goldberg, Zengjian Hu, and Russell Martin. **Distributed Selfish Load Balancing**. In *SODA '06: the* 17^{th} *Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 354–363, 2006. 28, 59, 60, 65, 76

[50] Baruch Awerbuch, Yossi Azar, and Rohit Khandekar. **Fast Load Balancing via Bounded Best Response**. In *SODA '08: the* 19^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 314–322, 2008. 28, 59

[51] Heiner Ackermann, Simon Fischer, Martin Hoefer, and Marcel Schöngens. **Distributed Algorithms for QoS Load Balancing**. In *SPAA '09: the* 21^{st} *Annual Symposium on Parallelism in Algorithms and Architectures*, pages 197–203, 2009. 28, 59

[52] Avrim Blum, Prasad Chalasani, Don Coppersmith, Bill Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. **The Minimum Latency Problem**. In *STOC '94: the* 26^{th} *Annual ACM Symposium on Theory of Computing*, pages 163–171. ACM, 1994. 30

[53] S.Y. Shi and J.S. Turner. **Routing in Overlay Multicast Networks**. In *INFOCOM '02*, pages 1200–1208, 2002. 30

[54] Sherlia Y. Shi, Jonathan S. Turner, and Marcel Waldvogel. **Dimensioning Server Access Bandwidth and Multicast Routing in Overlay Networks**. In *NOSSDAV '01: the* 11^{th} *International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 83–91, 2001. 30

[55] Paolo Narváez, Kai-Yeung Siu, and Hong-Yi Tzeng. **New Dynamic Algorithms for Shortest Path Tree Computation**. In *IEEE/ACM Transactions on Networking*, **8**:734–746, 2000. 30

[56] Paolo Narváez, Kai-Yeung Siu, and Hong-Yi Tzeng. **New Dynamic SPT Algorithm Based on a Ball-and-String Model**. In *IEEE/ACM Transactions on Networking*, **9**:706–718, 2001. 30

[57] P. Boustead, F. Safaei, and M. Dowlatshahi. **DICE: Internet Delivery of Immersive Voice Communication for Crowded Virtual Spaces**. *VR '05: the IEEE Virtual Reality*, pages 35–41, March 2005. 31

[58] B. Awerbuch. **Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems**. In *STOC '87: the* 19^{th} *Annual ACM Symposium on Theory of Computing*, pages 230–240, 1987. 38

[59] David Peleg and Vitaly Rubinovich. **A Near-Tight Lower Bound on the Time Complexity of Distributed Minimum-Weight Spanning Tree Construction**. In *SIAM Journal of Computing*, **30**:1427–1442, May 2000. 38

[60] Mohit Singh and Lap Chi Lau. **Approximating Minimum Bounded Degree Spanning Trees to Within One of Optimal**. In *STOC '07: the* 39^{th} *annual ACM Symposium on Theory of Computing*, pages 661–670, 2007. 38

[61] Christian Lavault and Mario Valencia-Pabon. **A Distributed Approximation Algorithm for the Minimum Degree Minimum Weight Spanning Trees**. In *Journal of Parallel and Distributed Computing*, **68**:200–208, 2008. 38

[62] Michael L. Fredman and Robert Endre Tarjan. **Fibonacci heaps and their uses in improved network optimization algorithms**. In *Journal of ACM*, **34**(3):596–615, 1987. 42

[63] Broadband Reports and Speed Test Statistics. **Online: http://www.dslreports.com/archive**. 48, 88, 117

[64] Alexander Kesselman Eyal Even-Dar and Yishay Mansour. **Convergence Time to Nash Equilibria**. In *ICALP '03: the* 30^{th} *International Conference on Automata, Languages and Programming*, pages 502–513, 2003. 58

[65] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007. 60

[66] Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. **The Complexity of Pure Nash Equilibria**. In *STOC '04: the* 36^{th} *Annual ACM Symposium on Theory of Computing*, pages 604–612, 2004. 60, 109

[67] A. W. Marshall and I. Olkin. *Inequalities: Theory of Majorization and Its Applications*, **143** of *Mathematics in Science and Engineering*. Academic Press, 1979. 63, 82, 129, 130

[68] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. **A Survey and Comparison of Peer-to-Peer Overlay Network Schemes**. In *IEEE Communications Surveys and Tutorials*, **7**:72–93, 2005. 77

[69] Bryan Ford, Pyda Srisuresh, and Dan Kegel. **Peer-to-Peer Communication Across Network Address Translators**. In *ATEC '05: the Annual Conference on USENIX Annual Technical Conference*, pages 13–13, 2005. 87

[70] D. Seah, Wai Kay Leong, Qingwei Yang, B. Leong, and A. Razeen. **Peer NAT Proxies for Peer-to-Peer Games**. In *NetGames '09: the* 8^{th} *Annual Workshop on Network and Systems Support for Games*, pages 1 –6, 2009. 87

[71] Bernard Wong, Aleksandrs Slivkins, and Emin Gn Sirer. **Meridian: A Lightweight Network Location Service without Virtual Coordinates**. In *SIGCOMM*, pages 85–96, 2005. 94

[72] Jakob Puchinger, Günther R. Raidl, and Ulrich Pferschy. **The Multidimensional Knapsack Problem: Structure and Algorithms**. In *INFORMS Journal on Computing*, **22**:250–265, 2010. 106

[73] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990. 106

REFERENCES

[74] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., 1986. 106

[75] Niv Buchbinder and Joseph Naor. **The Design of Competitive Online Algorithms via a Primal-Dual Approach**. In *Foundations and Trends in Theoretical Computer Science*, **3**:93–263, 2009. 106, 107

[76] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003. 106

[77] Alessandro Panconesi and Mauro Sozio. **Fast Distributed Scheduling via Primal-Dual**. In *SPAA '08: the* 20^{th} *Annual Symposium on Parallelism in Algorithms and Architectures*, pages 229–235, 2008. 107

[78] Yuval Emek, Magnús M. Halldórsson, Yishay Mansour, Boaz Patt-Shamir, Jaikumar Radhakrishnan, and Dror Rawitz. **Online Set Packing and Competitive Scheduling of Multi-Part Tasks**. In *PODC '10: the* 29^{th} *ACM Symposium on Principles of Distributed Computing*, pages 440–449, 2010. 107

[79] Fischer, Simon and Olbrich, Lars and Vcking, Berthold. **Approximating Wardrop Equilibria with Finitely Many Agents**. In *Distributed Computing*, **21**:129–139, 2008. 115

[80] J. E. Littlewood G. H. Hardy and G. Polya. *Inequalities*. Cambridge University Press, 1988. 129