University of Alberta

Multimedia Transmission over Unreliable Multi-Source Networks

by

Lihang Ying 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of

the requirements for the degree of **Doctor of Philosophy**

Department of Computing Science

Edmonton, Alberta
Fall 2008

Your file Votre référence
ISBN: 978-0-494-46455-7
Our file Notre référence
ISBN: 978-0-494-46455-7

To my beloved wife, Xinxin Gao, and our families.

ABSTRACT

With the wide adoption of the high-speed Internet, multimedia streaming is becoming more and more popular. Very often, a poor Internet connection with low throughput cannot offer acceptable quality in multimedia service. In such a context, multiple connections to the same server, multiple servers or multiple peers are necessary. With the demand for tetherless connectivity, wireless networks such as IEEE 802.11b/g or Bluetooth are becoming both practical and popular. Unlike wired networks, wireless communication suffers from packet loss caused by fading, shadowing and interference.

In this thesis, we address multimedia delivery over unreliable multi-source networks. We improve the bandwidth-monitoring scheme of multi-server retrieval by adopting collaborative retrieval and extend the statistical model by estimating the sum of the bandwidth of servers directly. We propose an Internet P2P Peer-Cached VOD system (pcVOD) and implement a real-world prototype. pcVOD requires peers to provide storage cache space and allows our cache-replacement algorithm to manage the cache. A novel cache-replacement policy based on the ratio of demand and cache is proposed. Based on the observation that the close peers share the routers along the traceroute path to the same destination, we propose a new fast traceroute-based peer pre-selection scheme that narrows the scope of the candidate peers to the close peers without time-consuming probing. To transmit the geometry data of 3D meshes over lossy channels without retransmission and Error Correcting Coding (ECC), we comprehensively study the impacts of packet size, sending rate, sending interval, and buffer size on packet loss through real-world experiments and develop an optimal packet size model to maximize the received payload. We propose the strategy of distributing neighboring vertex information into different packets, as well as interpolation at different neighborhood levels. Due to the arbitrary distribution of the vertices on a 3D mesh, a predefined distance or pattern of interleaving is not effective. We propose probabilistic mesh-dependant interleaving for 3D mesh, which try to avoid the neighboring vertices to be transmitted in the continuous packets.

ACKNOWLEDGMENT

First, I would like to sincerely thank Dr. Anup Basu for his attentive guidance and help during my PhD studies. He always encouraged me and increased my confidence. With his broad background in theory and his rich research experience, he always inspired me and directed me towards creative research. I also thank Dr. Irene Cheng, who gave me timely help whenever I faced difficulties and always provided detailed suggestions and guidance during my research.

I would also like to thank my supervisory committee, Dr. Walter Bischof and Dr. Paul Lu, and my examining committee, Dr. Norman Beaulieu, ECE, University of Alberta, and Dr. Avideh Zakhor, EECS, Berkeley for their careful reading and valuable comments on my thesis.

I also thank all my friends in the department, especially Tao Wang, Fan Deng, Shoudong Zou and Xin Li, for their help and valuable conversations.

I express my heartfelt gratefulness to my wife, Xinxin Gao, and our families. Xinxin faced every difficulty together with me, encouraged me to persist, and shared every moment of my happiness. Our families provided me endless support and were always the source of my ability to achieve success.

Table of Contents

List of Tables

List of Figures

GLOSSARY

3DMC 3D Mesh Coding.

C/S Client/Server.

CDN Content Delivery Network.

DHT Distributed Hash Table.

EB Effective Bandwidth.

EC Effective Capacity.

ECC Error Correction Codes.

ISP Internet Service Providers.

MDC Multiple Description Coding.

MPEG Moving Pictures Expert Group.

P2P Peer-to-Peer.

pcVOD Peer Cached VOD.

QoS Quality of Service.

RSVP Resource reSerVation Protocol.

SNR Signal-to-Noise Ratio.

RTT Round-Trip Time.

UPS United Parcel Service.

VCI Video Cache Information.

VOD Video-On-Demand.

WLAN Wireless LAN.

Chapter 1

INTRODUCTION

1.1 Overview of the Thesis

With the wide adoption of the high-speed Internet, multimedia streaming is becoming more and more popular. The Internet is a heterogeneous and best-effort network, and the quality of Internet connections varies dramatically. One connection of multimedia streaming consumes much more bandwidth than traditional text-based messaging, and many multimedia applications require Quality of Service (QoS). For example, a picture has to be retrieved and displayed before a user gives up. The initial buffering time should be as short as possible in Video-On-Demand (VOD). When transmitting multimedia objects over the Internet, a poor Internet connection with low throughput often cannot offer a quality multimedia service. In this situation, multiple connections are considered. We can set up multiple connections by using multiple connections to the same server (Client to Server, C/S) [137], multiple connections to multiple servers (Peer to Server Peers, P2SP) [78][138], or multiple connections to multiple peers (Peer to Peers, P2P) (Figure 1.1).

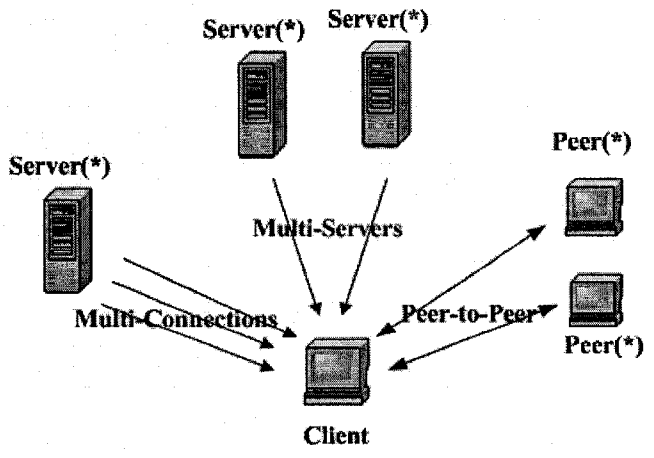


Figure 1.1: Multiple connections to the same server, multiple connections to multiple servers, and multiple connections to multiple peers.

With the demand for tetherless connectivity, wireless networks such as IEEE 802.11b/g or Bluetooth have become practical and popular. Currently, most laptops are equipped with IEEE 802.11b/g. More and more devices such as cell phones, PDA and laptops support Bluetooth. Unlike wired networks, wireless communication experiences packet loss due to fading and interference. To cover a larger service area and support more user capacity and traffic, we need to consider packet loss when transmitting multimedia objects over wireless networks.

In this thesis, we address multimedia delivery over unreliable multi-source networks. To transmit multimedia objects over the best-effort bandwidth-limited Internet, we apply the multi-server and P2P architecture. We also consider transmitting multimedia objects over lossy channels without retransmission and Error Correction Codes (ECC). Figure 1.2 shows the framework of the different components addressed in this thesis.

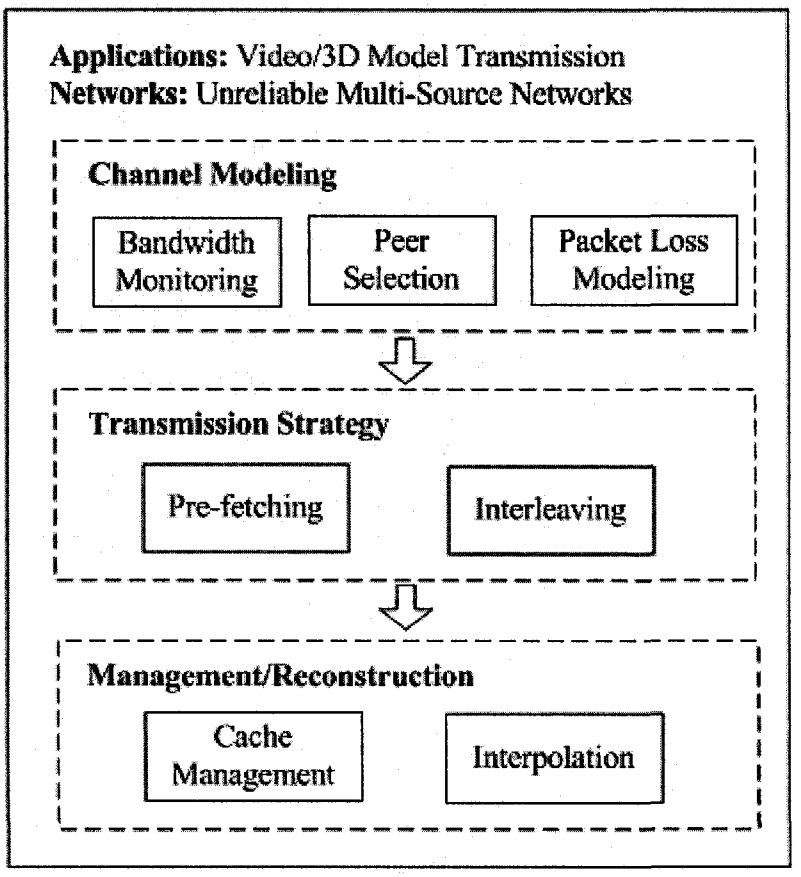


Figure 1.2: Framework of the different components.

Accurate bandwidth monitoring is critical for multimedia delivery applications that guarantee QoS parameters like the time limit for transmission. We assume the purpose of the potential application is to deliver a best-quality multimedia object from a number of distributed servers to a client program within a time limit T. To estimate the size of the best-quality object, the client first uses a fraction (t) of T to perform bandwidth testing. By using the student's t-distribution, the real bandwidth is estimated based on the samples within time t. According to statistical theory, with more samples, i.e., more time used to perform bandwidth testing, the estimated bandwidth is more accurate and larger. However, if more time is used to estimate a more accurate and larger bandwidth, then less time can be used to transfer real data. Therefore, a tradeoff exists, and we can determine the optimal t to achieve the largest expected size for transferring real multimedia object data. We also consider a retrieval strategy where the different servers collaborate in the retrieval process; i.e., a server that finishes its share of the retrieval task will help with the retrieval tasks of other servers. Given this "collaborative" retrieval scenario, we develop a new optimal model to determine the optimal t and the largest expected size.

Video-On-Demand (VOD), such as YouTube-like online video sharing, online theatres and online news videos, is becoming increasingly popular on the Internet. VOD consumes a large bandwidth, and the cost of traditional Client/Server (C/S) solutions is expensive and increases proportionally with increasing concurrent demands. Owing to its advantage of cheaply solving the server-side bottleneck, Peer-to-Peer (P2P) becomes useful in this situation. We propose to use the Peer Cached VOD (pcVOD), an Internet P2P VOD with storage caching on peers. First, instead of depending on a user to manually decide how much cache space is shared and which video is to be kept for sharing, our system requires peers to provide storage cache space and allows our cache replacement algorithm to manage the cache. The cache-replacement strategy is based on the ratio of the number of peers demanding to the number of peers with a cache, and is unique in the P2P VOD environment. Second, instead of assigning tasks to peers in equal-sized blocks, we assign as much as possible to one peer and pre-fetch at a time while guaranteeing that the data for the current rendering are ready by monitoring the bandwidth of the connections to each

peer. In simulations, pcVOD shows a great improvement compared to P2P VOD without cache management. The optimum performance can be achieved for a reasonably small cache size. Through mathematical analysis and a case study, we show that pre-fetching is useful when the video size and the retrieving bandwidth are both large.

The extreme heterogeneity in the P2P Internet environment causes the connections to the candidate peers to vary significantly. Thus, selecting good candidate peers is critical to P2P networking performance. The Internet infrastructure environment in China is much more complex than in developed countries. The quality of links within China varies greatly. Furthermore, the links between different Internet Service Providers (ISP) are much slower than the links within the same ISP. To improve P2P applications in a networked environment similar to that in China and reduce the traffic between different ISPs and different locations, peers from different ISPs and different locations must be distinguished. While most research in the literature focus on selecting peers with low latency, high uploading bandwidth, and high serving stability by time-consuming end-to-end measurements, we address how to return good candidate peers from the peer list server quickly and efficiently without probing. We propose to pick out the close peers from the same location or with low Round-Trip Time (RTT) from each other, by observing that the close peers share the routers along the traceroute path to the same destination. Our method maintains only the online peers' traceroute records, without any offline database. Oriented to P2P VOD application, a tracker-based architecture is used in order to shorten the initial buffering time. Trackers keep only the online peers' traceroute records. Based on experiments with real traceroute records from 352 different IP addresses around the world, our hop-based peer selection algorithm with maximal hop 6 can effectively select peers from the same city and same ISP without having significant false alarms. Our RTT-based algorithm with a maximum RTT of 150ms can be used to select more peers when not enough peers are available from the same city and same ISP.

Unlike wired communication, wireless communication has two challenging aspects. The first is the fading phenomenon, which includes small-scale multi-path fading and larger-scale fading such as path loss via distance attenuation and shadowing by

obstacles. The second is interference, which can be between transmitters communicating with a common receiver, between multiple receivers communicating with a single transmitter, or between different transmitter–receiver pairs. We carry out a comprehensive study of the impacts of packet size, sending rate, sending interval, and buffer size on packet loss through real-world experiments in a competing WLAN environment, where packet losses and congestion occur. When competing connections are present, both too small and too large a packet size increases the packet loss. The proper packet size can be found to minimize the packet loss. We propose an approach for estimating the optimal packet size.

3D transmission over unreliable wireless networks needs to take into account the possibility of packet loss. One of the major drawbacks with most 3D transmission algorithms is that they do not consider the possibility of packet loss over unreliable networks. In some studies, it is assumed that some parts of the mesh can be transmitted without loss, allowing progressive mesh transmission to give good results. However, for a lossy channel, this assumption implies retransmission, which is not feasible in an environment, such as multicast, real-time or time constraint applications, where retransmission is not possible or preferred. The adaptive scheme of error correction codes (ECC) could be applied to correct errors in different error rates. ECC schemes require additional bits to correct errors, and this requirement might result in significant overhead if the error rate is high. To transmit the geometry data of 3D meshes over a lossy channel without retransmission and ECC, we assume packet-based transmission where a certain percentage of the packets may be lost. We propose the strategy of distributing neighboring vertex information into different packets to minimize the risk of packet loss affecting a large neighborhood, and an interpolation scheme with different neighbor levels in the order of decoding. Experiments on models with different densities show that smoothness on the mesh's surface deteriorates only at a packet loss larger than 75%, and that the object is still recognizable. The reconstruction quality after transmission with packet loss depends on the original density of the model. If a projecting vertex is lost, the corresponding part will be flattened. We also compare the triangle-based linear spline, triangle-based cubic spline and 'v4' interpolation methods. The triangle-based cubic spline

interpolation method performs best overall. Compared to the subdivision-based approach, our interpolation scheme performs significantly better.

In many communication situations, the losses may be bursty. For example, communication over the Internet due to congestion has burst packet loss. Communication over wireless networks is expected to experience burst packet losses due to either temporary link outages or fading-induced bit errors. By converting burst errors into random-like errors, interleaving becomes an efficient way to combat burst errors. The existing interleaving schemes for 1D sequences and 2D images are not suitable for encoded 3D meshes because the distance in the encoded stream of neighboring vertices in 3D meshes varies. We propose the use of probabilistic mesh-dependant interleaving to minimize the risk of burst packet loss affecting a large neighborhood. The proposed scheme adjusts the interleaving parameters based on the statistical property of a 3D mesh model and different burst-length situations. By considering the majority of distances between neighboring vertices, we try to avoid neighboring vertices within the longest burst. We verify the proposed scheme over simulated Gilbert channels with different packet loss levels and burst lengths.

1.2 Contributions of the Thesis

This section summarizes the contributions of this thesis:

1) For bandwidth monitoring of multi-server retrieval, this thesis improves the bandwidth-monitoring scheme in [58] by adopting collaborative retrieval and extends the statistical model by estimating the sum of the bandwidth of servers directly [74]. In a previous study [58], the client requests each server to deliver only its own part of the object. When some servers finish delivering their own segments earlier than others, they idle and waste their remaining bandwidth, even though the transmission of the whole object is not finished. The essence of the statistical model of [58] is to estimate K channels individually, then to sum up the respective estimated result of each channel. Each channel should satisfy the respective estimation independently. This requirement is stricter than needing to satisfy only the estimated sum of the K channels. Our improved collaborative retrieval strategy utilizes all the channels better than the

previous strategy [58] does, and the object size estimated by our extended statistical model which directly estimates the sum of the bandwidth of the servers, is larger than the object size estimated by the previous model [58]. Additionally, in [58], some unreliable channels are dropped to achieve the better-estimated size. We extend the idea by introducing a reliability factor that can vary between 0 and 1, compared to only 0 or 1 in [58].

2) We propose an Internet P2P Peer-Cached VOD system (pcVOD) [105] and commercialize a real-world prototype with $1 million USD venture capital. In order to better support the demands of less popular videos, we describe a new scheme that requires peers to provide storage cache space and allows our cache replacement algorithm to manage the cache instead of traditionally depending on a user to manually decide how much cache space is shared and which video is to be kept for sharing. A novel cache-replacement policy based on the ratio of the number of peers demanding to the number of peers with a cache, which is unique to the P2P environment, is proposed. In simulations, pcVOD shows a significant improvement compared to P2P VOD, in terms of sharing storage managed by users. The optimum performance can be achieved for a reasonably small cache size.

3) Most research in the literature focuses on selecting peers with low latency, high uploading bandwidth, and high serving stability by making time-consuming end-to-end measurements. In contrasts, this thesis proposes a new quickly peer pre-selection scheme that narrows the scope of the candidate peers to the close peers from the same location or with low Round-Trip Time (RTT) from each other [140]. Our fast peer pre-selection scheme is based on the observation that the close peers share the routers along the traceroute path to the same destination. By our method, peers from different ISPs and different locations are distinguished in order to fast pre-select peers in a networked environment where the quality of links varies greatly and to reduce the traffic between different ISPs and different locations. Through the real traceroute records collected by our real-world pcVOD prototype, we obtain the proper parameters and verify the efficiency of our method.

4) The present study differs from previous ones on packet loss in wireless networks by carrying out a comprehensive study of the impacts of packet size, sending rate, sending

interval, and buffer size on packet loss through real-world experiments in a competing WLAN environment [139][141]. Unlike several published studies on packet size optimization based on different metrics, such as throughput, goodput, transmission range, and energy efficiency, this present study proposes a mathematical model for estimating the optimal packet size to maximize the expected payload.

5) Most 3D transmission algorithms do not consider the possibility of packet loss over unreliable networks. The existing research uses either retransmission or ECC to deal with packet loss. This thesis proposes to transmit and reconstruct a 3D mesh model over a lossy channel without retransmission and ECC [53] since retransmission is not applicable or preferred in real-time applications, and ECC results in significant overhead if the error rate is high. Our strategy distributes neighboring vertex information into different packets to minimize the risk of packet loss affecting a large neighborhood. With this strategy, the lost vertices are distributed evenly over the model and can be better constructed by interpolation. Experiments on models with different densities show that smoothness on the mesh's surface deteriorates only at a packet loss larger than 75%, and that the object is still recognizable.

6) Current interleaving schemes to combat bursty loss are suitable for 1D and 2D data elements that are arranged systematically, such as in rows and columns, so that neighboring elements can be identified based on their traversal order and distributed into separate packets. The strategy is to apply a predefined distance or pattern to separate neighboring elements in the transmitted stream independent of which 1D sequences or 2D images are being transmitted. Due to the arbitrary distribution of the vertices on a 3D mesh, a predefined distance or pattern is not effective because it may work on one geometry while fail on another. We introduce a new probabilistic mesh-dependent interleaving algorithm for 3D mesh transmission with bursty packet loss [142], which avoids neighboring vertices to be transmitted in continuous packets by adjusting a control parameter for different mesh geometries (different adjacent distance histograms) and takes the maximum burst length into consideration.

1.3 Thesis Organization

This thesis is organized as follows.

In Chapter 2, related work on 3D mesh coding, packet loss in wireless networks, and peer-to-peer networking is reviewed. The reviewed work on 3D mesh coding includes work on valence-driven 3D mesh coding, progressive coding, transmission in the MPEG-4 standard, and error-resilient transmission. The reviewed work on P2P networking includes work comparing C/S and P2P, and studying three generations of P2P architecture, peer selection, BitTorrent P2P file sharing, and P2P VOD.

In Chapter 3, we present our work on multi-server optimal bandwidth monitoring for collaborative distributed retrieval.

In Chapter 4, pcVOD, an Internet peer-to-peer video-on-demand system with storage caching on peers, is described.

In Chapter 5, we present traceroute-based fast peer pre-selection without an offline database.

In Chapter 6, we present our work on wireless packet loss and modeling.

In Chapter 7, the transmission, interpolation, and reconstruction of arbitrary 3D mesh models over unreliable networks are described.

In Chapter 8, probabilistic mesh-dependant interleaving for compressed 3D meshes transmission over a burst packet loss channel is presented.

In Chapter 9, the conclusions and possibilities for future work are presented.

Chapter 2

3D MESH CODING, WIRELESS PACKET LOSS AND PEER-TO-PEER NETWORKING

2.1 3D Mesh Coding

3D meshes are widely used in visualization applications, including applications in manufacturing, architecture, medical imaging, the military, geographic information systems, and entertainment. 3D meshes are represented by geometry and connectivity [43]. An uncompressed representation, such as the VRML ASCII format [46], is inefficient for transmission. In general, 3D mesh compression schemes handle geometry data by following three steps: quantization, prediction, and statistical coding. However, these schemes differ from each other in terms of connectivity compression.

Among the many 3D mesh compression schemes proposed since the early 1990s [38], the valence-driven approach [44] is considered to be the state-of-the-art technique for 3D mesh compression, with an average compression ratio of 1.5 bits per vertex to encode mesh connectivity. However, this approach is restricted to manifolds. Interested readers can refer to [38] for a literature review on 3D mesh compression. A number of 3D mesh compression algorithms have been accepted as international standards. For example, Topological Surgery [45] and Progressive Forest Split [44] have been adopted by the VRML version 2 [46] and MPEG-4 version 2, defined as 3D Mesh Coding (3DMC) [35].

In this section, we will review valence-driven 3D mesh coding, topological surgery 3D mesh coding, progressive forest split coding, transmission in MPEG-4 3DMC, and error resilient mesh transmission.

2.1.1 Valence-Driven Approach

Touma and Gotsman proposed a valence-driven algorithm to compress 3D meshes [44], which was then enhanced in [19]. The algorithm begins by randomly selecting a triangle. Starting from a vertex of that triangle and traversing all the edges in a counter-clockwise direction (Figure 2.1), the visited vertices are pushed into an active list. After visiting the associated edges, the next vertex is popped from the active list, and the process is repeated. The valence (or degree) of each processed vertex is output. Sometimes, we need to split the current active list or merge it with another active list, which is encoded with special codes. If the meshes are not closed, a dummy vertex is added before encoding for each boundary loop and connected to all the vertices in this boundary loop, making the meshes closed [38]. From the stream of vertex valences, the original connectivity can be reconstructed, as shown in Figure 2.2.

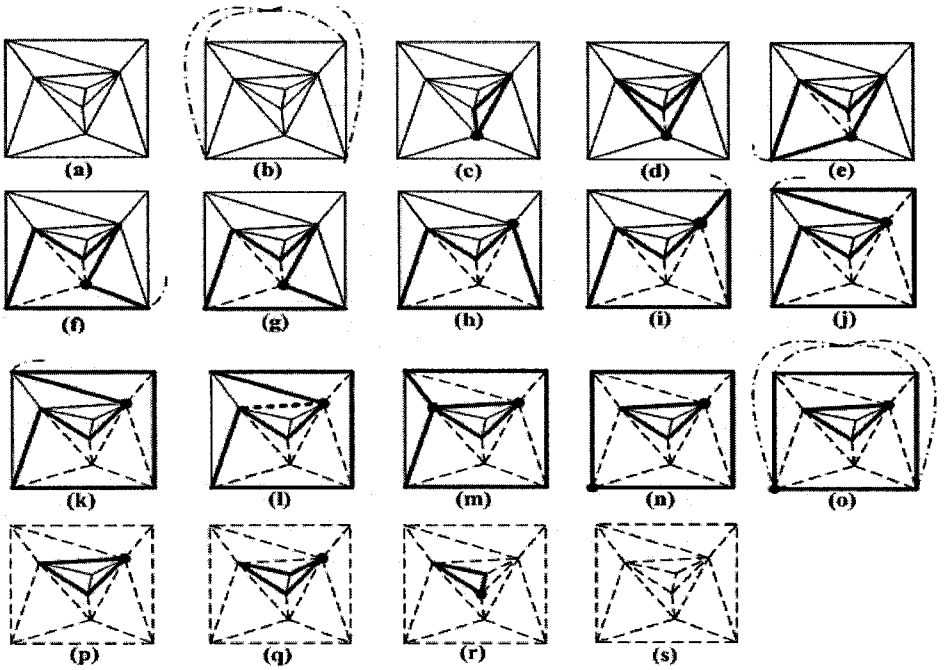


Figure 2.1: An example of a run of the valence-driven connectivity encoding algorithm. The active lists are indicated by thick lines, and edges already visited (encoded) by dashed lines.

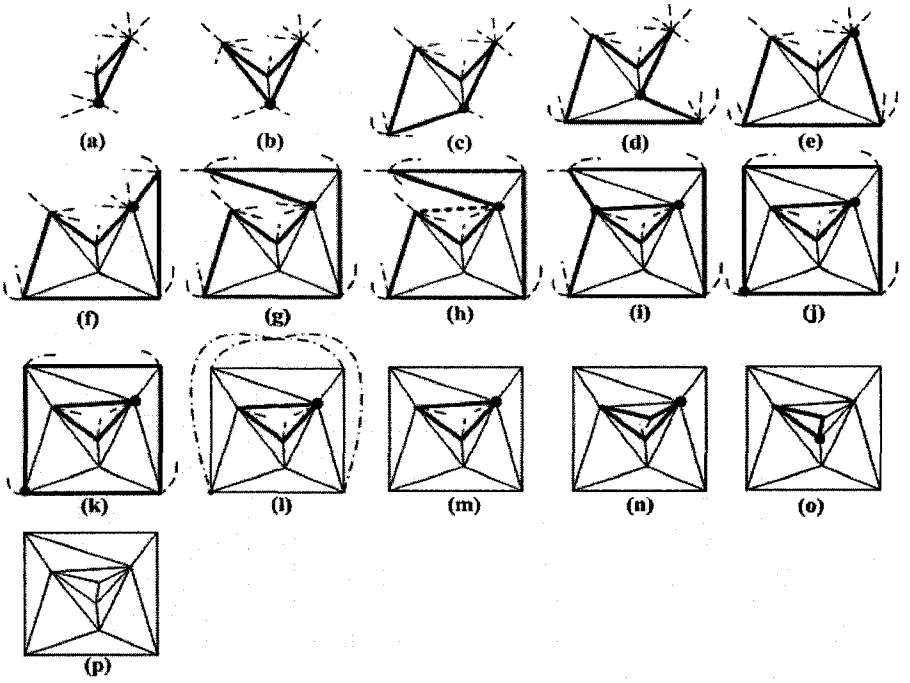


Figure 2.2: An example of connectivity decoding (or reconstructing) from the stream of vertex valences in the valence-driven algorithm.

2.1.2 Topological Surgery

[45] proposed a topological surgery method to compress mesh connectivity. The connectivity is represented by vertex spanning trees and triangle spanning trees. Figure 2.3 shows an example of a vertex spanning tree and a triangle spanning tree for a tetrahedron mesh.

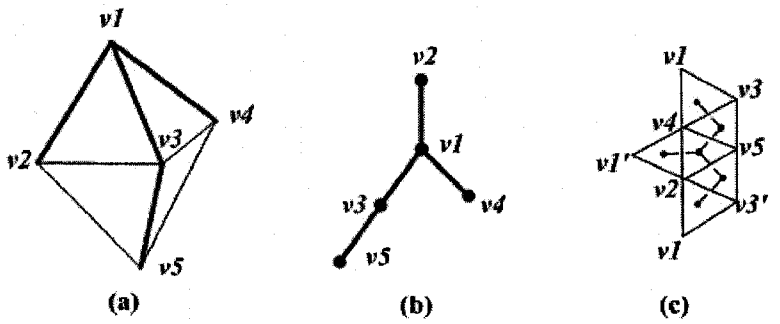


Figure 2.3: An example of a vertex spanning tree and a triangle spanning tree for a tetrahedron mesh: (a) A tetrahedron mesh; (b) The vertex spanning tree; (c) The cut and flattened mesh with its triangle spanning tree depicted by dashed lines.

The vertex and triangle spanning trees are run-length encoded. A run is defined as a tree segment between two nodes with degrees not equal to 2. For each run of the vertex spanning tree, its length, along with two additional flags, is encoded. The first flag is the branching bit indicating whether a run subsequent to the current run starts at the same branching node, and the second flag is the leaf bit indicating whether the current run ends at a leaf node. Similarly, for each run of the triangle spanning tree, its length with the leaf bit is encoded. Furthermore, the marching pattern with one bit per triangle is encoded to indicate how the planar polygon is triangulated internally. The original mesh connectivity can be reconstructed from this set of information.

A run in the vertex or triangle spanning tree is a basic coding unit. The coding cost is proportional to the number of runs, which in turn depends on how the vertex spanning tree is constructed. To maximize the length of each run and minimize the number of runs generated, the vertex spanning tree is built similar to the way "we peel an orange" [45] along a spiral path.

Experimentally, the Topological Surgery algorithm uses 2.48-7.0 bits per vertex for mesh connectivity. The time and space complexities of this algorithm are O(N), where N is the maximum of the vertex number v, the edge number e, and the triangle number f in a mesh. A large memory buffer is needed to allow global random vertex access when decompressing.

2.1.3 Progressive Forest Split

To achieve progressive transmission of 3D meshes, [44] proposed the Progressive Forest Split method. The 3D mesh data are structured into multiple layers: one base layer and one or more enhancement layers. The base layer is encoded by the Topological Surgery algorithm. The enhancement layers use Forest Split operations to refine the model by cutting the existing edges in the forest, creating new triangles in the crevices, and displacing the new vertices to their new positions (Figure 2.4).

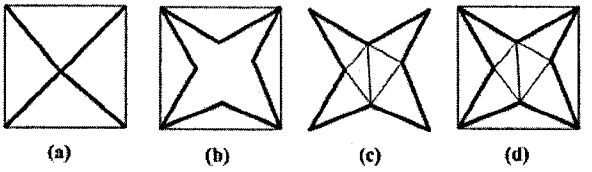


Figure 2.4: An example of the Forest Split operation: (a) A low resolution mesh with a forest depicted by thick lines; (b) cutting the mesh along the forest edges and forming a crevice; (c) triangulation of the crevice; (d) the refined mesh.

Each forest split operation encodes the structure of the forest, the triangulation of the crevices, and the vertex displacements. To encode the structure of the forest, one bit is used for each mesh edge, indicating whether or not it belongs to the forest. To encode the triangulation information of the crevices, the triangle spanning tree and the marching patterns, as in the Topological Surgery algorithm, are used. To encode the vertex displacements, the difference between the original vertex position and the new position is Huffman-coded.

To progressively encode a given mesh with four or five Levels of Detail, the Progressive Forest Split algorithm uses about 7-10 bits per vertex for mesh connectivity and 20-40 bits per vertex for geometry, at 6-bit quantization resolution.

2.1.4 Transmission in MPEG-4 3DMC

To transmit encoded 3D meshes, characteristics including incremental rendering, error resilience, and progressive transmission are required. In the 3DMC [37] bitstream, connectivity data are packed separately from the other geometry and photometry data and put into the initial part of the bitstream. This bitstream structure has two advantages. First, it makes incremental rendering possible. Once the decoder extracts the connectivity data, it possesses the full topology. Thus, it can render part of the 3D mesh object, as it begins to decode the geometry data. Secondly, the structure brings error resilience. With this bitstream structure, if the connectivity data are intact, the decoder can still form the 3D mesh structure with some missing geometry or photometry data.

In order to minimize the impact of packet loss on the 3DMC decoded data in an unreliable networking environment, each 3DMC partition can be rendered independently. Without vertex graphs, reconstructing the 3D mesh is impossible. Therefore, a vertex graph, which is partitioned separately and is transmitted with higher priority, is relatively important.

2.1.5 Error Resilient Mesh Transmission

Current 3D mesh coding techniques focus mainly on coding efficiency, i.e., the compression ratio, by transmitting incremental data. This approach is good without packet loss but is vulnerable to channel errors for irregular meshes. Figure 2.5 shows an example of the error sensitivity of the Edgebreaker 3D mesh coding method [31][40]. With one error character in the connectivity stream, the decoded mesh can change significantly and can be impossible to reconstruct.

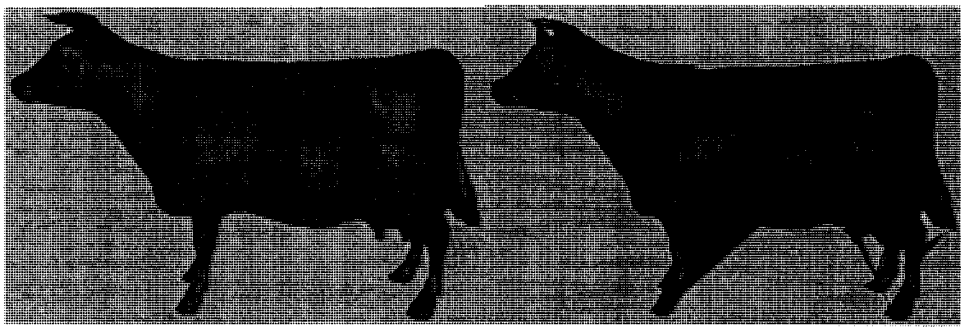


Figure 2.5: An example of error sensitivity of the Edgebreaker 3D mesh coding method. Left: original 3D mesh; right: decoded 3D mesh with one error character in the decoded connectivity stream.

Two approaches can be used to transmit compressed 3D meshes over a lossy network. The first approach is to compress 3D meshes in an error-resilient way. [50] proposed partitioning a mesh into pieces with joint boundaries and encoding each piece independently. However, if packets are lost, the missing pieces create holes in the mesh. [29] introduced multiple descriptions coding for 3D meshes. Each description can be

independently decoded, but this approach assumes the connectivity data are guaranteed to be received correctly. The second approach is to use error protection, such as Error Correction Codes (ECC), to restore lost packets [17] [39].

2.2 Packet Loss in Wireless Networks

With the demand for tetherless connectivity, research has increased in the area of wireless communication [1]. Unlike wired communication, wireless communication has two challenging aspects. The first is the fading phenomenon, which includes small-scale multipath fading and larger-scale fading such as path loss via distance attenuation and shadowing by obstacles. The second is interference, which could be between transmitters communicating with a common receiver, between multiple receivers communicating with a single transmitter, or between different transmitter–receiver pairs.

Some studies on packet loss in wireless networks have been published. [2] conducted the sensor node's field test of the packet loss rate against the distance and transmission power. The results showed that the packet loss rate was increased up to 100% by increasing the distance to 60 meters and decreasing the transmission power to 0. [3] [4] studied the packet loss due to interference between IEEE 802.11b and Bluetooth devices. In the presence of IEEE 802.11b interference with strong signal strength, the percentage of lost UDP packets in Bluetooth transmission can be as high as 70%. [5] [6] described the packet loss probability in an environment with a large number of piconets. (A piconet is an ad hoc network of devices connected by Bluetooth.) With 40 piconets in an area of 20*20 m^2, the packet loss probability could be up to 60%.

The packet loss in a wireless network can be characterized as a burst error. In the early 1960's, Gilbert [7] proposed modeling the burst-error channel by using a Markov chain with states Good and Bad. In state Good, transmission is error-free. In state Bad, a digit is transmitted correctly with only probability h. As well, the probabilities of transiting from state Good to state Bad and state Bad to state Good are p and P. The model with parameters h, p, and P characterizes the channel and can be estimated from easily measured statistics. Elliott [8] extended the Gilbert model by introducing the

probability of loss in state Good. The combination of the Gilbert model and Elliott model is referred to as the Gilbert-Elliott model or the GE model. The GE model has been widely adopted in the study of communication systems at the link, transport and application layers. Frossard and Verscheure [13], and Stuhlmuller et al. [14] proposed a solution for the joint video source-GE channel coding problem for MPEG-2 and H.263, respectively.

Some studies [11][12] pointed out that the GE model is inaccurate for wireless networks. By using a theoretical approach and computer simulations, [11] showed the need to extend the number of Markov states to more than two states in order to characterize a Rayleigh fading channel adequately. [12] claimed that the GE model is a bad approximation of IEEE 802.11b wireless networks due to the existence of heavy-tailed run (gap) and burst length distributions. The same phenomenon in IEEE 802.11g was observed by Carvalho et al. [10], who proposed a logarithmic series distribution model for IEEE 802.11g packet wireless networks with more accuracy in terms of fitting the heavy tail, and with similar complexity to that of the GE model. However, by using the Entropy Nomalized Kullback-Leibler measure to evaluate the performance, Khayam and Radha [9] demonstrated that the traditional two-state Markov chain provided a very suitable model for the IEEE 802.11b MAC-to-MAC packet loss process. However, the two-state Markov chain model is not adequate for bit errors. Khayam and Radha [9] evaluated the full-state, hidden, and hierarchical Markov chain bit error model and found that the full-state Markov chain bit error model rendered the best performance.

Some studies of the packet loss model the channel in terms of connection-level QoS metrics such as the packet loss rate, delay and bandwidth. The concept of effective bandwidth (EB) [15] is introduced to efficiently allocate resources to meet the applications' packet loss and delay requirements. EB generally refers to the minimum amount of network resources (in bits per second) that if allocated to a given traffic flow would guarantee a certain level of QoS (in terms of the packet loss rate or delay constraints). [15] proposed a new approach to partition the range of the received signal-to-noise ratio (SNR) into a set of states, and an algorithm to compute the various parameters of the model, which are then used in deriving closed-form

expressions for the effective bandwidth subject to packet loss and delay constraints. [16] proposed a wireless channel model termed the effective capacity (EC) model. [16] first modeled a wireless channel by using two EC functions: the probability of non-empty buffer and the QoS exponent of a connection. Next, the authors of [16] designed a simple and efficient algorithm to estimate these EC functions. The key advantages of EC channel modeling and estimation are (1) ease of translation into QoS guarantees, such as delay bounds; (2) simplicity of implementation; and (3) accuracy and, hence, efficiency in admission control and resource reservation.

2.3 Peer-to-Peer Networking

The idea of P2P actually existed early in the history of computer networks, such as Instant Message - ICQ, but the concept of P2P was specifically introduced and gained attention after the successful file-sharing system, Napster, had been developed. Substantial studies of P2P architectures and applications have been done in the past 10 years. In this section, we will investigate those studies that are closely related to P2P on-demanding streaming.

2.3.1 Comparison of C/S Architecture and P2P Architecture

In client/server (C/S) architectures, a server or server cluster serves multiple clients. C/S architecture is simple in that clients need to communicate with only one server or server cluster. (A server cluster is normally transparent to clients, who treat it as a server and usually do not notice the existence of a server cluster.) C/S architecture, which must often serve a mass of clusters on the Internet, has several disadvantages when doing so. First, a bottleneck of computing power and bandwidth at the server-side happens when many concurrent requests need to be served. An expensive, powerful, and large-bandwidth server or server cluster must be deployed. Second, C/S architecture has the problems of having single point failure and a single networking path. Service is not available when the server is down. Edge servers are required in order to serve different network areas.

P2P architecture can overcome the disadvantages of C/S architecture. The main characteristic of P2P architecture is that each client (peer) communicates with other

clients directly, so that the clients provide services to one another. A client is usually not as powerful and stable as a server. A client is not always powered on and may have a low bandwidth. As a result, the implementation of P2P architecture is more complex than that of C/S architecture and also has more communications and protocol loads.

Figure 2.6 presents diagrams of C/S and P2P architectures. Table 2.1 outlines their advantages and disadvantages.

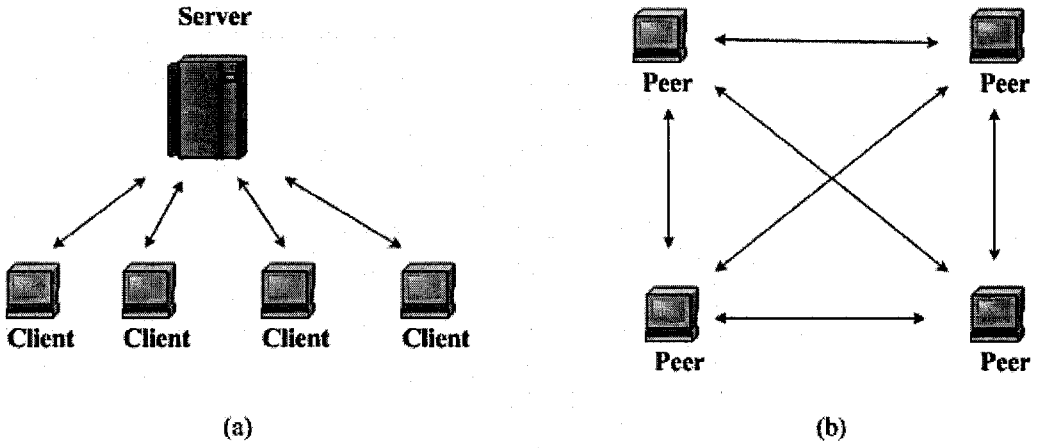


Figure 2.6: Client/Server architecture and peer-to-peer architecture. (a) Client/Server architecture; (b) peer-to-peer Architecture.

Table 2.1: Comparison of C/S architecture and P2P architecture

	C/S architecture	P2P architecture
Advantages	● Simplicity and easy implementation. ● Efficient networking	● Scalability ● Resiliency: a diversity of network paths ● Low delay: peers exchange message directly ● Cheap
Disadvantages	● Bottleneck of computing and bandwidth at server-side. ● Single point failure and single network path ● Expensive	● Complex implementation ● Inefficient networking (starts slower at the beginning and has more protocol load) ● Peers are unstable.

2.3.2 Three Generations of P2P Architecture

P2P architecture can be cataloged into three generations, as Figure 2.7 shows. In the first generation, architectures such as those of Napster, Kazza and ICQ have a centralized server managing the shared resources and connections of the peers and providing a directory. In the second generation, architectures such as that of BitTorrent [68] have no centralized server, but multiple Tracker Servers. Each Tracker Server manages part of the shared resources and the connections of peers related to these resources. The third generation is totally decentralized, without any kind of server. Peers join the system through a list of bootstrap peers, which is available by a variety of means, such as related websites, forums, newsgroups or being introduced by friends. Each peer maintains a list of neighboring sets. Peers are all known to each other by way of the neighbors' neighbors. By using the technique of Distributed Hash Table (DHT), a message about the routing and resource location is efficiently forwarded hop by hop. Along the way of the routed resource location message, the providing peers are located. A peer then contacts the providing peers and retrieves resources from them. [66] proposed a typical example of the decentralized P2P architecture with the DHT technique - Tapestry.

2.3.3 Peer Selection

In P2P networks, usually a peer first retrieves a list of candidate server peers, then selects an active set from the list, and then retrieves data from the peers in the set. During transmission, a list of new candidate server peers is periodically retrieved and inserted into the active set. In the meantime, good active peers are kept, and poor active peers are dropped. Due to the extreme heterogeneity in the P2P Internet environment, the connections to candidate server peers vary greatly. As a result, selecting good candidate server peers is critical to P2P's networking performance. The selection criteria include low-latency, high uploading bandwidth, and serving stability. As well, good candidate server peers must be selected quickly. Most existing studies focus on selecting new peers, keeping good active peers, and dropping poor active peers by probing and using end-to-end measurements during transmission.

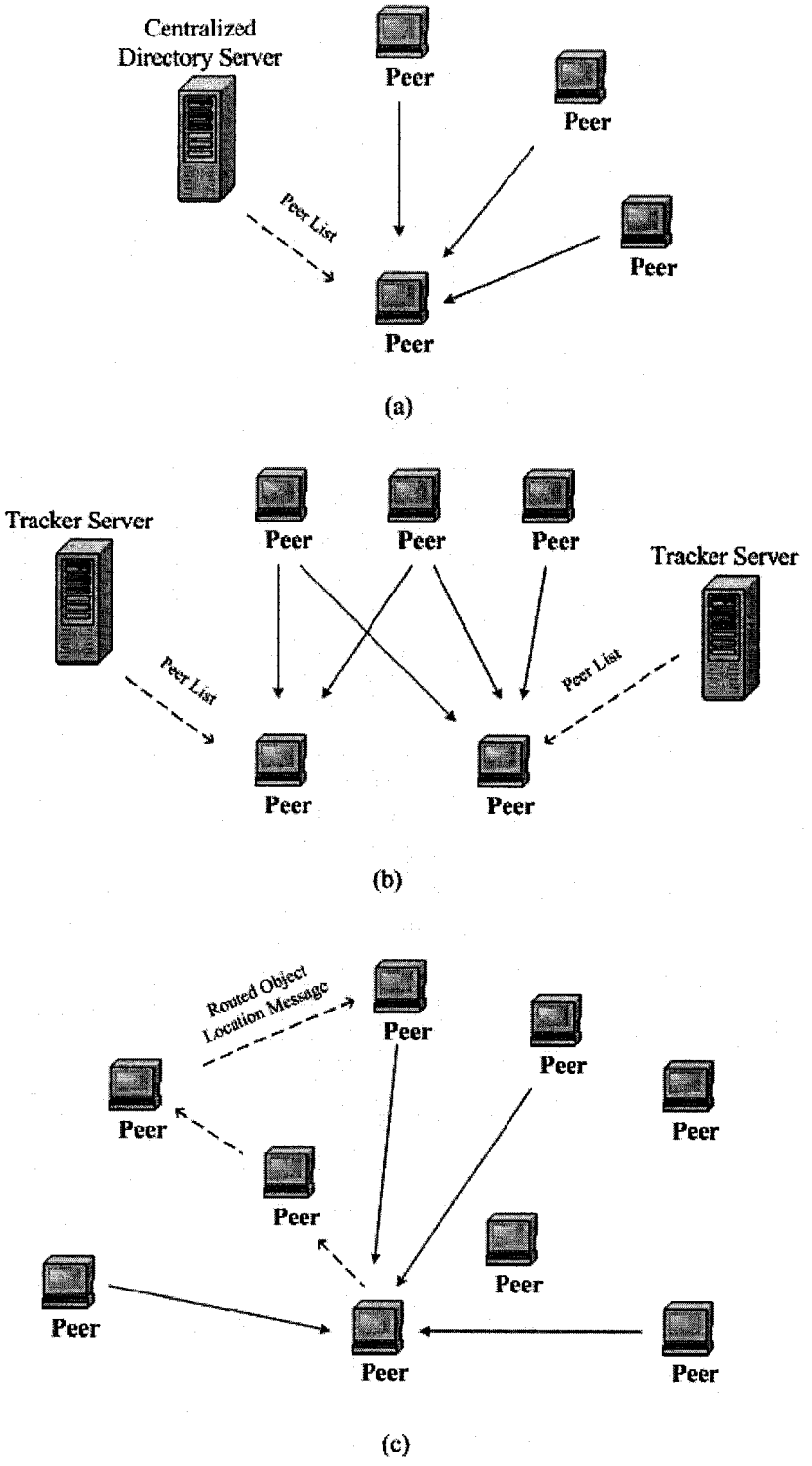


Figure 2.7: Three generations of P2P architecture. (a) First generation – centralized directory server; (b) second generation – multiple tracker servers; (c) third generation – decentralized DHT based.

Selecting peers from the returned candidate peers can be based on random, end-to-end measurements [93] [69] [95], and topology [76] [96]. Candidate peers are randomly picked. By conducting trace-based analyses, Ng et al. [93] investigated end-to-end measurement-based techniques including round-trip time (RTT) probing, 10KB TCP probing, and bottleneck bandwidth probing. These research found that the basic techniques achieved a 40-50% performance improvement, and that these techniques were limited by relying on eliminating the low-performance peers rather than reliably identifying the best-performing ones. The basic techniques can effectively select peers for adaptive applications, which can update the active peers. A combination of the basic techniques can better identify a high-performance peer, so that even applications that cannot adapt may benefit.

Bernstein et al. [69] proposed the use of machine learning for peer selection. A decision tree was used to rate peers based on combinations of collected connection information, such as the load, bandwidth, and past uploading experience. Next, a policy derived by using the Markov Decision Process was executed to select peers. Hefeeda et al. [76] proposed topology-aware peer selection. It infers an approximate topology and considers the shared network path when selecting peers.

Adler et al. [94] conducted research on optimal peer selection in a P2P resource economy, where the server peers charge the client for downloading. For downloading, the optimal selection is to minimize the download delay subject to a budget-cost constraint. For streaming, the optimal selection is to minimize cost subject to continuous playback.

2.3.4 P2P File Sharing

The power of P2P was first recognized through file sharing. The well-known P2P file-sharing systems include Napster, Kazza, eDonkey, Gnutella, and BitTorrent [68]. Among them, BitTorrent is a revolutionary P2P file-sharing system that has become very successful in a short time. The most remarkable character of BitTorrent is that a file is divided into blocks, and downloaded blocks can be uploaded immediately during the downloading process. In the earlier P2P file-sharing systems, peers begin to share a file only after downloading the whole file. In such a system, no collaboration occurs

among the peers downloading the same file. When many downloading peers and limited providing peers are present at the beginning of the sharing period of a popular file, a bottleneck happens, and the downloading peers have to share the limited bandwidth of the providing peers or queue to get access. BitTorrent introduces a scheme in which the downloading peers share downloaded blocks during the downloading process. With the scheme of sharing while downloading and with an algorithm in which the rarely duplicated blocks are downloaded first, the more downloading peers there are, the more uploading bandwidth is provided. BitTorrent allows for collaboration among the peers that are downloading the same file, eliminating the bottleneck problem in the previous peer-to-peer file-sharing systems.

Figure 2.8 shows the BitTorrent architecture, which has no central server. Instead, multiple trackers manage the shared resources and peers' IP and provide a directory.

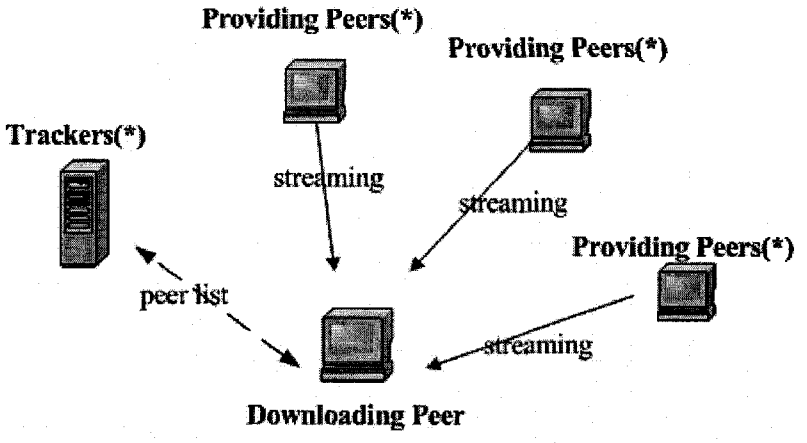


Figure 2.8: BitTorrent Architecture

Qiu and Srikant [86] quantitatively analyzed the BitTorrent-Like P2P file-sharing system by using a fluid model. The evolution of the number of downloaders and seeds is given by

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= \lambda - \theta x(t) - \min\{cx(t), \mu(\eta x(t) + y(t))\}, \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= \min\{cx(t), \mu(\eta x(t) + y(t))\} - \gamma y(t), \end{aligned}$$

where

$x(t)$: number of downloaders at time t.

$y(t)$: number of seeds at time t.

λ: the arrival rate of new requests.

μ: the uploading bandwidth of a peer.

c: the downloading bandwidth of a peer.

θ: the rate at which downloaders abort the download.

γ: the rate at which seeds leave the system.

η: the effectiveness of file sharing.

$cx(t)$ indicates the total downloading bandwidth. $\mu(\eta x(t) + y(t))$ indicates the total downloading rate. $\min\{cx(t), \mu(\eta x(t) + y(t))\}$ expresses the total uploading rate when considering the downloading bandwidth constraint.

By studying the system in steady-state ($\frac{dx(t)}{dt} = \frac{dy(t)}{dt} = 0$), the model explains the behavior of BitTorrent.

2.3.5 P2P VOD

In Video-On-Demand (VOD), each connection consumes a significant bandwidth. The traditional client/server solution cannot support many concurrent demands. P2P is suitable for applications in VOD. Several studies of P2P VOD [76] [71] [79] [81] [82] [85] [87] [88] [89] [90] [91] [92] have been conducted. Most of them are on P2P VOD architecture and system design. [89] proposed a P2P VOD with a cache service at the peers. A peer caches a video and acts as a proxy for other peers in the group. [89] focuses only on the topology where the group of clients is connected via a LAN. Also, it does not consider the instability of the peers, such as the peers that depart. [76] proposed a P2P VOD system: PROMISE. Besides featuring peer selection based on

topology inference, it implements monitoring and reacting to peer failure or degradation.

Quantitative Modeling of P2P VOD

[85] applied the idea of the population growth of biological consecutive generations into modeling a P2P and C/S hybrid VOD system, where peers upload data only after fully retrieving the whole video. In this system, Content Delivery Network (CDN) servers are introduced to disseminate video faster and to serve the demands when no duplications at peers are available at the beginning. Figure 2.9 shows the three stages of the bandwidth usage of the CDN servers. The servers become fully loaded after a short initial stage due to the massive demand. Some requests are rejected until the servers generate enough uploading peers at the server-peer transition time k_0.

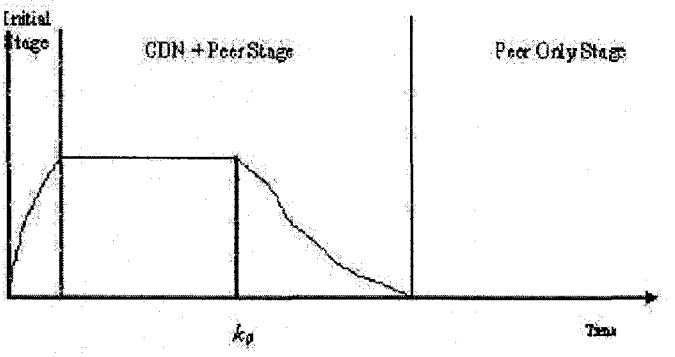


Figure 2.9: The bandwidth usage of the CDN servers in [85]

The basic model is first introduced in the simple situation where only one media file is in the system. The number of uploading peers produced between two consecutive generations k and $k+1$ is expressed as

$$P(k+1) - P(k) = \frac{N}{b} + P(k)\sum_{i=1}^{n} p_i \frac{c_i}{b},$$

where

$P(k)$: number of uploading peers at time k.

N: total server bandwidth.

b: required bandwidth to stream a video.

p_i: percentage of peers in the i-th class.

c_i: contributed uploading bandwidth by a peer of i-th class.

n: number of peer classes.

The model is expanded to a multi-file system with dependence among each file subsystem and the situation with peer failure. Through this model, the exponential growth of the system capacity is verified, and the server-peer transition time k_0 can be calculated.

In the next chapter, we will present our work on multi-server optimal bandwidth monitoring for collaborative distributed retrieval.

Chapter 3

MULTI-SERVER OPTIMAL BANDWIDTH MONITORING FOR COLLABORATIVE DISTRIBUTED RETRIEVAL

Accurate bandwidth monitoring is critical for multimedia delivery applications that guarantee QoS parameters like the time limit for transmission. In this chapter, we consider a retrieval strategy where the different servers collaborate in the retrieval process; i.e., a server that finishes its share of the retrieval task will help with the retrieval tasks of other servers. The potential application is to deliver a best-quality multimedia object from a number of distributed servers to a client program within a time limit T. To estimate the size of the best-quality object, the client first uses a fraction (t) of T to perform bandwidth testing. Because more time is used to estimate a more accurate, less time can be used to transfer real data. Therefore, a tradeoff exists, and we can determine the optimal t to achieve the largest expected size for transferring real multimedia object data. Given this "collaborative" retrieval scenario, we improve the previous multi-server retrieval strategy and extend the statistical bandwidth-monitoring model to determine the optimal t and the largest expected size. By adopting collaborative retrieval, our improved collaborative retrieval strategy utilized all the channels better than the previous method [58] does, and by using our extended statistical model to estimate the sum of the bandwidth of the servers directly, the estimated object size was larger than can be obtained by using the previous model [58]. We extend the idea by introducing a reliability factor that can vary between 0 and 1, compared to only 0 or 1.

3.1 Introduction

Significant research has been conducted in the area of "Quality of Service" (QoS) [61] [63]. For example, [60] and [65] developed the resource reservation protocol RSVP. Its

approach is to allow the receiver of a multimedia stream to reserve certain network resources in advance along the path to the multimedia stream's sender. Unfortunately, several problems are involved in deploying RSVP on a wide scale [64].

An alternative to resource reservation approaches is application-layer network-aware QoS approaches [59]. A critical module in network aware applications is bandwidth monitoring. In order for the application to determine a meaningful media object size, the application needs an accurate estimation of the future bandwidth. In [64], the authors described an algorithm, whose architecture is based on one client and one server, to determine the optimal amount of bandwidth testing. In [58], the authors extended the algorithm to a multi-server environment. They assumed the purpose of the potential application is to deliver a best-quality multimedia object from a number of distributed servers to a client program within a time limit T. To estimate the size of the best-quality object, the client first uses a fraction (t) of T to perform bandwidth testing. By using the student's t-distribution, the real bandwidth is estimated based on the samples within time t. According to statistical theory, with more samples, i.e., more time being used to perform bandwidth testing, the estimated bandwidth is more accurate and larger. However, if more time is used to estimate a more accurate and larger bandwidth, then less time can be used to transfer real data. Therefore, a tradeoff exists, and we can determine the optimal t objective to achieve the largest expected size for transmitting real multimedia object data. In [58], the authors developed an optimal model to determine the optimal t and the largest expected size.

However, in [58], the bandwidth at several servers was monitored independently, and then the most reliable links were chosen to transmit scalable multimedia information. The assumption behind the work in [58] was that each server retrieves a fixed part of a multimedia object, and this part is determined after the bandwidth monitoring process and cannot be changed during the retrieval process. In this present work, we consider a retrieval strategy where the different servers collaborate in the retrieval process; i.e., a server that finishes its share of the retrieval task helps with the retrieval tasks of other servers. Given this "collaborative" retrieval scenario, a new distributed monitoring algorithm is proposed. Through simulations, the new algorithm is demonstrated to have a better performance than that in [58].

The remainder of this chapter is organized as follows: Section 3.2 describes the proposed improvement, the architecture, and the mathematical model and algorithm of the new collaborative retrieval approach. In Section 3.3, we perform simulations to verify the proposed approach. Concluding remarks are given in Section 3.4.

3.2 Collaborative Distributed Retrieval

3.2.1 Proposed Improvement

In the architecture of [58], the client calculates the sizes of the segments of the object to be delivered by each server, according to the relative bandwidths of all the channels, and then the client requests each server to deliver only its own part of the object. When some servers finish delivering their own segments earlier than others, they idle and waste their remaining bandwidth, even though the transmission of the whole object is not finished. The essence of the statistical model of [58] is to estimate K channels individually, then to sum up the respective estimated result of each channel. Therefore, each channel should satisfy the respective estimation independently. Intuitively, this requirement is stricter than needing to satisfy only the estimated sum of the K channels. As a result, the estimated object size under the previous architecture is smaller. Based on this observation, we try to improve the architecture and the statistical model by estimating the sum of the K channels directly.

In [58], some unreliable channels were dropped to achieve the better-estimated size. To extend the idea, we introduce a reliability factor that can vary between 0 and 1, compared to only 0 or 1 in [58].

3.2.2 Architecture

Like the authors of [58], we assume the potential application is to deliver a best-quality multimedia object from a number of distributed servers to a client program within a time limit T. To estimate the size of the best-quality object, the client first uses a fraction (t) of T to perform bandwidth testing. We try to determine the optimal t and the largest expected size.

Once we obtain the largest expected size of the object, each server tailors the media object to the largest expected size. Each server can also calculate the approximate size of the strips of object to be delivered by each server, according to the relative bandwidth of all the channels. Next, the client requests each server to deliver its strips of the object in the remaining *T-t* time. A server that finishes its share of the retrieval task helps with the retrieval tasks of other servers. As we pointed out in the previous sub-section, in the architecture of [58], the servers finishing their own tasks early idle and waste their remaining bandwidth. The difference between the proposed architecture and the previous one is that the current architecture fully uses the available bandwidth of each channel.

In the remainder of this section, we will focus on the bandwidth estimation and how to determine *t* under the proposed architecture.

3.2.3 Mathematical Model and Algorithm

Definition 3.1: Basic Notation

- T is the time limit for object transmission.
- t is the time used for bandwidth testing.
- t_s is the time slice used to obtain bandwidth samples.
- $N = \frac{T}{t_s}$ and $n = \frac{t}{t_s}$. □

Suppose the bandwidth populations of all K channels, $X_1, X_2, ..., X_K$, conform to the normal distribution [58], $X \sim N(\mu, \Sigma)$,

where

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \cdots \\ X_K \end{pmatrix}, \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \cdots \\ \mu_K \end{pmatrix}, \Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1K} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2K} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_{K1} & \sigma_{K2} & \cdots & \sigma_{KK} \end{pmatrix}.$$

The assumption of normality of variable x is reasonable, since within a short period the available bandwidth tends to approach a constant [58].

Instead of simply dropping the unreliable channels as in [58], we introduce a factor β between 0 and 1 for each channel. We use this factor to maximize the estimated bandwidth. The factor is proportional to the reliability of each channel. We call it the **reliability factor**. We can then form a linear combination as follows:

$$Z = \beta_1 X_1 + \beta_2 X_2 + ... + \beta_K X_K = \beta^T X \text{, where, } \beta_1, ..., \beta_K \in [0,1].$$

According to statistical theory [62], Z has a $N(\mu_z, \sigma_z^2)$ distribution, where $\mu_z = E(Z) = \beta^T \mu$, $\sigma_z^2 = Var(Z) = \beta^T \Sigma \beta$, i.e., $Z \sim N(\beta^T \mu, \beta^T \Sigma \beta)$.

Suppose $x_{k1}, x_{k2}, ..., x_{kn}$ are the samples for channel k. $\bar{x}_1, \bar{x}_2, ..., \bar{x}_K$ are the sample means for channels 1, 2, …, K (i.e., $\bar{x}_k = \frac{1}{n}\sum_{j=1}^{n} x_{kj}$). $\bar{x}$ and S are the sample mean vector and covariance matrix of the samples, respectively.

The sample mean and variance of $z_1, z_2, ..., z_K$ are

$$\bar{z} = \beta^T \bar{x} \text{ and } s_z^2 = \beta^T S \beta.$$

For a fixed β and an unknown σ_z^2, a $100(1-\alpha)\%$ confidence interval for $\mu_z = \beta^T \mu$ is based on the student's t-distribution [62]; i.e.,

$$t = \frac{\bar{z} - \mu_z}{\frac{s_z}{\sqrt{n}} \cdot \sqrt{\frac{N-n}{N-1}}} = \frac{\beta^T \bar{x} - \beta^T \mu}{\frac{\sqrt{\beta^T S \beta}}{\sqrt{n}} \cdot \sqrt{\frac{N-n}{N-1}}} \quad . \tag{3.1}$$

The "student-t distribution" is known to be "robust", which means although it is assumed that the parent distribution (the bandwidth sample variable) is normal, this assumption can be relaxed without significantly changing the sampling distribution of "student-t" distribution [58].

Our problem is to find the estimate of μ : μ_{est} , where $P(\mu_{est} < \beta^T \mu < \mu) = 1-\alpha$; i.e., $P(\beta^T \mu > \mu_{est}) = 1-\alpha$. According to Equation (3.1),

$$P\left(\frac{\beta^T \bar{x} - \beta^T \mu}{\frac{\sqrt{\beta^T S \beta}}{\sqrt{n}} \cdot \sqrt{\frac{N-n}{N-1}}} < t_\alpha (n-1) \right) = 1-\alpha$$

or

$$P\left(\beta^T \mu > \beta^T \bar{x} - t_\alpha (n-1) \frac{\sqrt{\beta^T S \beta}}{\sqrt{n}} \cdot \sqrt{\frac{N-n}{N-1}} \right) = 1-\alpha \ .$$

We can now define the bandwidth estimation as follows:

Definition 3.2: Bandwidth Estimate is

$$\mu_{est} = \beta^T \bar{x} - t_\alpha (n-1) \frac{\sqrt{\beta^T S \beta}}{\sqrt{n}} \cdot \sqrt{\frac{N-n}{N-1}} \ . \qquad (3.2)$$

When α and n are fixed, we can try to compute an optimal β to maximize μ_{est}, where $0 \le \beta_1, \beta_2, ..., \beta_K \le 1$.

Definition 3.3: Expected Object Size is

$$V = \mu_{est} \cdot (T-t) = \left(\beta^T \bar{x} - t_\alpha (n-1) \frac{\sqrt{\beta^T S \beta}}{\sqrt{n}} \cdot \sqrt{\frac{N-n}{N-1}} \right) \cdot (N-n) \cdot t_s$$

We now state an important theorem for the property of the function *V(n)*, which considers V as a function of *n*, the number of samples.

Theorem 3.1: Suppose $\beta^T \bar{x}$ and $\sqrt{\beta^T S \beta}$ are constant, statistically, as a function of n; then the expected object size *V(n)* has a single maximum value. Before the maximum of *V(n)* is reached, *V(n)* increases with n, but the rate of growth decreases. After the maximum of *V(n)* is reached, *V(n)* decreases as *n* grows, and the rate of decrease grows with *n*.

Proof: Follows from similar theorems in [58] [64] with some of the terms and expressions changed.

Based on **Theorem 3.1**, we can have the following algorithm for determining the optimal Expected Object Size.

Algorithm 3.1: Estimating Expected Object Size for Multi-server Collaborative Distributed Retrieval

Obtain samples x_1, x_2 on each channel;

Compute optimal β to calculate maximal $V(2)$: $V_{max}(2)$;

Obtain sample x_3 on each channel;

Compute optimal β to calculate maximal $V(3)$: $V_{max}(3)$;

$n \leftarrow 3$;

while ($V_{\max}(n) > V_{\max}(n-1)$) {

$n \leftarrow n+1$;

Obtain a new sample on each channel;

Compute optimal β to calculate maximal $V(n)$: $V_{max}(n)$;

}

return $V_{\max}(n)$; □

3.3 Simulation and experimental results

The details of how the simulator works are similar to those in [58]. In order to verify and clearly analyze the typical cases of network transmission, we first simulate with 2 servers and typical combinations of the mean and standard deviation of the two channels. Next, to verify the effectiveness of random cases with multiple servers, we carry out the simulation with an arbitrary number of servers, with the mean and standard deviation of each channel also randomly generated.

3.3.1 Simulation with Two Servers

To do a 2-server simulation, as Table 3.1 shows, without loss of generality, we let the sample-generating parameters of Channel #1 be fixed and the ones of Channel #2 vary over several typical cases. For Channel #1, its bandwidth mean (μ_1) is 1000, and

standard deviation (σ_1) is 10; i.e., $\mu_1 = 1000$ and $\sigma_1 = 10$. For Channel #2, its mean (μ_2) varies over {1000, 500, 100, 50, 10, 5, 1}, and its standard deviation (σ_2) is chosen to be {0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8} times of the mean. The unit of measurement of the bandwidth mean, which could be Kbps or Mbps, is relatively unimportant. What is important is the relative size among the bandwidth mean of each channel. We consider Channel #1 to be more reliable than Channel #2, since the mean of Channel #1 is higher, and the standard deviation is lower. We do the simulation 100 times (with confidence levels 0.95 and 0.75) and compute the averages. The number of time slices is 100; i.e., N = 100.

Table 3.1: Parameter set used in 2-server simulation.

Channel #1	μ_1	1000
	σ_1	10
Channel #2	μ_2	{1000, 500, 100, 50, 10, 5, 1}
	$\sigma_2(*\mu_2)$	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8}

To evaluate the improved algorithm, we first verify how well it guarantees that the real transmission will be finished with the confidence level within a time limit. We observe the percentage of the overtime runs, which could not finish transmission within the time limit. We then estimate the improvement over the previous algorithm [58], which is measured by the improved percentage of the real size of the successfully transmitted media objects. In addition, we observe the behavior of the parameter β.

Guarantee of Confidence Level

The results are shown in Figure 3.1. In most cases, when the confidence level is 95%, the percentage of overtime runs ranges from 4% to 8%. In most cases, when the confidence level is 75%, the percentage of overtime runs range from 22% to 30%. These results mean the confidence level of the improved algorithm is preserved within a reasonable range.

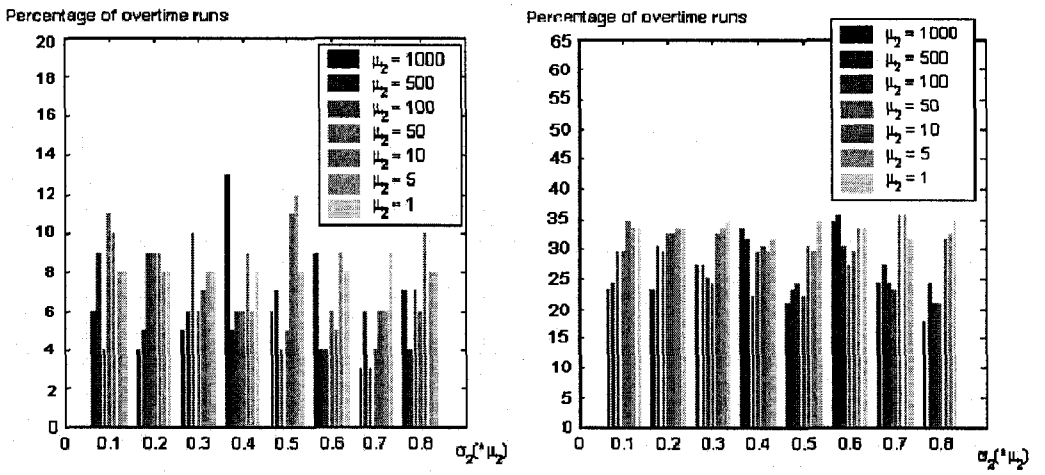


Figure 3.1: The percentage of overtime runs which fail to finish transmission within time limit; confidence level 95% (left) and 75% (right); i.e., $\alpha = 0.05$ (left) and 0.25 (right).

Improvement over Previous Algorithm

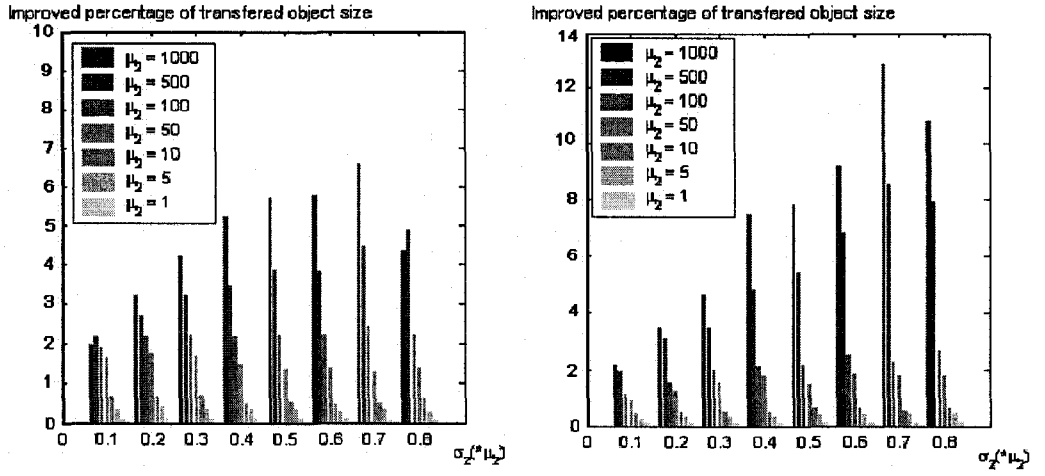


Figure 3.2: Improved percentage of estimated object size, which finishes transmission within time limit; confidence level 95% (left) and 75% (right).

As Figure 3.2 shows, for both the 75% and 95% confidence levels, we achieve a higher estimated object size. The improved percentage ranges from 0.2% to 13%. We also can find that the higher mean of Channel #2, or the closer the means of the two channels, the more the improvement. This finding occurs because with the previous method, the servers idle after finishing their independent tasks, resulting in bandwidth being wasted. The higher the mean of Channel #2, the higher the waste if Channel #2 finishes earlier, resulting in a higher proportion of waste.

Analysis of β

When the confidence level is 95%, i.e., $\alpha = 0.05$, the values of β_1 are all equal to 1. Table 3.2 shows the values of β_2 for different combinations of the means and standard deviations of Channel #2. As a general trend, when Channel #2 becomes more unreliable, i.e., as the standard deviation σ_2 becomes larger, the value of β_2 becomes smaller.

Table 3.2: The values of β_2 for different combinations of the means and standard deviations of Channel #2; confidence level is 95%; i.e., $\alpha = 0.05$.

$\sigma_2 (*\mu_2)$ / μ_2	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80
1000	1.00	1.00	1.00	1.00	0.99	0.96	0.98	0.91
500	1.00	1.00	1.00	0.99	0.99	0.93	0.94	0.95
100	1.00	1.00	1.00	1.00	0.97	0.97	0.95	0.94
50	1.00	1.00	1.00	0.99	0.97	0.97	0.97	0.95
10	1.00	1.00	1.00	1.00	0.99	0.95	0.97	0.98
5	1.00	1.00	1.00	1.00	0.99	0.97	0.97	0.96
1	1.00	1.00	1.00	0.99	1.00	0.98	0.94	0.94

When the confidence level is 75%, i.e., when $\alpha = 0.25$, the values of β_1 are all equal to 1. The values of β_2 for different combinations of the mean and standard deviation of Channel #2 are also all equal to 1. The reason is as follows: In Equation (3.2), when β_2 is less than 1, the first item, $\beta\bar{x}$, becomes smaller, and the second item,

$t_\alpha(n-1)\frac{\sqrt{\beta^T S\beta}}{\sqrt{n}}\cdot\sqrt{\frac{N-n}{N-1}}$, also becomes smaller. When α increases, $t_\alpha(n-1)$ becomes smaller; thus, the second term has less effect. Therefore, β_2 is always equal to 1, since the decrease in the first term owing to the decrease of β_2, is larger than the decrease in the second item, because the second item has less effect when α increases.

In order to further analyze the optimal β values for various network mean and variance combinations, we conducted several experiments. In most cases, if the means and variances of different channels were close to one another, then β turned out to be close to 1. However, β can vary significantly from 1 when the means and variances of different channels are quite different from one another. For the 2-channel case, we set μ_1 =1000, σ_1 =10. Figure 3.3 shows several typical cases of the ObjectSize-β_2 curve. In Figure 3.3(a), the estimated object size reaches its maximum when β_2 = 1.00. In Figure 3.3(b), when β_2 = 0.16, the estimated object size reaches its maximum; note that the mean and the std. deviation (relative to the mean) for the second channel for this test case are significantly different from those in Channel 1. In Figure 3.3(c), when β_2 = 0.69, the estimated object size reaches its maximum.

For the 3-channel case, we set μ_1 = 1000, σ_1 =10. β_1 was always equal to 1 for the situations discussed below. Figure 3.4 shows several typical cases of the optimization function surface for varying β_2 and β_3. In most cases, the surfaces are as in Figure 3.4(a), in which the estimated object size reaches its maximum when β_2 = 1.00 and β_3 = 1.00. However, in many cases, when β_2 or β_3 are not equal to 1.00 (such as in the case in Figure 3.4(b), when β_2 = 0.16 and β_3 =1.00), the estimated object size reaches its maximum.

3.3.2 Multi-Server Simulation

To do a multi-server simulation, we first generate the number of channels (K) randomly between 1 and 10. For each channel, its mean (μ_i) is generated randomly between 1 and 1000, and its standard deviation (σ_i) is generated randomly between 0 and μ_i. We do 30 tests. For each test, we do the simulation 100 times (with two different confidence levels, 0.95 and 0.75) and compute the averages. The number of time slices (N) is 100.

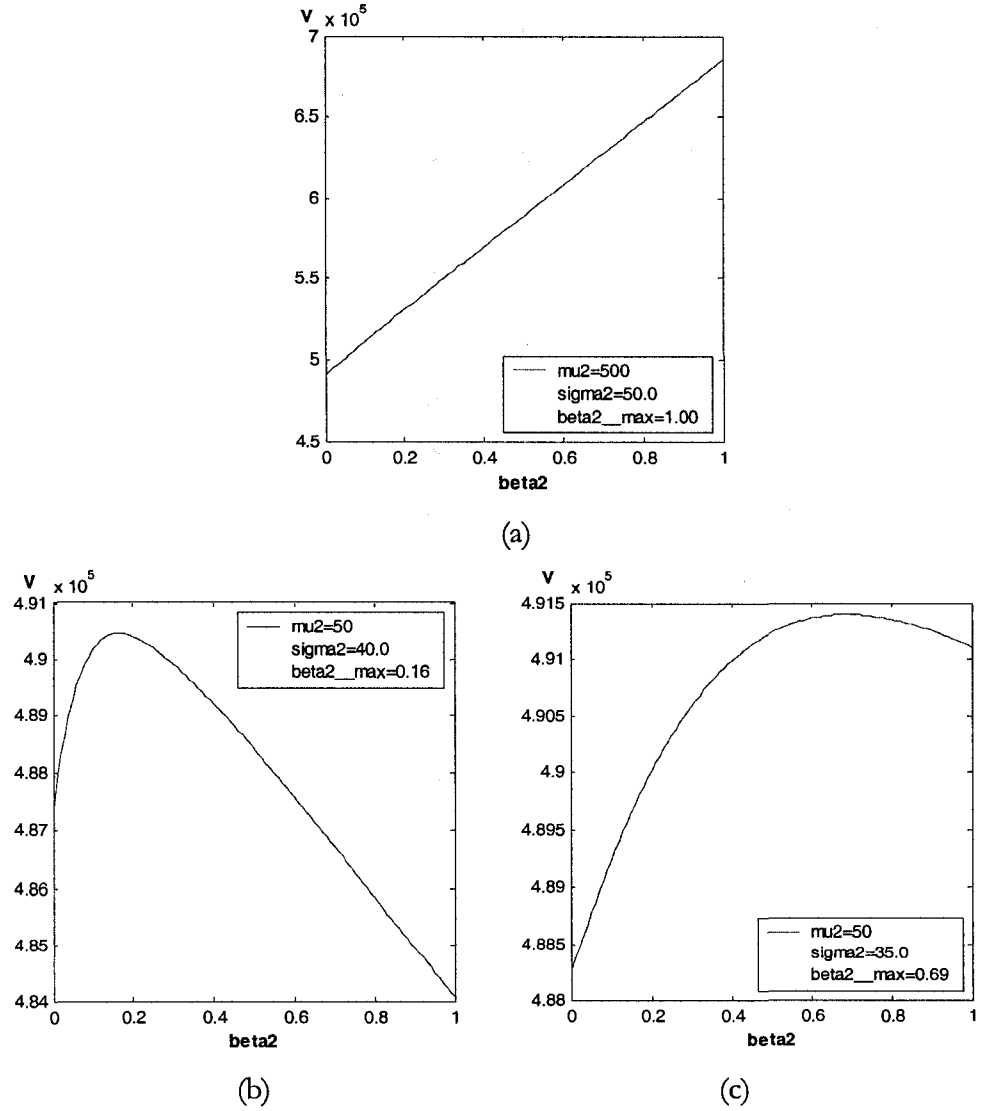
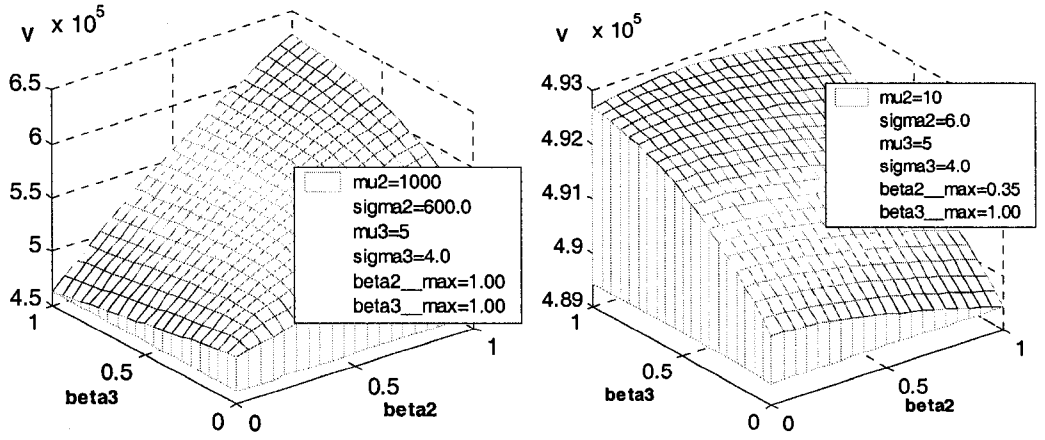


Figure 3.3: Expected object size curves for varying β_2, 2-channel case.



(a) (b)

Figure 3.4: Expected object size surfaces for varying β_2 and β_3, 3-channel case.

To estimate improvement, as with the 2-server simulation, we first verify how well our approach guarantees that the real transmission will be finished at the confidence level within a time limit. We observe the percentage of the overtime runs, which could not finish transmission within the time limit. Next, we measure the improved percentage of the real size of the successfully transmitted media objects.

Figure 3.5 shows how well the confidence level is guaranteed. When the confidence level is 95%, in most cases, the percentage of runs that fail to finish on time ranges from 4% to 8%. When the confidence level is 75%, in most cases, the percentage of runs which fail to finish on time ranges from 20% to 27%. These findings mean that the confidence level of the improved algorithm is preserved within a reasonable range.

Figure 3.6 demonstrates how well we improve on the estimated object size. When the confidence levels are both 95% and 75%, we achieve a better-estimated object size. The improved percentages range from 1% to 41%. We also find that the higher the number of channels, the more the improvement, because with the previous method, the higher the number of channels, the higher the confidence level of each channel needs to be, for the previous method estimates each channel independently, resulting

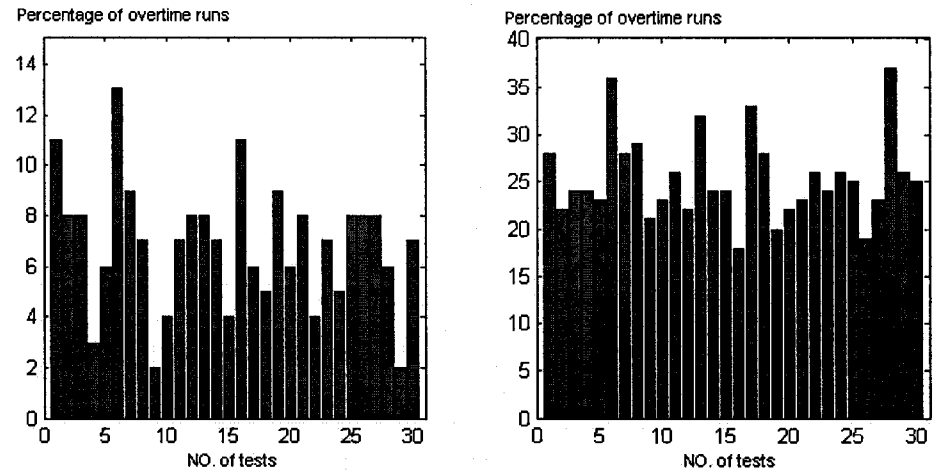


Figure 3.5: The percentage of overtime runs, which fail to finish transmission within time limit; confidence level 95% (left) and 75% (right).

in joint probabilities being the product of individual probabilities. In other words, with the previous method, the higher the number of channels, the more the servers idle after finishing their independent tasks, resulting in more bandwidth being wasted.

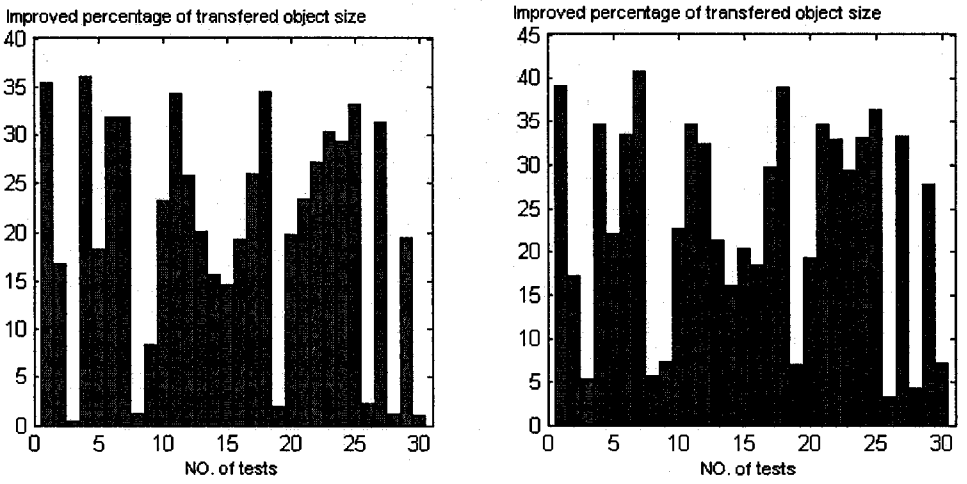


Figure 3.6: Improved percentage of estimated object size, for transmissions completed within time limit; confidence level 95% (left) and 75% (right).

3.4 Conclusions

In this chapter, we analyzed the problem of optimal bandwidth monitoring for QoS-based multimedia retrieval applications with distributed servers. The improved algorithm has several important features: first, it delivers a user confidence level for QoS specified by the user by accurately estimating the future bandwidth. Second, it dynamically determines how much sampling should be performed under different characteristics of the available channels so as to maximize the size of the multimedia object transmitted. Third, all the channels work together, and we estimate the total bandwidth of K channels directly. Finally, we introduce a reliability factor between 0 and 1, for each channel, to maximize the estimated object size.

In the next two chapters, our work on P2P VOD and peer selection will be presented.

Chapter 4

PCVOD: INTERNET PEER-TO-PEER VIDEO-ON-DEMAND WITH STORAGE CACHING ON PEERS

Video-On-Demand (VOD), such as YouTube-like online video sharing, online theatres and online news videos, is becoming much more popular on the Internet than VOD was previously. However, VOD consumes a large bandwidth, and the cost of traditional Client/Server (C/S) solutions is expensive and increases proportionally with the increasing concurrent demands. Owing to its advantages of cheaply solving the server-side bottleneck, P2P becomes useful in this situation. In this chapter, we propose Peer Cached VOD (pcVOD), an Internet P2P VOD with storage caching on peers. In order to better support the demands of less popular videos, we propose a new scheme that requires the peers to provide storage cache space and manages the cache instead of traditionally depending on a user to manually decide how much cache space is shared and which video is to be kept for sharing. A novel cache-replacement policy based on the ratio of the number of peers demanding to the number of peers with a cache, which is unique to the P2P environment, is proposed. As our simulation results show, for popular news video demands, with little cache space on each peer, pcVOD results in a significant improvement compared to P2P VOD without the management of resource sharing. We implemented and commercialized a real-world prototype.

4.1 Introduction

The purpose of P2P is to share resources among peers and to utilize all the available resources on the Internet, so that peers benefit from one other. P2P solves the problem of the bottleneck on the server under the centralized client/server architecture. Another advantage of P2P is its low cost. P2P is an application-layer

solution, which does not need upgrading to an existing network. P2P utilizes the resources of peers, greatly reducing the requirements on the capability of a server, and, in fact, can make servers unnecessary.

With the wide adoption of the high-speed Internet, multimedia streaming is becoming more and more popular. However, one connection of multimedia streaming consumes much more bandwidth than traditional text-based messaging. The server-side bottleneck constrains the scalability. In traditional client/server solutions, every demand is handled by a centralized server, requiring a powerful server and large bandwidth. Server clusters or proxy servers [67][72] can improve the processing ability, but this option is expensive and still not very scalable. P2P architecture mitigates the problems in this situation. In recent years, substantial research on application-layer multicasting and streaming with P2P has been conducted [71] [73] [75] [76] [80].

Multimedia streaming can be categorized into live streaming and on-demand streaming. For live streaming, all the clients are synchronous. For on-demand streaming, most clients demand different videos or different parts of the same video. For popular videos, which many clients demand at the same time, some methods, such as batching or patching [83], have been proposed. These approaches work only if one video is demanded simultaneously. Less popular videos still consume limited server bandwidth. Another problem is that only a single path runs from the server to the client. P2P VOD [76] [71] [79] [81] [82] [85] [87] [88] [89] [90] [91] [92] was conceived to solve this problem. We propose pcVOD, an Internet P2P VOD with storage caching on peers. First, instead of depending on a user to manually decide how much cache space is shared and which video is to be kept for sharing, our system requires peers to provide storage cache space, and allows our cache-replacement algorithm to manage the cache. The cache-replacement strategy is based on the ratio of the number of peers demanding to the number of peers with caches, and is unique in a P2P VOD environment. Second, instead of assigning tasks to peers in equal-sized blocks, we assign as much as possible to one peer at a time while guaranteeing the data for current rendering are ready by monitoring the bandwidth of the connections to each peer.

This remainder of this chapter is organized as follows: Section 4.2 describes the architecture and model. System details are given in Section 4.3. In Section 4.4, we

perform simulations to verify the proposed approach. Real-world implementation is described in Section 4.5. Concluding remarks are given in Section 4.6.

4.2 Architecture and Model

4.2.1 Architecture

Figure 4.1 shows the architecture of pcVOD. The motivation for this work is to improve the BitTorrent-like P2P file-sharing systems [68] to support legal commercial on-demand streaming. In order to charge users for commercial applications, user authentication is critical. pcVOD has one centralized and reliable Control Center for user and node authentication and content-directory service. Although totally decentralized P2P systems [66] are currently popular technology, they are complex and inefficient for networking, and more suitable for situations without copyright and user-authentication considerations. Also, totally decentralized P2P systems have the risk that service will not be available when all peers with a needed resource are not online.

pcVOD has content-providing peers and demanding peers. Online content-providing peers are introduced to avoid the "no-seeds" problem in the P2P systems, such as BitTorrent, where unreliable normal peers provide content. Only authorized content-providing peers can release new videos. Thus, the pirating problem of copyrighted video can be better controlled than it can be with other systems. Content-providing peers with large bandwidths can better serve the demands for newly released videos and lack-of-cache videos. Each demanding peer has storage space to cache videos. A video cached on normal peers is encrypted to prevent illegal distribution. For implementation simplicity and networking efficiency, we use reliable Trackers to manage the connections of peers and caching, as does BitTorrent [68].

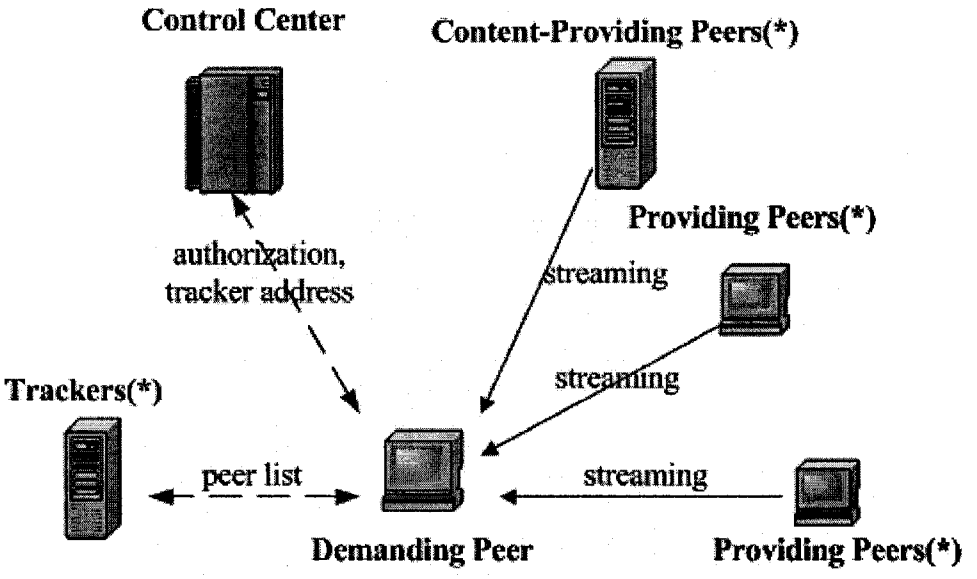


Figure 4.1: pcVOD Architecture

Multiple Trackers are available to manage the connections and cache of different videos. When a peer demands a video from the video list at the Control Center, the address of the Tracker with the demanded video will be obtained from the URL of the video list. The peer then sends a request to the tracker, which returns a list of candidate peers with the video cached. Finally, the video data are retrieved simultaneously from the candidate peers.

In our architecture, for considerations of commercial use, simplification, networking efficiency and reliability, the always-online reliable Control Center and Trackers handle the processing requiring a light burden and small bandwidth. P2P is introduced only in the most needed part, video data transmission, which has a heavy burden and is bandwidth-consuming.

4.2.2 Storage Cache and Replacement Strategy

In BitTorrent-like P2P systems, users decide which files are going to be shared, regardless of whether the files are really needed by other peers. In BitTorrent or P2P

VOD, it is very often that the streaming speed of the popular videos is very fast while that of the less popular downloads can be very slow. The reason behind is that much resources are shared for popular videos while no enough resources are shared for less popular videos. In order to better support the demands of less popular videos, we introduce storage management to improve the utilization of peer resources in P2P systems. Each peer is required to share a certain size of hard-drive storage cache. (This assumption is reasonable if the required cache space is small or in applications like a TV Set-top Box, in which a large hard-drive storage cache can be integrated.) Also, our algorithm manages how to use shared cache space. When the shared cache space is full, the replacement algorithm is used to determine which stored video should be replaced. The replacement algorithm is described as follows.

In a P2P VOD environment, the greater the number of available peers with video caches, the higher the probability that the video is available, and the higher the effective bandwidth. We want a video with a high-demand ratio to have more cache available. Thus, we design the rank of the cache requirement, *cache_needed_rank,* which is the ratio of the number of peers demanding to the number of online peers with a cache; *i.e.*,

$$cache\_needed\_rank = \frac{the\ number\ of\ \ peers\ demanding}{the\ number\ of\ online\ peers\ with\ cache}.$$

The higher the *cache_needed_rank*, the higher is the priority of a stream being cached. The cache-replacement policy is based on the *cache_needed_rank*. The cached stream with a lower *cache_needed_rank* is replaced by one with a higher *cache_needed_rank*. The cache strategy takes into account both the video's popularity and the number of peers that are currently caching that video. This strategy is unique in the P2P environment and is different from traditional caching schemes, such as the Least-Recently-Used, aging, and popularity-based schemes.

4.2.3 Pre-fetching

Pre-fetching Retrieval Strategy

Instead of "play-after-retrieving" in file sharing, on-demand streaming is "play-during-retrieving." The front part of the data should be retrieved first in streaming, but unlike usual streaming, our streaming fetches data in advance (pre-fetching) and retain them in storage if the retrieving rate is faster than the playback rate. Also, unlike traditional streaming from one server, our method retrieves from many peers in parallel and does not retrieve in exact bit-order. The retrieval procedure is divided into many pre-fetching periods. In each period, data are not fetched in order, but are assigned to the connections of candidate peers by block. The retrieval procedure pre-fetches as much data as possible while guaranteeing the data for current rendering are ready. With this strategy, we assign tasks to the peers in larger blocks and thus reduce the protocol load. The detailed retrieval strategy is as follows:

1) At the beginning of streaming, set an initial pre-fetching size, which is determined by the initial buffering time.
2) Assign one retrieving part to each connection of a candidate peer with equal size.
3) When a connection to one peer finishes transmitting its assigned part, a new part from the unfinished current assignments of other connections is re-allocated, whose size is determined according to the statistically estimated bandwidth of this connection.
4) If the video is not all fetched, according to the sum of the statistically estimated bandwidth of each connection, a new pre-fetching size is computed, and a new pre-fetching period starts.
5) Assign a new pre-fetching task to each connection according to its current bandwidth and go to Step 3.

During retrieval, the newly joining peers with a video cached, and new peers finishing caching the whole video, will be added to a candidate peer list. Also, the departed peers and slow peers will be deleted. Our retrieval method deals well with network bandwidth fluctuations, since the retrieval procedure monitors each connection's

bandwidth, assigns tasks dynamically, and has relatively large hard-drive storage as a buffer.

Every time a new retrieving task to a candidate peer is assigned, a control message is sent. In order to reduce the protocol load, a lower number of task assignments is preferred. Obviously, the larger the size of each retrieval task is, the less the number of the task assignments is. However, if the size of a pre-fetching task is too large, later data might be pre-fetched while the earlier data for current playback are not ready. The pre-fetching size should be as large as possible while guaranteeing the data for current playback are ready. The safe maximum pre-fetching size is calculated as follows.

Figure 4.2 shows the pre-fetching of a video file. The x-axis is the rendering time of the video. The gray part is the fetched data, and the white part is the data to be fetched. T_r is the time point of the current rendering, and T_f is the time of the end of the fetched data. Before the rendering time reaches the end of the fetched data, a time period of $(T_f - T_r)$ occurs. In order to guarantee that the data for rendering at time T_f are ready, the pre-fetching must finish within the time period $(T_f - T_r)$, which is the safe pre-fetching time.

The bandwidth is estimated by using an interval estimate using the student's t-distribution [58]. After the bandwidth μ_{est} is estimated, the safe maximal pre-fetching size is equal to the estimated bandwidth μ_{est}, times the safe pre-fetching time, $(T_f - T_r)$.

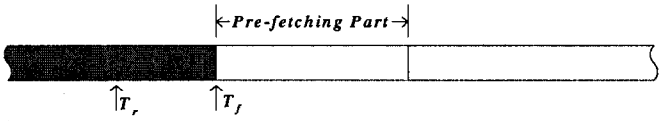


Figure 4.2: Diagram of Pre-fetching

Thus, the pre-fetching size V is

$$V = (T_f - T_r)\mu_{est} \quad . \tag{4.1}$$

Analysis of Protocol Load Decrease

We analyze the protocol load decrease mathematically by assuming that the overall retrieving bitrate is constant and the connection number does not change. We assume the retrieving bitrate is larger than the video bitrate since pre-fetching is meanful only under this condition. The following notations are used:

V: Video size

I: Initial pre-fetching size

R: Bitrate of the video

B: Retrieving bitrate (the sum of retrieving bitrates for all connections)

p: $\frac{B}{R}$, the ratio of retrieving rate to bitrate of the video

n: Number of pre-fetching

m: Number of connections (active candidate peers)

b: the block size for the strategy of block-by-block task assignment

We can calculate the size of every pre-fetched block as follows.

The time that the 2nd pre-fetch can use: $\frac{I}{R}$.

The size of the 2nd pre-fetch F_2: $B\frac{I}{R} = I\frac{B}{R} = Ip$.

The time that the 3rd pre-fetch can use: $\frac{F_2}{R}$.

The size of the 3rd pre-fetch:

$$B\frac{F_2}{R} = B\frac{I\frac{B}{R}}{R} = I(\frac{B}{R})^2 = Ip^2.$$

...

The size of the i^{th} pre-fetch: Ip^{i-1}.

Thus, we have

$$V = \sum_{i=1}^{n} Ip^{i-1} = I\frac{1-p^n}{1-p}.$$

It follows that

$$n = \left\lceil \frac{\ln((p-1)\frac{V}{I}+1)}{\ln p} \right\rceil.$$

Thus, the total number of tasks assigned by pcVOD is

$$mn = m\left\lceil \frac{\ln((p-1)\frac{V}{I}+1)}{\ln p} \right\rceil. \quad (4.2)$$

The total number of protocol messages sent by using the strategy of block-by-block task assignments is $\frac{V}{b}$.

Finally, the ratio of protocol load decrease can be computed as

$$mn/\frac{V}{b} = \frac{bm}{V}\left\lceil \frac{\ln((p-1)\frac{V}{I}+1)}{\ln p} \right\rceil. \quad (4.3)$$

4.3 System Details

4.3.1 Data Structure

A specified Tracker manages the connections of the peers and caching. The Tracker maintains a table of the Video Cache Information (*VCI*) of each video (see **Data Structure 4.1**).

The *VCI* includes a video's id, *video_id;* its bitrate, *bitrate*; the number of peers demanding the video, *demanding_number*; the number of online peers with the video

cached, *online_cache_number*; its rank of the cache requirement, *cache_needed_rank*; and a table of current caching information, *cache_info*. The *cache_needed_rank* is the ratio of the number of peers demanding to the number of online peers with the video cached. The *cache_info* includes the caching peer's id, *peer_id*, and the online status, *is_online*.

Data Structure 4.1:

```
stucture {
    int video_id;
    float bitrate;
    int demanding_number;
    int online_cache_number;
    float cached_needed_rank;
    structure {
        int peer_id;
        bool is_online;
    } cache_info[];
} VCI; □
```

4.3.2 Main Procedure

The demanding procedure is described in **Algorithm 4.1**. In order to reduce the burden on the Tracker, the *cached_needed_rank* for the decision about the cache replacement is requested only at the beginning.

4.4 Performance Evaluation

4.4.1 Evaluation of Storage Cache Replacement Strategy

(1) Experimental Setup

We simulate a typical high-volume application, the demand for short videos of hot events at a news website. Since we focus on comparing different distribution and retrieval strategies, the simulation deals with the processing at the application level only

and does not consider the low-level networking issues, such as the topology. We implement the simulator by using the C++ language and can run on both Windows and Unix platforms. Our simulations run on a PC, with Pentium-4 2.4G CPU, 1G Memory, and on a Linux Operating System.

Algorithm 4.1: P2P video-demanding procedure

Peer (*P*) gets the Tracker's address from video URL at Control Center;

P sends required video ID to Tracker, and Tracker updates the *demanding_number* of the *VCI* and returns the *VCI* to *P*;

P connects to candidate peers given in the returned *VCI* to retrieve the video;

```
caching_video = true;
while (retrieval is not finished && demanding is not cancelled) {
    if (the cache space is full && caching_video == true) {
        pick the video with the lowest cache_needed_rank in the local cached video:
        local_cached_streams[l];

        if (video.needed_rank
                    < local_cached_streams[l].cache_needed_rank) {
            caching_video = false;
        }
        else {
            free the space of local_cached_streams[l];
            Tracker deletes local_cached_streams[l] in VCI;
        }
    };
    if (caching_video == true)
        cache the video;
};
if (caching_video == false) {
    free the space of partly cached video;
}
if (retrieval is finished and caching_video == true) {
    Tracker adds P into VCI;
} □
```

The parameters of the simulation are designed by considering the practical situations and the balance between making the simulation possible and avoiding loss of generality. As shown in Table 4.1, the simulation has 10,000 peers and 206 videos. The samples are from all available news videos on May 23, 2004 at the CNN website (http://edition.cnn.com/video/). Figure 4.3 shows snapshots of some of the videos used. The bitrate of the videos is 37.6KBps (301Kbps), and the duration ranges from 37 to 384 seconds with a mean of 140.0 and standard deviation of 65.7. The duration data are available at http://www.cs.ualberta.ca/~lihang/pcvod/video_from_cnn.txt.

Figure 4.3: Snapshots of sample videos used.

Table 4.1: P2P VOD Simulation Setup

Number of Peers	10,000
Online Peers	2,000
Number of Videos	206
Bitrate of Videos	37.6KBps
Mean of Duration of Videos	140.0 seconds
Std. Dev. of Duration of Videos	65.7
Up-Link Bandwidth of Content-Provider Peer	2MBps
Down-Link Bandwidth of Other Peers	31.25-187.5KBps
Up-Link Bandwidth of Other Peers	15.625-62.5KBps
Rate of Peer Login	1 per second
Rate of Peer Logout	1 per second
Rate of Peer Demanding	10 per second

We make the following assumptions for the simulation: At the beginning, one content-provider peer (the release seed) and 2,000 other peers without any cached videos are online. One peer is logging out every second. The peers logging out are randomly selected from the online peers. Also, one peer is logging in every second. The peers logging in are randomly selected from the offline peers. Ten peers are starting to demand a video in one second. The demanding peers are randomly selected from the online peers, which are not demanding currently. Finally, the video demanded is randomly selected.

The up-link bandwidth of the content-provider peer (the release seed) is 2 MBps. The peers' parameters are designed to reflect the diversity in a P2P community [76]. The available bandwidth of each connection is chosen randomly in the range [31.25KBps, 187.5KBps]. The up-link and down-link bandwidth limitation of the peers is chosen randomly in the range [15.625KBps, 62.5KBps]. When the upper-bound of the bandwidth on the peers is reached, the bandwidth of each connection is assigned proportionally to the ideal bandwidth of the connection without competition from other connections.

(2) Comparison among pcVOD, P2P VOD without cache management, and client/server VOD

We compare pcVOD with P2P VOD without cache management, which has the same amount of total cache space in the system. Since the P2P architecture is used to solve the problem of server bottleneck, we compare by focusing on the volume of demands served. The reject rate of the demanding requests is used, reflecting the volume of demands served. Suppose pcVOD has a cache of C Mega-bytes on each peer. In P2P VOD without cache management, 50% percent of the peers have no cache, and another 50% percent of the peers have a cache. The cache size is chosen randomly in the range [0, 4*C]. As a result, the total cache space in the system is the same as pcVOD's. One of the cached videos is chosen randomly, to be replaced by a new video when the cache space is full. In the simulation, C = 10; also, a peer quits demanding if frequent buffering occurs, so that pre-fetching is slower than playing.

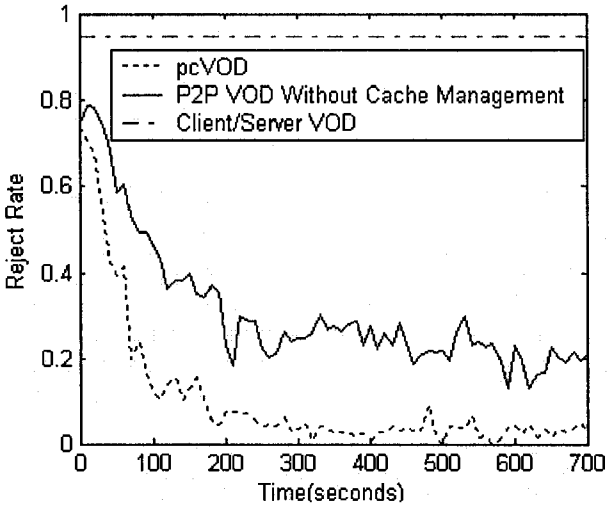


Figure 4.4: Comparison among pcVOD, P2P VOD without cache management, and client/server VOD

Figure 4.4 shows the comparison among pcVOD, P2P VOD without cache management, and client/server VOD. The dotted line is the curve of pcVOD. The solid line is the curve of P2P VOD without cache management. The dash-dot line is the minimal reject rate by a traditional client/server solution. As Figure 4.4 shows, pcVOD and P2P VOD have much lower reject rates, illustrating the benefits of P2P architectures. Note that both reject rates decrease as time increases because more replications (as a result, more up-link bandwidth) are available in a P2P network as time goes by. Also, pcVOD shows a significant improvement compared to P2P VOD. The reasons are as follows: In P2P VOD, peers are without cached videos, and some peers have a larger cache space and more video cached. As a result, while the up-link bandwidth of the peers without caches is not utilized, more peers retrieve data from the peers with caches, making these peers reach the limitation of the up-link bandwidth. In pcVOD, cached videos are distributed to more peers, better utilizing the bandwidth of more peers. Our simulation also shows that pcVOD has a better cache hit rate than P2P VOD without cache management. This result also justifies the above observation.

The above simulation with the same total cache space in a system shows pcVOD has a lower reject rate than that of P2P VOD without cache management. We can thus state that with the same objective performance, pcVOD requires less total cache space.

(3) Analysis of pcVOD with different Cache Sizes

Reject Rate

Figure 4.5 compares the reject rates of pcVOD with different cache sizes. Figure 4.5 shows the larger the Cache Size is, the lower the reject rate is, and the higher the volume of demands served is. However, the reject rates for Cache Sizes of 20M and 40M are almost the same. Thus, after the Cache Size reaches a certain threshold, the reject rate stays almost the same because when the Cache Size reaches the threshold, a sufficient number of caches of each video are available, and pcVOD can serve every demand.

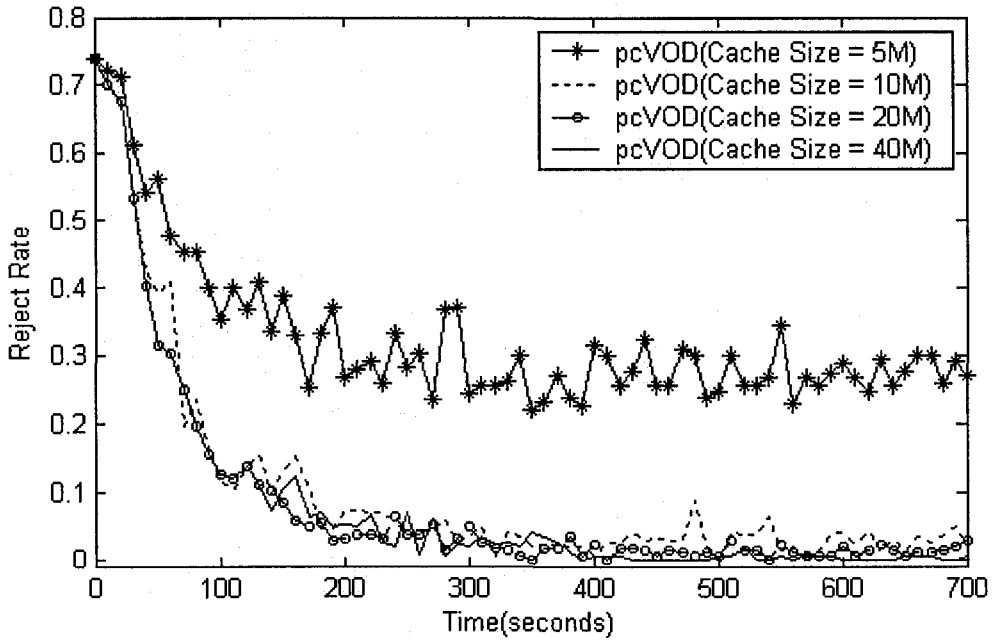


Figure 4.5: Analysis of Reject Rate of pcVOD with different Cache Sizes

Cache Hit Rate

Figure 4.6 compares the cache hit rate of pcVOD with different cache sizes. Figure 4.5 shows that as time goes by, more replications are available, and the cache hit rate increases. The larger the Cache Size is, the higher the cache hit rate is. Together, Figure 4.5 and Figure 4.6 show that the reject rate decreases when more replications and up-link bandwidth are available. This decrease results in the cache hit rate increasing.

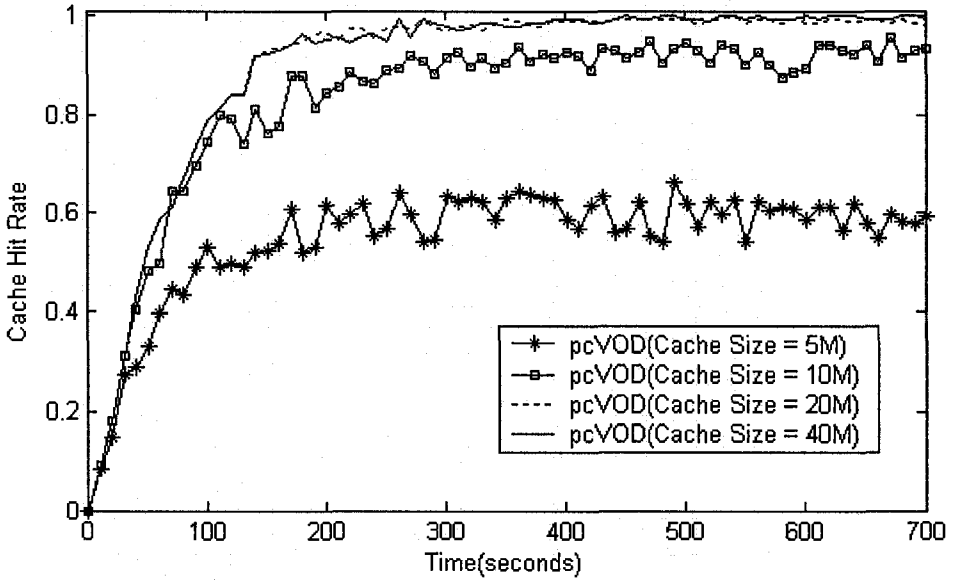


Figure 4.6: Analysis of cache hit rate of pcVOD with different cache sizes

Release Seed Bandwidth Utilization

A release seed releases and initially shares files. A content-provider peer is a release seed. Through the analysis of bandwidth utilization of a release seed, we can see how the bandwidth of a content-provider peer is utilized, based on which proper releasing strategy can be designed. Figure 4.7 compares the release seed bandwidth utilization of pcVOD with different Cache Sizes. The release seed bandwidth utilization, with Cache Size = 5M, is full all the time because not enough replications (not enough up-link bandwidth) are available in a P2P system. As a result, the release seed bandwidth is used all the time, and demands are rejected (Figure 4.5). The release seed bandwidth utilization after 300 seconds with Cache Size = 20M or 40M is kept at 0 most of the

time, so that the cache hit rate is 100% (Figure 4.6), and most demands are served by the caches.

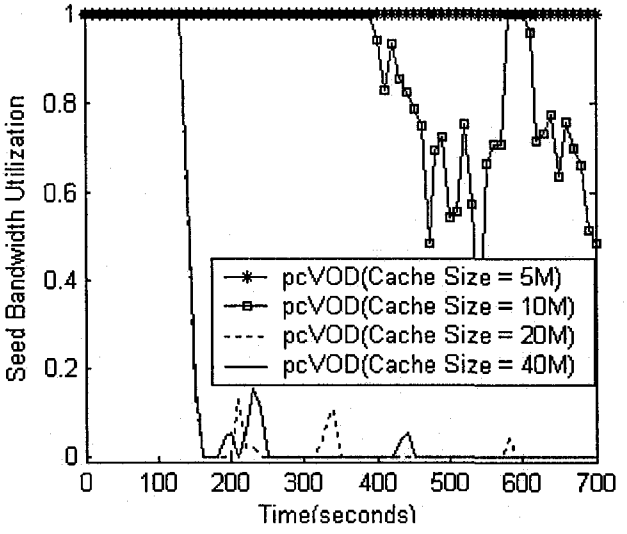


Figure 4.7: Analysis of release seed bandwidth utilization of pcVOD with different cache sizes

4.4.2 Analysis of Pre-fetching Strategy

From Equation 4.3, the ratio of protocol load decrease is $\frac{bm}{V}\left\lceil \frac{\ln((p-1)\frac{V}{I}+1)}{\ln p} \right\rceil$.

Intuitively, pre-fetching reduces the protocol load significantly only when the video size is large. Our observations are based on a typical case of movie-sharing in the BitTorrent community, with video size V=256MB, connection number m=10 and the block size b=256KB. (In BitTorrent, the block size b is normally chosen as 256KB[1]).

Figure 4.8 presents the curves of the protocol load decrease ratio with different initial pre-fetching sizes, I, and the ratio of the retrieving bitrate to the video bitrate, p. Figure 4.8 reveals that the larger p is, the lower the protocol load is. Also, the larger I is, the lower the protocol load is. With I=1MB and p=1.6, the number of task-assigning messages by pcVOD is 11% of the message number in the block-by-block fetching.

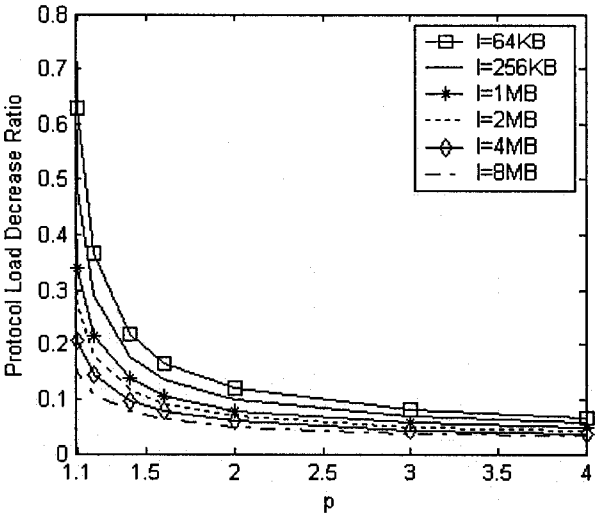


Figure 4.8: Analysis of protocol load decrease ratio

Pre-fetching is useful when the video size is large, the large initial pre-fetching is allowed, the retrieving bandwidth is large (p is large), and the number of connections is small; i.e., the peers with the highest bandwidths are efficiently selected.

4.5 Real-World Implementation

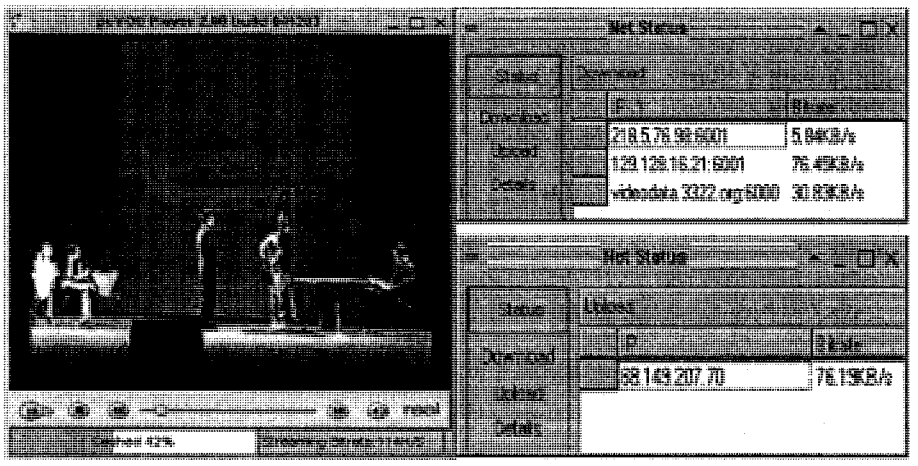


Figure 4.9: Snapshot of peer in pcVOD prototype.

We have implemented a prototype of pcVOD. In this demo system, we have a control center, one content-provider peer ('videodata.3322.org') and one normal peer ('218.5.76.98') with a cached video located in Xiamen, China, and one normal peer ('129.128.16.21') with a cached video located at the University of Alberta, Edmonton, Canada. Figure 4.9 shows a snapshot of an experiment with a demanding peer ('68.149.207.70') with high-speed cable access from a residence in Edmonton, Canada. The left side of Figure 4.9 shows the main playbacking window of the demanding peer. The right-top side of Figure 4.9 shows the window of the demanding peer tracking the candidate peers. To guarantee sufficient streaming bitrate will be available, multiple concurrent connections are set up to retrieve data from the content-provider peer. In order to reduce the burden on the normal peer, only one connection is created to the normal peer. The right-bottom side of Figure 4.9 shows the window of the peer '129.128.16.21', which tracks the uploading bitrate.

We commercialized the prototype and have attracted over 100,000 downloads so far.

4.6 Conclusions

We proposed pcVOD - a P2P VOD with caching on peers. pcVOD allows for the management of sharing the peers' storage in P2P VOD. Unlike systems with traditional streaming, our system has a hard-drive storage at each peer for the cache. The cache strategy is based on the ratio of the number of peers demanding to the number of online peers with caches. As our simulation results show, pcVOD shows a great improvement compared to P2P VOD without cache management. The effect of the size of the cache space on the VOD-serving capability of pcVOD was also analyzed. It was shown that the optimum performance can be achieved for a reasonably small cache size. Pre-fetching was proposed to fetch as much as possible at a time, while guaranteeing the data for current playback are ready. Through mathematical analysis and case study, pre-fetching was shown to be useful in the case where the video size is large and the retrieving bandwidth is large. As well, we implemented and commercialized a real-world pcVOD prototype.

Chapter 5

TRACEROUTE-BASED FAST PEER PRE-SELECTION WITHOUT OFFLINE DATABASE

In the last chapter, we have presented our work on Internet peer-to-peer video-on-demand system with storage caching on peers. The extreme heterogeneity in the P2P Internet environment causes the connections to candidate peers to vary significantly. Thus, selecting good candidate peers is critical to P2P networking performance. While most existing studies focus on selecting peers with low latency, high uploading bandwidth, and high serving stability by using time-consuming end-to-end measurements, we address how to return good candidate peers from the peer list server quickly and efficiently without probing. To improve P2P applications where the quality of the links varies greatly and to reduce the traffic between different ISPs and different locations, peers from different ISPs and different locations must be distinguished. In this chapter, we propose a new scheme to distinguish the close peers from the same location or with low round-trip time from each other by observing that close peers share routers along the traceroute path to the same destination. Our method maintains only the online peers' traceroute records, without any offline database. Experiments with real traceroute records collected by our pcVOD prototype from 352 different IP addresses around the world verify the efficiency of our method.

5.1 Introduction

As mentioned in Section 4.1, Peer-to-Peer (P2P) networking is widely used in applications of file sharing and instant messaging. In P2P, peers communicate directly and share resources with one other. P2P utilizes the resources of peers, greatly reducing the requirements on the capability of a server and, in fact, can make servers unnecessary. Thus, P2P can overcome the server-side bottleneck problem in the centralized client/server architecture. Other advantages of P2P are its low cost and

ease of deployment. P2P is an application-layer solution, which does not need an upgrade to an existing network. P2P technologies are being adopted in more and more applications such as multicasting and online databases.

As mentioned in Section 2.3.3, in P2P networks, usually a peer first retrieves a list of candidate server peers, then selects an active set from the list, and retrieves data from the peers in the set. During transmission, a list of new candidate server peers is periodically retrieved and added into the active set. In the meantime, good active peers are kept, and poor active peers are dropped. Due to the extreme heterogeneity in the P2P Internet environment, the connections to the candidate server peers vary greatly. As a result, selecting good candidate server peers is critical to P2P networking performance. The selection criteria include low-latency, high uploading bandwidth, and serving stability. As well, good candidate server peers need to be selected quickly. Most studies in the literature focus on selecting new peers, keeping good active peers and dropping poor active peers by probing and making end-to-end measurements during transmission.

With the wide adoption of the high-speed Internet, multimedia streaming is becoming more and more popular. However, one connection of multimedia streaming consumes much more bandwidth than traditional text-based messaging. The server-side bottleneck of traditional client/server solutions constrains the scalability. P2P architectures mitigate the problems in this situation. In recent years, substantial research on application-layer multicasting and streaming with P2P has been conducted. Some studies have examined P2P on-demand streaming [76] [105] [71] [81] [82] [85] [91], but most of these are on architecture and system design. Few detailed, quantitative and optimal studies are oriented towards P2P Video-On-Demand (VOD) and topics such as retrieval strategy, shortening the initial buffering time and packet assignments. This present work is motivated partly by the desire to shorten the initial buffering time in P2P VOD.

One critical factor for QoS of VOD is the need to quickly start to play after a user clicks to demand a video, i.e., to shorten the initial buffering time. However, the P2P architecture has drawbacks that work against shortening the initial buffering time. A peer takes time to pick out good peers from the candidate list. End-to-end

measurement approaches select peers by using time-consuming probing. In this chapter, we address how to return good candidate peers from the peer list server quickly and efficiently without probing. (In this chapter, a peer is any client participating in a P2P network. The client is also a peer; however, the client is running on a local machine. The reader can think of himself or herself as the client, who is connected to numerous peers.) Thus, the chance of picking out good peers from the candidate list increases. (The "closeness" of two peers has two meanings: first, the two peers are close to each other physically; second, the link between the two peers has a low RTT or high bandwidth.) Unlike topology-based methods, our method tries to select the close peers without using inferring topology. The main idea of this work is to select peers sharing the same routers along the path of the traceroute.

The remainder of this chapter is organized as follows: Section 5.2 discusses the motivations for this work. Section 5.3 describes the details of traceroute-based peer selection. Section 5.4 presents our experiments and their analysis. The work is concluded in Section 5.5.

5.2 Motivations

Peer-to-peer live streaming, such as Coolstreaming [97], PPLive [98], and PPStream [99], has been very successful in China. PPLive claims to successfully support live broadcasting with 500,000 concurrent users. The Internet infrastructure environment in China is much more complex than in developed countries. The quality of links within China varies greatly. Furthermore, the links between different Internet Service Providers (ISP) are much slower than the links within the same ISP. Intuitively, the links within a given city and the same ISP are preferred to get good links and to reduce the traffic on the backbone network. On the other hand, Cohen [68] introduced a very successful file-sharing system -- BitTorrent. Because of its popularity of BitTorrent, BitTorrent consumes too much of the Internet traffic, creating problem for ISPs. BitTorrent considers only end-to-end performance when selecting peers. If peers physically close are selected with high priority, the traffic between different ISPs and different locations will be reduced. Therefore, to improve P2P applications in a similar networking environment to that in China and to reduce the traffic between different

ISPs and different locations, peers from different ISPs and different locations must be distinguished.

Several approaches can be used to distinguish peers from different ISPs and locations [100]. First, we could utilize a table of IP address designations. However, this table usually is too coarse and out-of-date. Second, IP location databases, which could be based on the "whois" database and collect IP address locations manually or through reports from users, could be used to find the locations of peers. (For convenience, we later use IP to represent "IP address.")However, the first drawback of this approach is that the coverage of such databases is limited. For example, the QQWry [101] database covers China well, but has limited coverage outside China. IP2Location [102] covers US adequately, but does not work well in China. Second, querying such large databases is time-consuming because the databases include all the IP subnets around the world. Another straightforward method to select close peers is to match IPs with the same prefix. Although this simple scheme has already improved peer selection, it has obvious disadvantages. First, it mistakenly picks peers from different locations with the same IP prefix as the client if the IP prefix is short. Second, the peers from the same city and the same ISP with different IP subnets than that of the client cannot be distinguished. This chapter proposes a method to automatically and efficiently distinguish peers from different ISPs and locations without using an offline database.

"Traceroute is widely used to detect and diagnose routing problems, characterize end-to-end paths through the Internet, and discover the underlying network topology. Traceroute identifies the interfaces on a forwarding path and reports round-trip time statistics for each hop along the way." [103] [104] Figure 5.1 presents an example of a traceroute from two different IPs (222.84.110.115, 219.159.216.235) of the same Internet Service Provider (China Telecom) in the same city (Liuzhou, China) to the same destination (IP:129.128.4.241, University of Alberta, Canada). In the figure, "RouterIP" means the IP of the routers along the path; "Hop" means the hop along the path from the source to a router; "RTT1", "RTT2", and "RTT3" mean the round-trip time in milliseconds between the source and a router by three tests. Although the two IPs belong to different IP subnets, they share many routers. In contrast, we can determine that the two IPs are close according to the sharing of

routers. This determination is based on the observation that the chance that all of the routers along the path are changed is low, although multiple paths run between two nodes.

In the next section, the details of how to utilize a traceroute to distinguish peers from different locations and different ISPs will be discussed.

5.3 Traceroute-Based Peer Selection

5.3.1 Tracker-Based Achitecture and Burden on Tracker

Although totally decentralized DHT-based P2P systems [66] are currently popular, they are inefficient for networking. With them, finding peers through multiple hops is time-consuming when the initial buffering time for VOD must be decreased. Therefore, we argue that using reliable Tracker servers like BitTorrent [68] to manage the connections of peers is more suitable for P2P VOD, especially for decreasing the initial buffering time. To balance the load and avoid single point failure, multiple Trackers are used to manage the connections. A totally decentralized architecture that will not delay the startup time will be developed in our future work.

With a tracker-based architecture, the burden on trackers must be reduced in order to require fewer trackers. For example, a tracker in BitTorrent maintains only the information about whether a peer is online. By default, a peer reports every minute if it is still online. Our traceroute-based peer selection scheme does not add a significant burden to a tracker because of the following measures:

- The traceroute record from a client is reported to trackers only once at the beginning.
- A client gets most of the candidate peers at the first time and gets only a few candidate peers periodically afterwards. Traceroute-based peer selection happens only once when a client requires a peer list at the first time or when the load of the tracker server is not heavy.

RouterIP	Hop	RTT1	RTT2	RTT3
172.0.0.1	1	47	16	31
202.103.201.149	2	16	31	32
202.103.201.33	3	31	16	31
202.103.201.117	4	31	31	16
202.97.21.181	5	15	32	15
202.97.40.229	6	31	47	47
202.97.33.126	7	31	31	31
202.97.51.234	8	312	297	313
202.97.49.193	9	312	328	328
154.11.3.33	10	312	328	297
154.11.12.10	11	313	312	313
154.11.5.205	12	2000	328	329
207.229.13.210	13	328	328	328
129.128.3.129	14	344	343	360
129.128.153.34	15	359	328	344
192.168.254.1	16	344	375	375
129.128.4.241	17	328	359	2000

(a)

RouterIP	Hop	RTT1	RTT2	RTT3
172.0.0.1	1	907	46	63
202.103.201.149	2	31	31	16
202.103.201.33	3	16	31	47
202.103.201.9	4	47	16	31
202.97.21.181	5	31	31	32
202.97.40.229	6	46	32	31
202.97.33.130	7	16	62	78
202.97.51.230	8	438	297	312
202.97.49.197	9	313	312	297
154.11.3.33	10	297	313	297
154.11.10.1	11	328	359	328
205.233.111.132	12	391	328	359
207.229.13.210	13	375	406	2000
129.128.3.129	14	359	360	343
129.128.153.34	15	407	343	344
192.168.254.1	16	344	344	2000
129.128.4.241	17	343	344	344

(b)

Figure 5.1: An example of traceroute from two different IPs from the same city (Liuzhou, China) and the same ISP (China Telecom) to the same destination (IP:129.128.4.241, University of Alberta). (a) From IP:222.84.110.115; (b) From IP: 219.159.216.235. Note: 2000 means that the round-trip time is equal to or larger than 2000ms.

- Trackers keep only the online peers' traceroute record. Furthermore, only the first 6 hop routers and routers with the median RTT less than 150ms to the client are kept. Suppose the size of source IP is 4 bytes. The size of the router IP is 4 bytes. The size of the hop (with the range 0-6) is 1 byte. The size of RTT (with the range 0-150ms) is 1 byte. Therefore, the total size of each entry is 4+4+1+1=10 bytes. Assume one tracker manages 100,000 online peers. The size of the online peers' traceroute records is 6 * 10 * 100,000 = 6,000,000 bytes, an amount which is relatively small and efficient for querying.

5.3.2 Report Traceroute

If only one destination is used for the traceroute, clients close to the destination will lack enough routers along the path to find close peers. For example, if the traceroute destination is 129.128.4.241, the traceroute from the client 129.128.4.51 has only one entry (Figure 5.2). Thus, a client will traceroute to two destinations. The traceroute record to the destination with more hops of routers to the client is reported to the trackers. The two destinations are chosen to be as far apart as possible, such as one in China and another one in North America. Thus, the peers close to the destination in China will have enough router entries when tracerouting to the destination in North America and vice versa.

RouterIP	Hop	RTT1	RTT2	RTT3
129.128.4.241	1	0.169	0.147	0.129

Figure 5.2: An example of traceroute from 129.128.4.241 to the close destination 129.128.4.241.

To avoid waiting for a traceroute when a client starts, a client keeps a record of the previous traceroute. If an updated traceroute is not available yet, the previous record will be used. This strategy works because most routers along the path do not change

frequently. This way, only the first-time use of the program needs to wait for a traceroute.

5.3.3 Peer Selection Algorithm

Intuitively, the less the hop of a shared router to both two peers, the closer the two peers are. Peer selection is conducted from the routers with less hops to the client. **Algorithm 5.1** shows the Hop-based (or RTT-based) peer selection algorithm.

Algorithm 5.1: Hop-based (or RTT-based) Peer Selection Algorithm

Client:
D_1 = The traceroute record to the first destination;
D_2 = The traceroute record to the second destination;
$D_{selected}$ = The traceroute record of the destination with more hops of routers to the client;

Send the message of the requesting peer list to trackers with selected traceroute record $D_{selected}$ listed by hop in ascending order;

Tracker:
R_{max}= maximal round-trip time for peer selection
H_{max}= maximal hop for peer selection

Keep only the online peers' traceroute record with the first H_{max} hop routers (or routers with RTT less than R_{max} to the client);

For (each RouterIP in the client's traceroute record, whose hop to the client is less than H_{max} (or whose round-trip time to the client is less than R_{max}))
{
    Add peers with the same RouterIP, whose RTT to the client is less than R_{max} (or whose hop to the client is less than H_{max}));

    If (the number of candidate is larger than required candidate num)
    {
        Break;
    }
}
If (the number of candidate is less than required candidate num)
{
    Randomly select more peers into candidate.
}
return candidate to the client. □

Figure 5.3 presents an example of hop-based peer selection with H_{max}=6. Peer1 and the client share a router within 6 hops along the traceroute path to one of the two

traceroute destinations, Destination1. Thus, Peer1 will be selected as a candidate peer of the client.

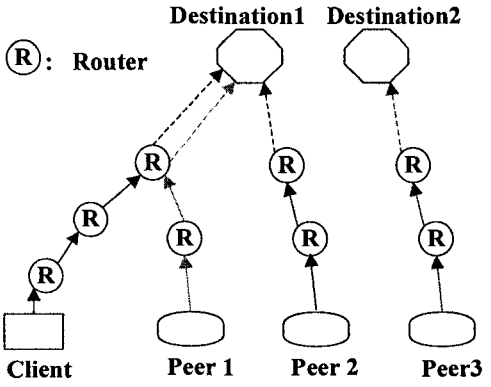


Figure 5.3: An example of hop-based peer selection.

5.4 Experiments and Analysis

We used the pcVOD real-world system proposed in Chapter 4 [105] to collect the traceroute records from 352 different IPs from Asia, North America and Europe to two different destinations, 222.89.109.150 (Zhengzhou, China) and 129.128.4.241 (University of Alberta, Edmonton, Canada). To evaluate the accuracy of selecting peers from the same city and the same ISP, we observe the data from the four cities with multiple IPs as listed in Table 5.1. The IPs from the same city are from different subnets. For example, the 21 IPs from Changsha Telecom, China are listed in Table 5.2, which shows there are 14 subnets.

Table 5.1: Cities with multiple IPs in our traceroute records

City and ISP	IP Number
Changsha Telecom, China	21
Liuzhou Telecom, China	8
Hangzhou Netcom, China	17
Shanghai Telecom, China	11

Table 5.2: IP Subnets and IPs from Changsha Telecom China

IP Subnets	IP
218.77.56.*	218.77.56.140
	218.77.56.203
218.77.58.*	218.77.58.85
218.77.59.*	218.77.59.147
	218.77.59.231
220.168.41.*	220.168.41.246
	220.168.41.89
220.168.61.*	220.168.61.155
220.168.62.*	220.168.62.125
220.168.63.*	220.168.63.70
222.240.120.*	222.240.120.179
222.240.121.*	222.240.121.54
222.240.24.*	222.240.24.116
	222.240.24.210
	222.240.24.34
222.240.25.*	222.240.25.67
222.240.26.*	222.240.26.175
	222.240.26.192
	222.240.26.208
222.240.27.*	222.240.27.149
61.150.138.*	61.150.138.70

We randomly pick one IP as the client from a city. The peer selection algorithm is used to select peers from the 352 IPs. We use the following two criteria to evaluate the efficiency of the algorithm. The **Detection Rate** is the number of selected IPs, from the client's city, divided by the total number of IPs from this city. **False Alarms** are the number of selected IPs which are not from the client's city.

5.4.1 Hop-Based Peer Selection

To analyze what the proper threshold of the maximal hop from peers to shared routers should be, we observe with different maximal hops as Table 5.3 shows. This table reveals that with increasing maximal hop, the overall detect rate increases. Before the maximal hop increases to 7, no significant false alarms occur. When the maximal hop is 6, the detect rate is quite high without false alarms. This finding verifies the efficiency of our method to automatically select peers from the same location and from different IP subnets.

Table 5.3: Detect rate and false alarms with different maximal hop

Max Hop	Overall Detect Rate	Overall False Alarms
2	35.8%	0
3	86.8%	0
4	86.8%	0
5	90.6%	0
6	90.6%	0
7	92.4%	11
8	98.1%	32

5.4.2 RTT-Based Peer Selection

Table 5.4 shows the detect rate and false alarms of RTT-based peer selection by a client from Changsha Telecom, China. This table reveals the RTT-based peer selection algorithm is not as efficient as the hop-based peer selection algorithm in selecting peers from the same city and same ISP. Specifically, the RTT-based peer selection algorithm has many false alarms.

Table 5.4: RTT-based peer selection from a client in Changsha Telecom, China

Max RTT	Detect Rate	False alarms
200	100%	5
150	90%	4
100	45%	4

5.4.3 Combined Peer Selection

Although the RTT-based algorithm is not as efficient as the hop-based algorithm in selecting peers from the same city and the same ISP, the former could be used to select more peers when not enough peers are available from the same city and the same ISP. This strategy is based on the intuition that if both of their RTTs to the same router are low, the chance of having two peers with low RTT between each other is higher. The following experiment below verifies this intuition.

Based on our collected traceroute records, the RTT of all links between China and North America is larger than 150ms, while the RTT of the most links within North

America is less than 150ms. We randomly pick one IP from Canada as the client. As Table 5.5 shows, with the hop-based peer selection algorithm with a maximal hop 6, only 3 peers from Canada can be selected. Table 5.6 shows that the RTT-based peer selection algorithm with maximal RTT 150ms can select more peers from North America without peers from China. This finding shows that the RTT-based peer selection algorithm can be used to select peers with a lower RTT to the client.

Table 5.5: Hop-based peer selection from a client in Canada

Max Hop	Number of Selected IP from Canada (outside China)	Number of Selected IP from China
6	3	0

Table 5.6: RTT-based peer selection from a client in Canada

Max RTT (ms)	Number of Selected IP from Canada/USA (outside China)	Number of Selected IP from China
200	11	2
150	10	0
100	10	0

5.5 Conclusions

In this chapter, we proposed an algorithm for selecting peers that are close to the same location or within low round-trip time from each other, following the observation that close peers share routers along the traceroute path. Oriented to P2P VOD application, a tracker-based architecture was used in order to shorten the initial buffering time. Trackers keep only the online peers' traceroute records. Based on experiments with real traceroute records from 352 different IP addresses around the world, the hop-based peer selection algorithm with maximal hop 6 could effectively select peers from the same city and the same ISP without significant false alarms. The RTT-based

algorithm with maximal RTT 150ms could be used to select more peers when not enough peers are available from the same city and the same ISP.

Chapter 6

WIRELESS PACKET LOSS AND MODELING

In the last chapters, we have discussed the multimedia transmission over multi-source networks. In the following chapters, we will discuss the multimedia transmission over wireless networks with packet loss. Unlike wired networks, wireless communication experiences packet loss due to fading and interference. To cover a larger service area and support more user capacity and traffic, we need to consider packet loss when transmitting multimedia objects over wireless networks. In this chapter, we study packet loss in congested wireless networks by performing real-world experiments in a competing WLAN environment, where packet losses and congestion are present. Although studies of packet loss in wireless networks have been published, we differ from previous researches by carrying out a comprehensive study of the impacts of packet size, sending rate, sending interval, and buffer size on packet loss. When competing connections are present, both too small and too large packet size increases packet loss. An appropriate packet size to minimize packet loss can be found. Unlike the researchers who have discussed packet size optimization based on different metrics, such as throughput, goodput, transmission range, and energy efficiency, we propose an approach for estimating the optimal packet size to maximize the expected payload.

6.1 Introduction

Some studies on packet loss in wireless networks have been published. In [107], adjusting the packet sending interval based on feedback from the network was proposed, but this work did not study the effect of different packet sizes. In [106], the authors proposed that the sending rate T could be determined as a function of the packet size s, round-trip time R, steady-state loss event rate p, and the TCP retransmit timeout value t_{RTO}. However, they did not study how their theoretical model would work in a Wireless LAN (WLAN) environment. UDP throughput and CPU utility for

bulk data transfer with different sender and receiver buffer sizes and packet sizes were studied in [109], but packet loss was not considered. The delay of UDP, TCP and TCP with the option NODELAY for voice applications sending 160 bytes per 20 milliseconds was discussed in [108]. Our work differs from that of others by carrying out a comprehensive study of packet size, sending rate, sending interval, and buffer size through real-world experiments in a competing WLAN environment, which has packet losses and congestion. Our experiments show that when competing connections are present, both too small and too large packet size causes larger packet loss. An appropriate packet size to minimize packet loss can be determined. We propose an approach for estimating the optimal packet size.

The remainder of this chapter is organized as follows: the experiments on different factors of packet loss over a lossy network are presented in Section 6.2. A strategy for packet size optimization is proposed in Section 6.3. The chapter is concluded in Section 6.4.

6.2 Network Experiments on Packet Loss

The objective of this work is to identify the appropriate parameters (packet size, sending rate, sending interval and buffer size) for different applications to maximize the throughput and minimize the packet loss of UDP transmission in different Internet environments with Wireless LAN access. In our experiments, the server side was a desktop computer in the Department of Computing Science, University of Alberta, Edmonton, Canada. The client was a laptop linked to a router following 802.11b. The router accessed the Internet by using a cable network (with a maximum capacity of 640KBps). The client was located in the same city as the server. Both the client and server ran Red Hat Linux Release 9 Shrink (2.4.30 Kernel). The experiments were conducted during the day (8:00-19:00) from November 25 to November 27, 2005.

6.2.1 Buffer Size

We first discuss the effect of socket buffer size on packet loss. Table 6.1, rows (a) and (b), show that with a fixed packet size (4096 bytes) and sending rate (128KBps),

different buffer sizes larger than the packet size have no significant affect on the packet loss. However, if the buffer size is less than the packet size, all packets are lost, as shown in Table 6.1, row (d). In row (c), when the buffer size is equal to the packet size, a significant packet loss occurs as well. This finding can be attributed to the fluctuating bandwidth; reducing the bandwidth capacity can cause an overflow on a congested buffer. In the experiments reported in sub-sections 6.2.2 - 6.2.4 below, we wanted to study the packet loss independent of the buffer size, so a large enough buffer size of 65,536 was used.

Table 6.1: Receiving rate and packet loss for different buffer sizes

Buffer Size (bytes)	Receiving Rate(Bps)	Packet Loss (%)
(a) 32,768	126611	0.98
(b) 16,384	127450	0.39
(c) 4,096	123138	3.91
(d) 2,048	0	100.00

6.2.2 Sending Rate

Next, we look into how the sending rate affects the packet loss. We consider a large enough sending interval and a fixed packet size (4096 Bytes), and let the sending rate increase from 128 to 1024 KBps. Figure 6.1 reveals that as the sending rate increases, the receiving rate increases, and the packet loss remains around zero until the sending rate is around 500 KBps. However, after the sending rate overflows the connection (after the sending rate becomes larger than 500 KBps), the packet loss dramatically increases, and the receiving rate drops owing to the packet loss.

6.2.3 Sending Interval

After fixing the packet size at 32Bytes and without overflowing the connection, sending intervals varying from 10000, 4000, 3000, 2000, 1000, 800, 600, 500 nanoseconds were used to test the packet loss rate. Figure 6.2 shows the packet loss plotted against the time interval before transmitting the next packet. Clearly, the sending interval should not be too small (< 2 ms); otherwise, the loss rate can be high.

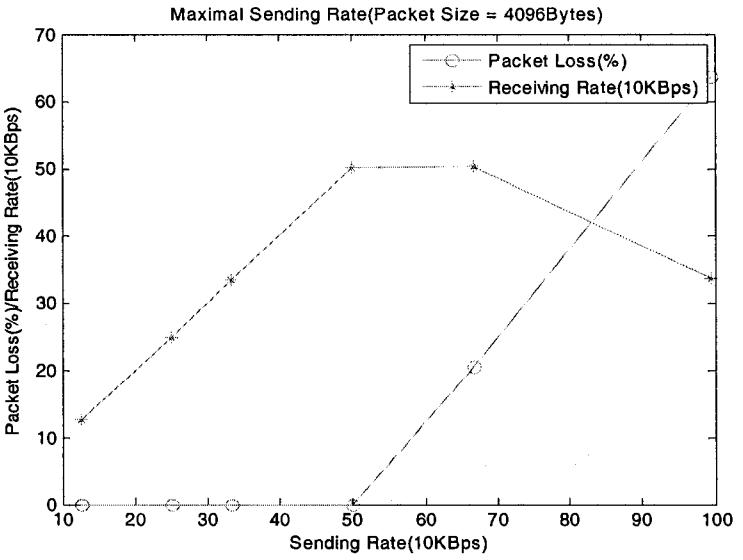


Figure 6.1: Effect of sending rate on packet loss.

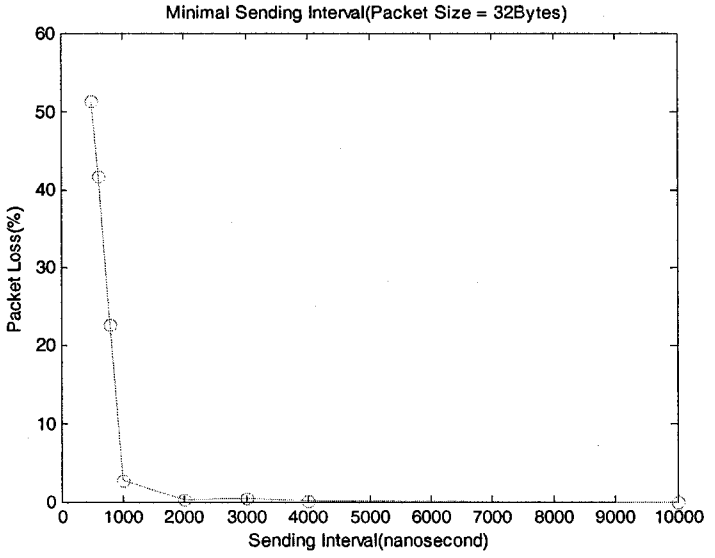


Figure 6.2: Packet loss vs. sending interval.

6.2.4 Packet Size

Next, we wanted to see how different packet sizes affect the receiving rate, as well as the packet loss. First, we performed experiments in an environment without other competing connections. With a sending rate of around 256KBps, and without

overflowing the maximum connection capacity of 640KBps, a packet size was selected from the set of [65536, 32768, 16384, 8192, 4096, 2048, 1024, 512, 256, 128, 64, 32] bytes. Table 6.2 shows that when the sending interval was large enough (equal to or larger than 2 ms), different packet sizes did not have much effect on the sending rate, receiving rate, or packet loss. The sending rate remained stable independently of the packet size. However, as mentioned above, when the sending interval was too small, the packet loss sharply increased, and the receiving rate dropped accordingly.

We then performed experiments in an environment with competing connections. In order to set up a competing environment, we configured the bandwidth of 802.11b to 1Mb. Four additional FTP concurrent connections were opened between the client and the server. While using a sending rate at 64KBps, packet sizes were selected from [128, 256, 512, 1024, 2048, 4096, 8192, 16384] bytes. Figure 6.3 shows how the packet loss varied with the packet size under such conditions. When the packet size was very small or very large, the packet loss was large. The minimum loss occurred when the packet size was around 1,000.

Table 6.2: Effect of different packet sizes on sending/receiving rate and packet loss (without competing connections)

Packet Size (Byte)	Sending Interval (nano-second)	Sending rate (Bps)	Receiving rate (Bps)	Packet Loss (%)
65500	256000	255510	253852	0.00
32768	128000	255694	255694	0.00
16384	64000	255682	252899	0.78
8192	32000	255876	255637	0.00
4096	16000	255690	252476	0.98
2048	8000	255686	256480	0.00
1024	4000	255688	255374	0.00
512	2000	255724	255668	0.00
256	1000	255691	210535	16.58
128	500	255691	118407	53.69
64	250	255682	485	99.81
32	125	255401	499	99.80

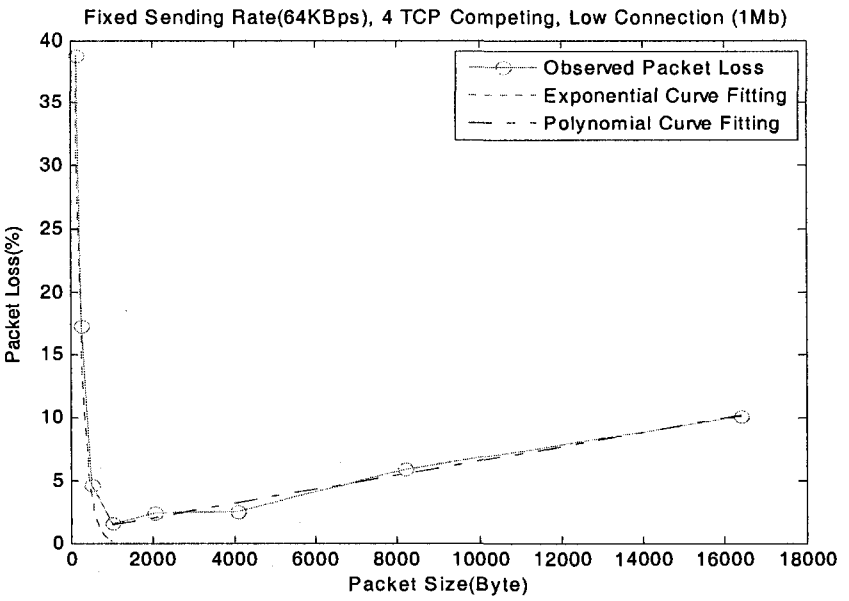


Figure 6.3: Packet loss vs. packet size in a congested network.

6.3 Packet Size Optimization

Even though various models for packet loss over wireless networks have been proposed [110][47], determining the optimal packet size has not received much attention. Larger packet sizes minimize the overhead from packet headers, but have a higher probability of being corrupted. Several studies have been published on packet size optimization based on different metrics, such as throughput, goodput, transmission range, and energy efficiency, in wired and wireless networks [111] [112] [113] [114] [115] [116] [117] [118]. Figure 6.3 reveals that in the competing environment when the packet size is small, the loss can be quite high because of congestion resulting from the need to route many packets. As the packet size gradually increases beyond an optimum point with low loss, the loss rate increases again. Unlike the researchers who have discussed packet size optimization based on different metrics, we propose an approach for estimating the optimal packet size to maximize the expected payload. We assume exponential and linear models for the packet loss depending on packet size, and this approach can be extended to other models as well.

For simplicity, we use the following assumptions and notation:

H: total network header size required for each packet;

S: amount of payload (application) data transmitted, if only 1 packet is used;

s_n: amount of payload data transmitted in each packet if n packets used ;

Probability that packet of size D is not lost is

$$e^{-\lambda D}; \tag{6.1}$$

i.e., an exponential packet loss model is used with parameter λ and the packet size as a variable.

Let the amount of data transmitted be $B = S+H$ when only 1 packet is transmitted, and $s_1 = S$. When 2 packets are transmitted, we have $B = 2s_2+2H$. Since

$$B = S+H = 2s_2+2H,\ s_2 = (S-H)/2.$$

In general,

$$s_n = (S-(n-1)H)/n\,. \tag{6.2}$$

For large n, i.e. when $(n-1)/n \approx 1$,

$$s_n \approx S/n - H\,. \tag{6.3}$$

Lemma 6.1: Following the model and assumptions above and independent transmission of packets, for large n the total expected payload received with n packets given $B=S+H$ is

$$f(n) \approx e^{\lambda H}\,[e^{-(\lambda S)/n}\,(S - nH)]. \tag{6.4}$$

Proof: Follows from the definitions, and noting that the total expected payload received equals the sum of the individual packet sizes times the probability that it is received:

$$f(n) = e^{-\lambda D}[n \quad (S/n - (n-1)H/n)\,]$$

$$= e^{-\lambda(S/n-H)}\,(S - nH).$$

Since $(n-1)/n \approx 1$ for large n,

$$f(n) \approx e^{\lambda H} [e^{-(\lambda S)/n} (S - nH)].$$

Theorem 6.1: The number of packets n optimizing the expected amount of payload transmitted for the exponential model, given $B=S+H$, is an integer equal to either

$$\left\lfloor \frac{S\sqrt{\lambda^2 H^2 + 4\lambda H}}{2H} - \frac{\lambda S}{2} \right\rfloor \text{ or } \left\lceil \frac{S\sqrt{\lambda^2 H^2 + 4\lambda H}}{2H} - \frac{\lambda S}{2} \right\rceil . \qquad (6.5)$$

Proof: Follows from optimizing the function in **Lemma 6.1** and the fact that the number of packets is an integer.

Now, suppose that the probability that a packet of size D is not lost is defined by $a - \lambda D$; i.e., a linear packet loss model is used with parameter λ and the packet size as variable. This model is more meaningful when the network characteristics follow the data in Figure 6.3.

Lemma 6.2: Following the linear model and the assumptions above and the independent transmission of packets, for large n the total expected payload received with n packets, given $B=S+H$, is

$$f(n) \approx [a - \lambda(S/n - H)] (S - nH) . \qquad (6.6)$$

Proof: Follows from the definitions and noting that the total expected payload received equals the sum of individual packet sizes times the probability that it is received:

$$f(n) = (a - \lambda D)[n \quad (S/n - (n-1)H/n)].$$

Since $(n-1)/n \approx 1$ for large n,

$$f(n) \approx [a - \lambda (S/n - H)](S - nH).$$

Theorem 6.2: The number of packets optimizing the expected amount of payload transmitted for the linear model given, $B=S+H$, is an integer equal to either

$$\left\lfloor S\sqrt{\lambda/(aH+\lambda H^2)} \right\rfloor \text{ or } \left\lceil S\sqrt{\lambda/(aH+\lambda H^2)} \right\rceil . \quad (6.7)$$

Proof: Follows from optimizing the function in **Lemma 6.2** and the fact that the number of packets is an integer.

Figure 6.3 reveals that for the first part, the curve fits a decreasing exponential function. If we consider only this part of the curve, the optimum point is the rightmost point because with increasing packet size, the overhead from the total header sizes of all packets is lower, and the packet loss is also lower.

For the second part, after the minimum point of the exponential part, we can fit a linear function $y = 0.8842 + 0.0006\ x$. Thus, the probability of a packet not being lost equals $(1-y/100) = 0.9912 - 0.000005683$ D; i.e., $a = 0.9912$ and $\lambda = 0.000005683$ in **Theorem 6.2**. Given S and H, we can determine the optimum number of packets by following **Theorem 6.2** for the linear section of the graph in Figure 6.3.

The network packet size in our optimization strategy is independent of the processing performed at the application level; no matter how the application data, e.g., the texture image and mesh information, are redistributed and segmented, the processed data, likely compressed, are passed to the network as a byte stream (compressed or uncompressed), which is then packed into the network packets. In our simulation, an IP header (8 bytes) and a UDP (20 bytes) are added to each network packet. 2 Mbytes of application data was used in our packet loss experiment, the result of which is plotted in Figure 6.3. Substituting S = 2M bytes and H = 28 bytes in Equation (6.7), we obtain the optimal number of packets by using **Theorem 6.2** to be either 904 or 905, corresponding to a packet size of about 2,212 bytes. This result shows that the optimal packet size for maximizing the payload (the actual multimedia data without packet headers) may not necessarily correspond to the packet size with the lowest loss rate. For this experiment, we assumed that the header for the multimedia data was not

included in the packets. If we consider duplicating multimedia header information in packets for increased reliability under packet loss, H in the formula will increase, giving a lower optimal number of packets or a higher optimal packet size for this example.

6.4 Conclusions

In this chapter, we studied packet loss in congested wireless networks by performing real-world experiments in a competing WLAN environment, which has packet losses and congestion. We carried out a comprehensive study of the impacts of the packet size, sending rate, sending interval, and buffer size on the packet loss. According to our experiments, first, the sending interval should be larger than 2000 nanoseconds. Second, the buffer size is large enough when it is larger than the packet size. Third, when competing connections are present, both too small and too large a packet size causes a larger packet loss. An appropriate packet size to minimize the packet loss can be determined. We proposed an approach for estimating the optimal packet size.

Chapter 7

TRANSMISSION AND RECONSTRUCTION OF ARBITRARY 3D MODELS OVER UNRELIABLE NETWORKS

In the last chapter, we have presented our work on the packet loss experiments and the optimal packet size model. In the next two chapters, we will present the work on 3D transmission over unreliable networks. 3D transmission over unreliable networks needs to account for the possibility of packet loss. The existing research uses either retransmission or ECC to deal with packet loss. In this chapter, we propose to transmit and reconstruct the geometry data of a 3D mesh model over a lossy channel without retransmission and ECC since retransmission is not applicable or preferred in real-time applications, and ECC results in significant overhead if the error rate is high. Our strategy distributes neighboring vertex information into different packets to minimize the risk of packet loss affecting a large neighborhood. We also present an interpolation scheme with different neighbor levels and in the order of decoding. Our experiments on models with different densities show that the smoothness on the mesh's surface deteriorates only at packet loss of more than 75%, and that the object is still recognizable. We also compare the triangle-based linear spline, triangle-based cubic spline and 'v4' interpolation methods. Among them, the triangle-based cubic spline interpolation method is found to perform best overall. Our interpolation scheme is found to perform significantly better than the subdivision-based approach.

7.1 Introduction

With a 3D model, a user can rotate and zoom in/out to view an object from many different viewpoints, whereas doing so is not possible for with a 2D presentation. Therefore, a 3D model can be widely used in online applications including online shopping, online museums, and online education. For online applications, a 3D model is transmitted online with varied limited bandwidths and within a constrained time

which the user can tolerate. Therefore, a 3D model should be transmitted in a compressed, error-resilient and scalable way.

One of the major drawbacks with most 3D transmission algorithms is that they do not consider the possibility of packet loss over lossy channels. For wireless channels, where packet loss occurs as a result of fading and interference [1] [2], the TCP strategy leads to further delays and degradation of the transmission quality. Even though issues of multimedia transmission over wireless networks have been discussed [119] [47], relatively little work has been done to address 3D transmission over lossy channels. In recent studies, approaches for the robust transmission of 3D meshes over lossy channels [17] [24] are outlined. In [24], it is assumed that some parts of the mesh can be transmitted without loss, allowing progressive mesh transmission to give good results. However, for lossy channels, this assumption implies retransmission, which is not feasible in an environment where retransmission is not possible or preferred, such as in real-time or time constraint applications. In [17], an adaptive scheme of error correction codes (ECC) is applied to correct errors in different error rates. ECC schemes require additional bits to correct errors, and this requirement might result in significant overhead if the error rate is high. To transmit the geometry data of 3D meshes over a lossy channel without retransmission and ECC, we assume packet-based transmission where a certain percentage of the packets may be lost. In this chapter, we study the transmission strategy of vertex geometry data to minimize the risk of packet loss affecting a large neighborhood, and also describe an interpolation scheme to better reconstruct the meshes. We conduct experiments on models of different densities with different percentages of packet loss and different neighbor levels of interpolation. Our experiments show the efficiency of our scheme. We also compare the triangle-based linear spline, triangle-based cubic spline and 'v4' interpolation methods. Among them, the triangle-based cubic spline interpolation method performs best overall. Our interpolation scheme performs significantly better than the subdivision-based approach.

The remainder of this chapter is organized as follows: Section 7.2 presents our work on a transmission strategy to minimize the risk of packet loss affecting a large neighborhood. In Section 7.3, the proposed interpolation scheme is described.

Experimental results, comparison and analysis are presented in Section 7.4. Finally, Section 7.5 presents our conclusions.

7.2 Transmission Strategy

When transmitting irregular mesh data, not only vertex information but also connectivity information plays a crucial role in 3D reconstruction at the client site. In order to preserve the original geometry of the object, many transmission algorithms suggest retransmission of the base layers to safeguard the successful transmission of important features of the object [17] [24]. Retransmission adds an overhead on bandwidth limited connections, particularly on wireless and mobile networks. Without needing to retransmit the base layer, we wish to find a trade-off between the compression rate and robustness to packet loss. For example, although the Edgebreak 3D mesh coding method discussed in Section 2.1.5 has a high compression ratio, the cow object (Figure 2.5) shows significant distortion even when only one character in the connectivity chain is lost. Most mesh compression algorithms, including the Valence-driven method [44] [19] and Edgebreak 3D mesh coding method [31] [40], are designed for reliable transmission, and, therefore, a high compression ratio can be achieved by adopting an incremental strategy; subsequent information is useful only if previous information is correctly received. Any discontinuity in the data received makes the rest of the mesh difficult to reconstruct. In our strategy, we focus on the following criteria:

We distribute the neighboring vertex and connectivity information into different packets to minimize the risk of lost data affecting a large neighborhood. Let the total number of packets transmitted be p. Starting from the first vertex, traverse the vertices as in the valence-driven approach [44]. The first p vertices are distributed to p different packets. The process is repeated with the next p vertices, and so on. In other words, the possibility of lost adjacent vertices creating a large void region is reduced. With any loss of the connectivity information, it can make the reconstruction impossible. The connectivity information, which has a size of roughly 10% of the vertex information, is transmitted in duplicate packets to decrease the loss probability.

Figure 7.1 shows an example of how the vertices are distributed into different packets. The vertices are encoded and colored in the order of 1, 2, …, 16. The vertices labeled by the same number are distributed in the same packet. For example, the vertices labeled by 1 are grouped in the first packet, and the ones labeled by 2 are in the second packet.

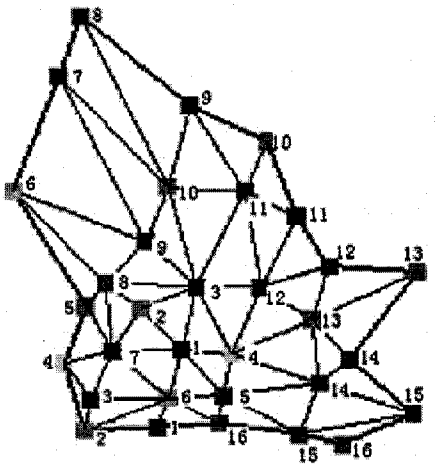


Figure 7.1: Encoding order and packet grouping.

7.3 Proposed Interpolation Scheme

7.3.1 Interpolation Scheme of Loss Geometry

After the mesh data are received, first, the meshes are reconstructed based on the received geometry and connectivity in the same order as the encoding. Second, the vertices are traversed in the reconstructed order of the valence-driven decode algorithm. When a vertex with loss geometry, L, is encountered, the neighbors of L, whose geometry is either not lost or interpolated previously, are used to interpolate the geometry of L. The interpolation procedure is as in **Algorithm 7.1**.

We studied and compared the triangle-based linear, triangle-based cubic spline and 'v4' [51] interpolation methods implemented in Matlab's 'griddata' function [57]. The function 'griddata' fits a surface to the non-uniformly spaced 3D points. For triangle-based linear and cubic methods, first the Delaunay triangulation is applied on the neighbor vertices. Next, a linear or cubic polynomial is found on each triangle to fit a surface. The 'v4' method produces the same surface as the cubic spline method if the

Algorithm 7.1: Interpolation Procedure

```
for each of vertex v[i] in the reconstructed order of the valence-driven decode algorithm {
    if (the geometry of v[i] is lost) {
        add the neighbors of v[i], whose geometry is either not lost or interpolated
            previously, into the neighbors list;
        apply an interpolation method to the neighbors list to get the geometry of v[i];
        lable v[i] is interpolated;
    }
} □
```

slopes of the end data points are also constrained to be zero. This method is relatively inefficient and can be unstable. Its major advantage is that the slope can be used as input. The 'v4' method is based on the Green function of the biharmonic operator. Green functions usually are used to solve linear ordinary and partial differential equations. The interpolating surface is a linear combination of Green functions with the center at each data point. The Green function's amplitudes are found so that the interpolating surface passes through the data points. After fitting a surface, we pick the surface point, whose x and y coordinates are equal to the mean x and y coordinates of the neighbors, as the interpolated lost vertices.

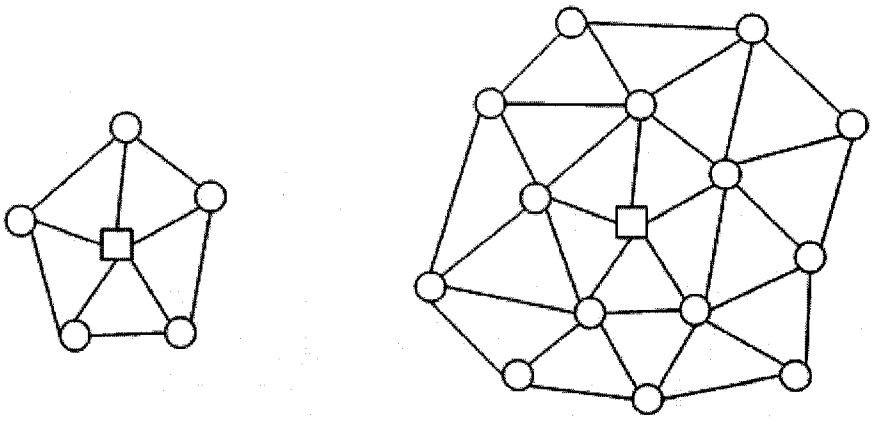
7.3.2 Neighbor Level

To reconstruct the lost vertex, we interpolate its neighbors. One issue is how many of its neighbors should be used for interpolation. We use the term *neighbor level* to refer to how further the lost vertex's neighbors are used. As Figure 7.2 shows, if the neighbor level is one, only the direct neighbors with connected edges to the lost vertex are used. If the neighbor level is two, the direct neighbors with connected edges to the lost vertex and the direct neighbors of the lost vertex's direct neighbors are used.

7.4 Experimental Results

7.4.1 Examples of Distributed Transmission Strategy

Figure 7.3 shows the comparison of whether packet loss is distributed or not. This figure shows that if the packet loss is not distributed, a whole part of one leg cannot be reconstructed. Our distributed scheme avoids this problem.



(a) (b)

Figure 7.2: Neighbor level. Circle vertices are neighbors used to interpolate the lost vertex. Square vertex is the interpolated lost vertex. (a) Neighbor level=1; (b) Neighbor level=2.

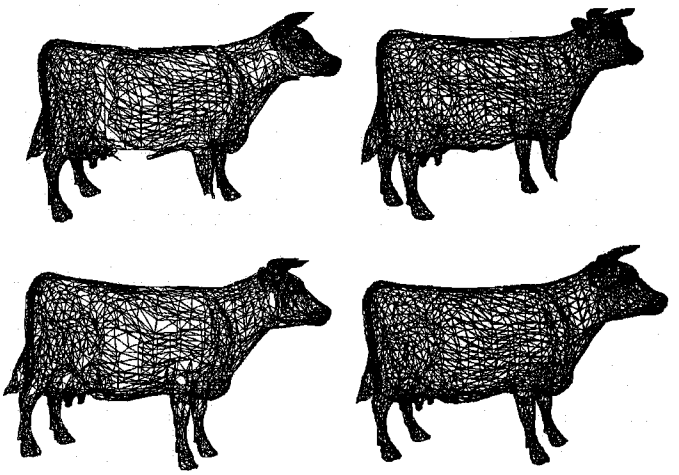


Figure 7.3: Comparison of whether packet loss is distributed or not. 1st row: packet loss not distributed (1 out of 16 packets loss); 2nd row: packet loss distributed (1 out of 16 packets loss). Left: before interpolation; Right: after linear interpolation (neighbor level=1).

7.4.2 Examples of Interpolation Scheme

Figure 7.4 shows an example of the Cow meshes before interpolation after transmission in different packet numbers and with different percentages of packet loss. Figure 7.5 shows the Cow meshes after linear interpolation with a neighbor level equal

to 1. Figure 7.6 gives the meshes mapped with color after linear interpolation with a neighbor level equal to 1. This figure shows that after transmitting in both 4 and 16 packets, the smoothness on the reconstructed object's surface after interpolation deteriorates only at 75% packet loss, and that the object is still recognizable as a cow.

Figure 7.7 shows more examples which are transmitted in 16 packets. The reconstruction quality after transmission with packet loss depends on the original density of the model. The Dinosaur meshes have a high density, with 14070 vertices, whose reconstruction quality with 75% packet loss is still quite good. The reconstructed models of the coarse Queen (650 vertices) and Body (711 vertices) meshes with 75% packet loss deteriorate more. If a projecting vertex is lost, the corresponding part will be flattened. For example, the top of the reconstructed Queen with 25% packet loss (Figure 7.7 1st row, 2-3rd column) is flattened owning to the loss of the top projecting vertex.

7.4.3 Comparison of Different Interpolation Methods

We conducted our experiments by applying the triangle-based linear, triangle-based cubic spline and 'v4' [51] interpolation methods with different neighbor levels on nine models. The nine models had different densities, whose number of vertices varied from 428 to 5000. We considered the different levels of packet loss as well. The numbers of lost packets (out of 16) in the experiments are 4, 8 and 12. We used the metro tool [52] to measure the error between the original model and the reconstructed model by using Hausdorff distance. The metro tool is based on surface sampling and *point-to-surface* distance computation. It samples vertices, edge and faces by taking a number of samples approximately 10 times the face number.

Table 7.1 shows the reconstructed error comparison of the experimental results. This table reveals that the triangle-based cubic spline interpolation method with the neighbor level equal to 1 is the best overall and has the most amount of the cases which have the minimal reconstructed errors. The 'v4' method performs significantly worse because the number of data points are not large enough, and the slopes of the end data points are not constrained to be zero.

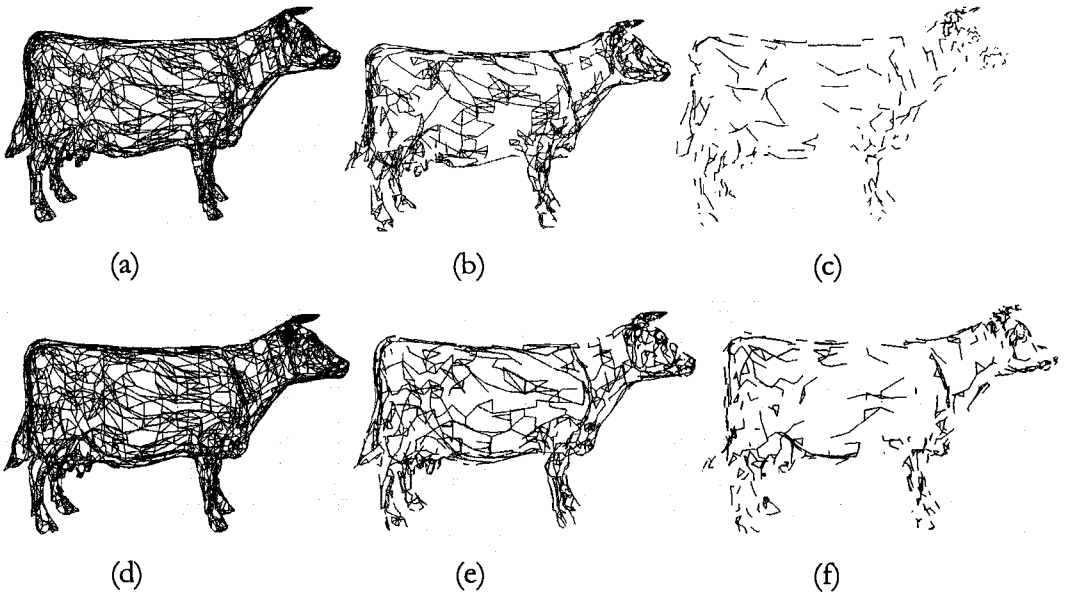


Figure 7.4: Before interpolation.

(a) 4 lost out of 16 packets; (b) 8 lost out of 16 packets; (c) 12 lost out of 16 packets; (d) 1 lost out of 4 packets; (e) 2 lost out of 4 packets; (f) 3 lost out of 4 packets.

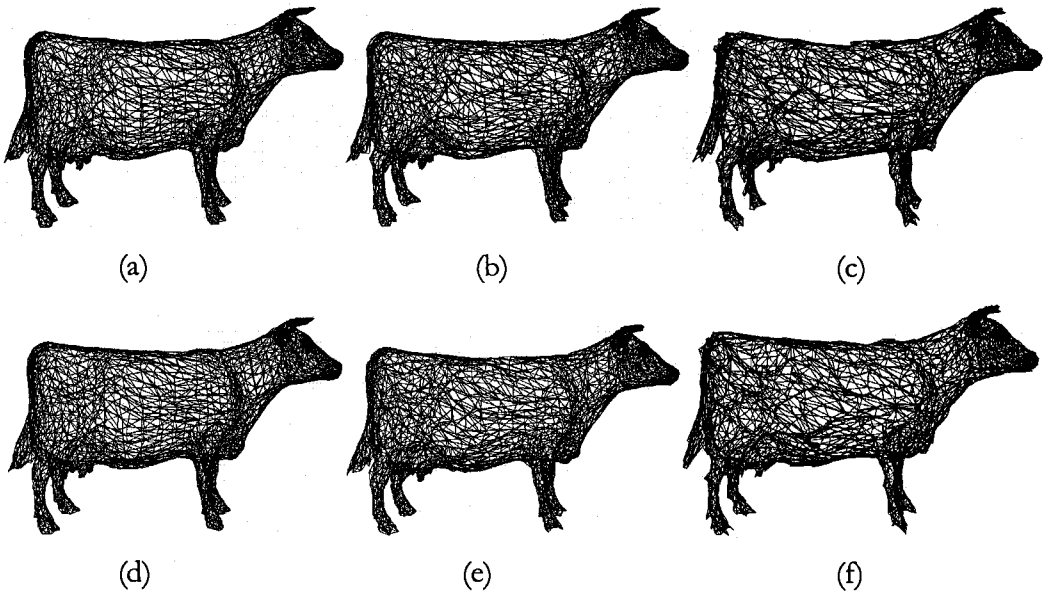


Figure 7.5: After linear interpolation (neighbor level=1).

(b) 4 lost out of 16 packets; (b) 8 lost out of 16 packets; (c) 12 lost out of 16 packets; (d) 1 lost out of 4 packets; (e) 2 lost out of 4 packets; (f) 3 lost out of 4 packets.

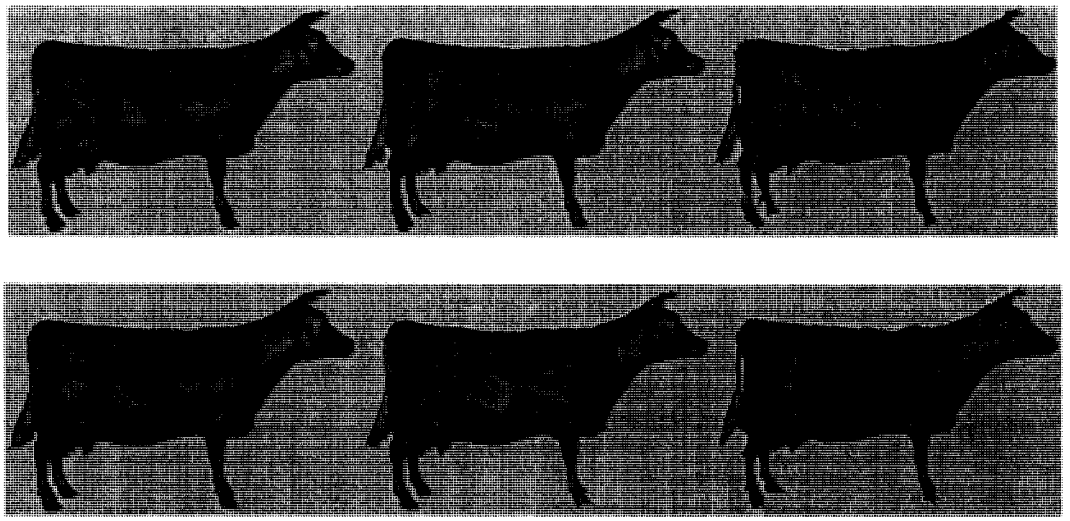


Figure 7.6: Meshes mapped with color after linear interpolation (neighbor level=1).

(c) 4 lost out of 16 packets; (b) 8 lost out of 16 packets; (c) 12 lost out of 16 packets; (d) 1 lost out of 4 packets; (e) 2 lost out of 4 packets; (f) 3 lost out of 4 packets.

7.4.4 Comparison with Other Approaches

One objective of this work is to reconstruct the surface of the 3D meshes after a transmission with packet loss and without retransmission. A possible approach is to reconstruct from the oriented point sets [54]. With this approach, only the coordinate and normal of points without connectivity information are transmitted. From the received coordinate and normal of points, the surface of 3D meshes can be reconstructed with some points lost. One disadvantage of this approach is that the reconstructed meshes can be in many pieces if the points are sparse. Unlike this approach, our approach transmits connectivity information and can work well on sparse meshes.

Another approach is to reconstruct the surface from the partially received meshes by using subdivision methods such as the Catmull-Clark subdivision method [55] and the sqrt(3) subdivision method [56]. The surface subdivision method is usually used to generate a denser and smoother surface from a coarse surface. More than one vertices

are added, and their coordinates are interpolated when doing surface subdivision. With the Catmull-Clark subdivision method [55], the coordinates of the added vertices are interpolated by using the cubic spline scheme. The sqrt(3) subdivision method [56] differs from other subdivision methods by increasing the number of triangles in every step by a factor of 3 instead of 4.

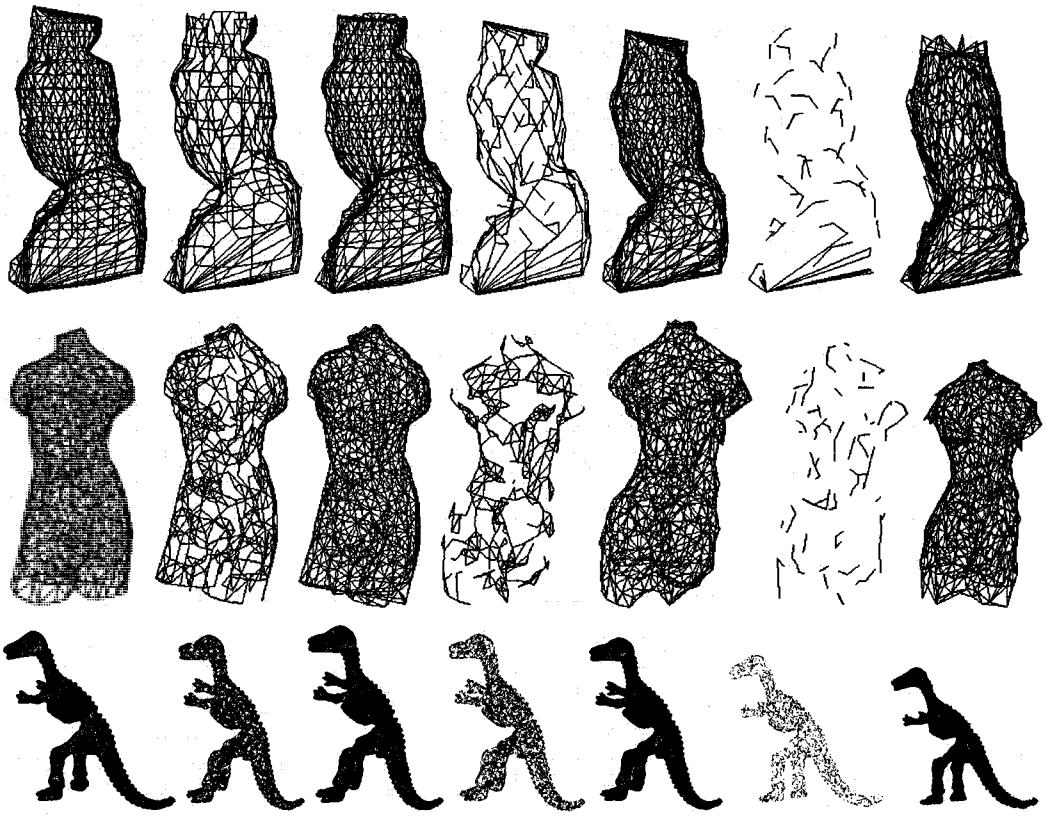


Figure 7.7: Different models: queen(1st row, 650 vertices);body(2nd row, 711 vertices); dinosaurs(3rd row, 14070 vertices).

1st column: original model;

2nd column: 4 loss out of 16 packets(before interpolation); 3rd column: 4 loss out of 16 packets(after linear interpolation, neighbor level=1);

4th column: 8 loss out of 16 packets(before interpolation); 5th column: 8 loss out of 16 packets(after linear interpolation, neighbor level=1);

6th column: 12 loss out of 16 packets(before interpolation); 7th column: 12 loss out of 16 packets(after linear interpolation, neighbor level=1).

Table 7.1: Comparison of different interpolation methods. The numbers marked with (*)are the minimal error in the reconstructed models for the same model with the same number of lost packets. ('-' means a value larger than 100,000.)

Number of Lost Packets (out of 16) = 4

	Reconstruction Error (Hausdorff distance of metro tool[52])					
	Interpolation Method :Linear		**Interpolation Method** :Cubic		**Interpolation Method** :V4	
Model(Vertex Number)	**Neighbor Level =1**	**Neighbor Level =2**	**Neighbor Level =1**	**Neighbor Level =2**	**Neighbor Level =1**	**Neighbor Level =2**
Armadillo (1752)	9.044372(*)	10.687811	9.298910	10.316089	21523.947266	31.398216
Body(711)	0.293317	0.308802	0.282268(*)	0.293965	0.358016	0.289887
Bunny(2503)	0.003009(*)	0.003759	0.003010	0.003927	0.034500	0.010478
Cow(2904)	0.025903	0.032442	0.025100(*)	0.034776	0.058983	0.030875
Dinosaur(5000)	1.617305	1.703516	1.462922(*)	2.316251	628.801147	155.185516
Dragon(1252)	9.619162	9.619162	9.619162(*)	9.619162	1975.683105	20.720869
HammerHead(752)	0.025389	0.030961	0.025343(*)	0.031520	0.867992	0.701022
Mannequin(428)	0.274351(*)	0.368580	0.299820	0.405500	0.629864	0.463766
Queen(650)	0.112574	0.200955	0.111644(*)	0.187389	0.192037	1.974105

Number of Lost Packets (out of 16) = 8

	Reconstruction Error (Hausdorff distance of metro tool[52])					
	Interpolation Method :Linear		**Interpolation Method** :Cubic		**Interpolation Method** :V4	
Model(Vertex Number)	**Neighbor Level =1**	**Neighbor Level =2**	**Neighbor Level =1**	**Neighbor Level =2**	**Neighbor Level =1**	**Neighbor Level =2**
Armadillo (1752)	14.016294	14.246200	14.014572(*)	14.210988	-	244.568726
Body(711)	0.324557	0.343627	0.323471	0.310863(*)	0.593218	0.326924
Bunny(2503)	0.004593(*)	0.005291	0.004615	0.005471	0.025934	0.033867
Cow(2904)	0.032304	0.034303	0.032176(*)	0.036055	0.082749	0.057212
Dinosaur (5000)	2.868300	3.228362	2.855550(*)	3.401511	-	99629.3438
Dragon(1252)	15.491241	16.276470	15.459133(*)	16.276007	-	35.064377
HammerHead(752)	0.065599	0.070335	0.065599(*)	0.071985	0.293371	1.191587
Mannequin(428)	0.469657(*)	0.494803	0.478435	0.495934	0.710001	0.590717
Queen(650)	0.187299	0.226249	0.177390(*)	0.227999	0.278772	2.211618

Number of Lost Packets (out of 16) = 12

	Reconstruction Error (Hausdorff distance of metro tool[52])					
	Interpolation Method :Linear		**Interpolation Method** :Cubic		**Interpolation Method** :V4	
Model(Vertex Number)	**Neighbor Level =1**	**Neighbor Level =2**	**Neighbor Level =1**	**Neighbor Level =2**	**Neighbor Level =1**	**Neighbor Level =2**
Armadillo (1752)	22.997353	15.513267(*)	23.019205	15.606725	-	5750.969727
Body(711)	0.615563	0.649467	0.615522(*)	0.643023	0.698486	1.586106
Bunny(2503)	0.008582	0.010518	0.008444(*)	0.010552	0.046456	0.023250
Cow(2904)	0.047938	0.052571	0.047795(*)	0.054300	0.153599	0.067901
Dinosaur (5000)	4.843575(*)	5.023284	4.888701	5.023284	-	-
Dragon(1252)	15.515945(*)	17.569109	17.091991	17.569109	-	788.059631
HammerHead(752)	0.121758	0.118163(*)	0.122472	0.123254	1.093346	0.693335
Mannequin(428)	0.673878	0.776635	0.670714(*)	0.765230	0.896839	0.896235
Queen(650)	0.301478	0.269726	0.302202	0.262581(*)	0.279607	3.439980

Table 7.2: Comparison with subdivision-based Approach. The numbers marked with (*)are the minimal error in the reconstructed models for the same model with the same number of lost packets.

	Reconstruction Error (Hausdorff distance of metro tool[52])			
	# of Lost Packets (out of 16) = 1		# of Lost Packets (out of 16) = 2	
Model(Vertex Number)	Sqrt(3) Subdivision	Proposed Approach(Interpolation Method:Cubic Neighbor Level =1)	Sqrt(3) Subdivision	Proposed Approach(Interpolation Method:Cubic Neighbor Level =2)
Armadillo (1752)	6. 529486	2. 923688(*)	12.067476	3. 829655(*)
Body(711)	0.251311	0.115403(*)	0.513330	0.213674(*)
Bunny(2503)	0.003671	0.001699(*)	0.010211	0.002416(*)
Cow(2904)	0.028814	0.013196(*)	0.057889	0.013489(*)
Dinosaur (5000)	1.773165	0.770778(*)	3.733543	1.275865(*)
Dragon(1252)	8.214386	5. 994485(*)	12.851923	8.348866(*)
HammerHead(752)	0.024065	0.010686(*)	0.060305	0.011494(*)
Mannequin(428)	0.328453	0.146919(*)	0.752031	0.259950(*)

We compared the proposed approach with the subdivision-based approach by carrying out experiments. With packet loss, the coordinates of the partial vertices are lost, resulting in holes in the meshes. Before applying the subdivision method to reconstruct the 3D meshes, we closed the hole by covering it with a new polygon whose shape was defined by the hole's boundaries. The added polygon is not planar if its vertices are not in a plane. If the coordinates of too many vertices are lost, the holes in 3D meshes cannot be closed. Therefore, the experiments were conducted for only two cases: 1 or 2 packets lost out of 16 packets. Table 7.2 shows the experiments' results, comparing the sqrt(3) subdivision-based approach with the one-step subdivision and the proposed approach. This table shows that the proposed approach has significantly lower reconstructed errors for all cases. We also observe that Catmull-Clark subdivision-based method and sqrt(3) subdivision-based method perform similarly and that the reconstructed error is not decreased significantly by carrying out more steps in the subdivision (Table 7.3).

Table 7.3: Comparison with different subdivision methods and subdivision steps. The test model is Cow.

Subdivision Method	Subdivision Step	Reconstruction Error
Sqrt(3) Subdivision	1	0.028814
	2	0.028742
	3	0.028725
	4	0.028646
Catmull-Clark Subdivision	1	0.028683

7.5 Conclusions

In this chapter, we proposed the strategy of distributing neighboring vertex information into different packets to minimize the risk of losing data affecting a large neighborhood, and we described an interpolation scheme with different neighbor levels in the order of decoding. Our experiments on models with different densities showed that the smoothness on the mesh's surface deteriorated only at 75% packet loss, and that the object was still recognizable. The reconstruction quality after transmission with packet loss depends on the original density of the model. If a projecting vertex is lost, the corresponding part will be flattened. We also compared the triangle-based linear spline, triangle-based cubic spline and 'v4' interpolation methods. Among them, the triangle-based cubic spline interpolation method performed best overall. Our interpolation scheme performed significantly better than the subdivision-based approach.

Chapter 8

PROBABILISTIC MESH-DEPENDENT INTERLEAVING FOR 3D TRANSMISSION OVER BURST PACKET LOSS CHANNELS

In the last chapter, we have discussed the 3D transmission over unreliable networks without considering burst packet loss. In a communication network, packet loss is often bursty. In order to minimize the impact of burst errors on the transmitted data, interleaving is an efficient strategy. The existing interleaving techniques for 1D sequences and 2D images are not suitable for arbitrary meshes because 3D vertices are unevenly distributed on a mesh surface. A predefined distance or pattern used in 1D or 2D data interleaving is not effective because it may work on one geometry but fail on another. In this chapter, we propose a new probabilistic mesh-dependent interleaving strategy, which is dependent on the mesh geometry, to minimize the adverse effect of burst packet loss. The goal is to minimize the probability of multiple lost vertices adjacent to each other resulting in an unacceptable reconstructed mesh at the receiver. Our probabilistic mesh-dependent approach adjusts two interleaving control parameters which are dictated by the geometric property of the mesh (different adjacent distance histograms) and by the maximum burst length to be considered. Our experimental results verify the efficiency of our method.

8.1 Introduction

In many communication situations, the losses are bursty [134] [133] [119]. This problem can be caused by congestion on the Internet. Burst packet loss over wireless networks can be caused by temporary link outage or fading-induced bit error. In order to minimize the adverse effect of burst errors, an interleaving technique aims to prevent errors from affecting a large neighborhood. Examples of interleaving techniques can be found in the literature [126], but their focus is on 1D sequences and

2D images. The strategy is to apply a predefined distance to separate neighboring elements in the transmitted stream independent of which 1D sequences or 2D images are being transmitted. Due to the arbitrary distribution of vertices on a 3D mesh, a predefined distance is not effective because it may work on one geometry but fail on another. In this chapter, we propose a mesh-dependent interleaving scheme for transmitting arbitrary 3D meshes over burst packet loss channels. The main idea is to adjust the control parameters following the geometric property of a 3D mesh based on the maximum burst length being considered. We conducted experiments to verify the efficiency of the proposed method.

The remainder of this chapter is organized as follows: Section 8.2 reviews interleaving strategies and proposes our mesh-dependent interleaving approach. Our experimental results and analysis are presented in Section 8.3. Finally, our conclusions are presented in Section 0.

8.2 Adjacent Distance and Mesh-Dependent Interleaving

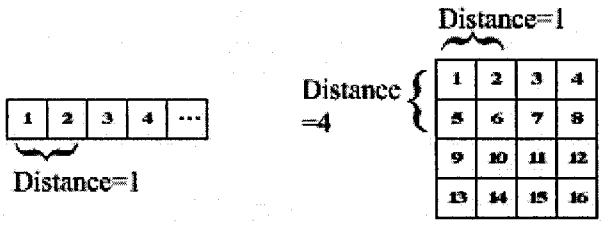
8.2.1 Encoding in 1D, 2D and 3D Data

By distributing burst errors into random-like errors, interleaving efficiently minimizes the adverse effect of burst errors [126]. A 1-D interleaving technique on error-correction coding was documented by Wicker [127], and some multiple dimensional (M-D) interleaving techniques were also proposed [128][129][130][126][131]. United Parcel Service (UPS) developed its own 2D bar code protection system [128]. In this method, a sequence of code symbols is first 1D interleaved. The 1D interleaved symbols are then written into a 2D array according to a Boustrophedonic or Spiral Data writing pattern. Zhang et al. [126] and Xu et al. [131] proposed optimal 2-D interleaving methods by introducing the interleaving distance defined by them.

Current interleaving schemes are suitable for data elements arranged systematically, such as in rows and columns, so that neighboring elements can be identified based on their traversal order and distributed into separate packets. Examples of the traversal orders in 1D sequences and 2D images are given in Figure 8.1. In a 1D sequence,

adjacent elements are traversed one after another. We define *adjacent distance* as the relative traversal order between two elements. In this case, the adjacent distance between neighboring elements is 1. Burst errors can be minimized by imposing an adjacent distance greater than 1 between elements in the same packet. For 2D images, the elements are traversed following the row and column order. In Figure 8.1 (b), the adjacent distance between horizontal neighbors is 1 and between vertical neighbors is 4. By putting elements of adjacent distance other than 1 and 4 in the same packet, we can avoid losing neighboring elements.

Interleaving techniques for 1D and 2D data which have fixed adjacent distance are not suitable for 3D meshes because of the geometry-dependant traversal pattern. An example using valence-driven encoding is given in Figure 8.2. A vertex is drawn as a node in the graph. The traversal order of the vertices is indicated by the numbers in the nodes. The adjacent distance between two connected vertices is written on the edge. Unlike 1D and 2D data, a 3D vertex is associated with various adjacent distances. In Figure 8.2, the neighbors of Vertex 20 are Vertex 21, 22, 54, 60, and 99, whose adjacent distances relative to Vertex 20 are 1, 2, 34, 50 and 79, respectively.



(a) (b)

Figure 8.1: Diagram of distance between neighboring elements in 1D sequence (a) and 2D image (b).

To illustrate how adjacent distances vary between connected vertices on arbitrary meshes, we compute the edge count (in descending order) associated with a particular adjacent distance for two 3D meshes models: Cow and HammerHead (Table 8.1). Observe that most edges have the shortest adjacent distance, 1 in this case, and that different meshes have different adjacent distance frequencies (See Edge Count column in Table 8.1). Figure 8.3 shows the histograms without the adjacent distance equal to 1.

Motivated by this finding, we introduce a probabilistic interleaving algorithm, which adjusts a control parameter for different mesh geometries (different adjacent distance histograms) and takes the maximum burst length into consideration.

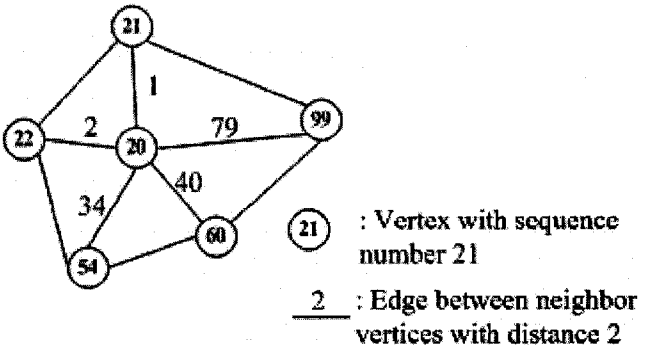


Figure 8.2: Diagram of neighboring vertex distance.

Table 8.1: Histogram of 3D meshes models.

	Cow			**HammerHead**		
index	Adjacent Distance	Edge Count	Cum. Percent	Adjacent Distance	Edge Count	Cum. Percent
0	1	2,865	32.9	1	721	32.0
1	13	282	36.1	2	66	35.0
2	12	279	39.4	9	54	37.4
3	14	210	41.8	12	51	39.7
4	11	197	44.0	10	48	41.8
5	2	162	45.9	11	48	43.9
6	15	115	47.2	49	48	46.0
7	10	105	48.4	43	47	48.1
8	97	74	49.3	42	44	50.1
9	98	71	50.1	48	43	52.0
.	99	69	50.9	8	42	53.9
.	96	68	51.7	50	42	55.7
.	82	65	52.4	13	40	57.5
.	16	63	53.1	39	40	59.3
.	88	59	53.8	46	39	61.0
.	89	58	54.5	47	36	62.6
.	100	56	55.1	45	31	64.0
.	109	54	55.7	40	30	65.3
.	...	...		...	...	
i	...	...		...	...	
Total Edges		8,706	100		2,250	100

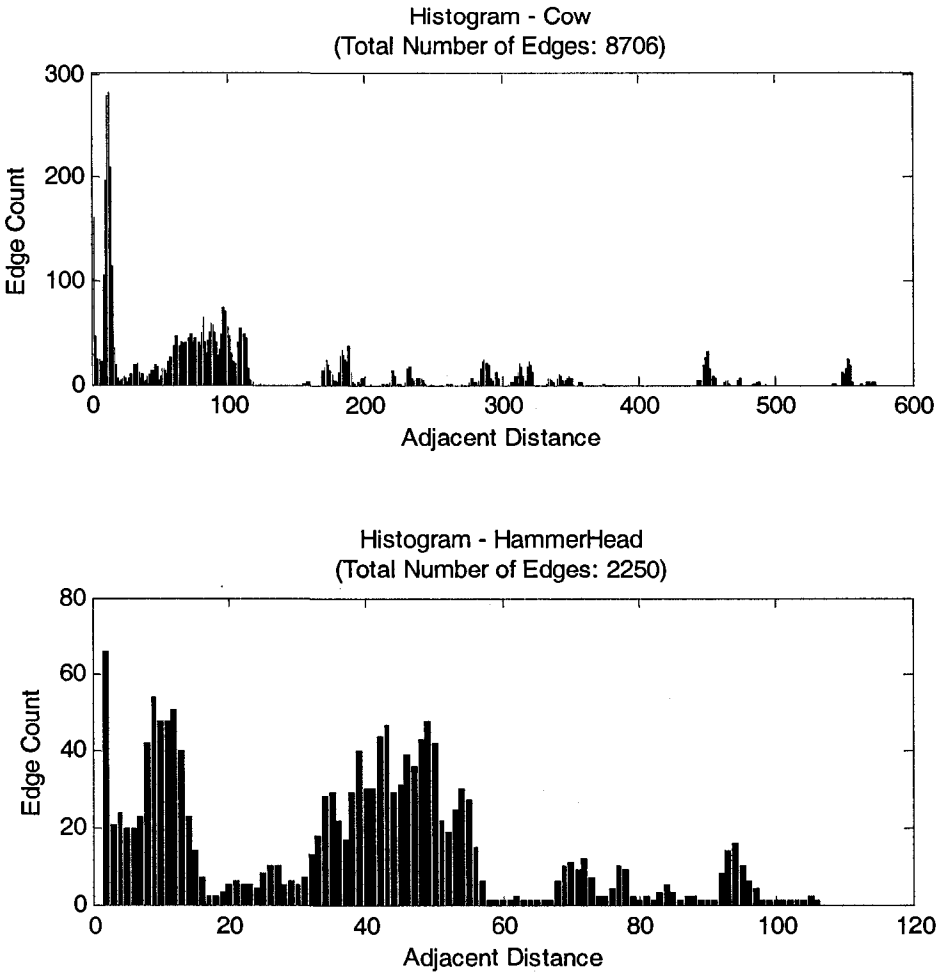


Figure 8.3: Histogram of 3D meshes models (Cow and HammerHead).

Although we use the valence-driven algorithm as an example to illustrate the traversal pattern on irregular meshes, our mesh-dependent interleaving technique can work with any mesh encoding algorithm.

8.2.2 Proposed Probabilistic Mesh-Dependent Interleaving

In order to minimize the adverse effect of burst packet loss, neighboring vertices are interleaved into separate packets. An example of how our interleaving technique works is shown in Figure 8.4. We follow the valence driven algorithm and label the vertices in traversal order, starting from 0. The network packets are denoted by PKT0, PKT1 .... The interleaving pattern is controlled by two parameters: B and L. Parameter B is the

maximum burst length considered and is used to define the number of packets in a block. The other parameter L is a control parameter computed using Algorithm 8.1 described below. L and the product $L \times B$ are used to compute the minimum distance required between elements in the same block. The interleaving pattern using B=2 and L=3 is given in Figure 8.4. Blocks are separated by dotted lines. When the maximum burst length of 2 occurs, this interleaving pattern ensures that the neighbor of a lost vertex survives the burst.

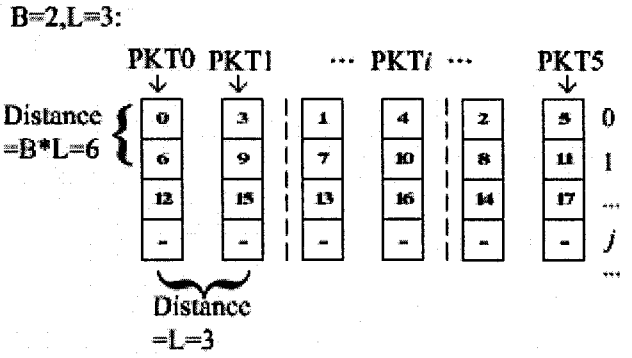


Figure 8.4: Vertex interleaving for network packets by using our probabilistic mesh-dependant interleaving strategy.

Let PKT_i be a network packet where i (i=0, 1, 2, ...) is the sequence number, and j (j=0, 1, 2, ...) is the position number of a vertex in a network packet. The blocks of packets can be viewed as a matrix with j rows and i columns. We can associate a vertex label ℓ (traversal order) with a (i,j) coordinate by using the following formula:

$$\ell = j \times B \times L + (i \bmod B) \times L + \left\lfloor \frac{i}{B} \right\rfloor . \tag{8.1}$$

For example, in Figure 8.4, the label ℓ of the 3rd vertex in packet 5, i.e., i=4, j=2, is calculated as follows:

$$\begin{aligned} & 2 \times B \times L + (4 \bmod B) \times L + \left\lfloor \frac{4}{B} \right\rfloor \\ &= 2 \times 2 \times 3 + (4 \bmod 2) \times 3 + \left\lfloor \frac{4}{2} \right\rfloor \\ &= 14 . \end{aligned}$$

The $j \times i$ matrix has L blocks, and each block has B packets. In Formula (8.1), the first term $j \times B \times L$ indicates that each row has $B \times L$ elements in the $j \times i$ matrix. $(i \bmod B)$ in the second term $(i \bmod B) \times L$ indicates the sequence number of a packet in each block, and L indicates the adjacent distance of the neighboring corresponding vertices in each block. The third term $\left\lfloor \frac{i}{B} \right\rfloor$ indicates the block number in the matrix. We have the following lemma.

Lemma 8.1: The adjacent distance between the vertices in a same block is a multiple of L, and the adjacent distance of the vertices in each packet is a multiple of $B \times L$.

Proof: Suppose two vertices in a same block are (j_1, i_1) and (j_2, i_2). For the vertices in the same block, we have $\left\lfloor \frac{i_1}{B} \right\rfloor = \left\lfloor \frac{i_2}{B} \right\rfloor$. Therefore, the adjacent distance between the vertices in a same block is

$$\begin{aligned} & | j_1 \times B \times L + (i_1 \bmod B) \times L + \left\lfloor \frac{i_1}{B} \right\rfloor \\ & - (j_2 \times B \times L + (i_2 \bmod B) \times L + \left\lfloor \frac{i_2}{B} \right\rfloor) | \\ & = | (j_1 - j_2) \times B + (i_1 \bmod B) - (i_2 \bmod B) | \times L, \end{aligned}$$

which is a multiple of L.

The vertices in each packet have the same i. Suppose the two corresponding vertices are (j_1, i) and (j_2, i). Therefore, the adjacent distance between the vertices in each packet is

$$\begin{aligned} & | j_1 \times B \times L + (i \bmod B) \times L + \left\lfloor \frac{i}{B} \right\rfloor \\ & - (j_2 \times B \times L + (i \bmod B) \times L + \left\lfloor \frac{i}{B} \right\rfloor) | \\ & = | j_1 - j_2 | \times B \times L, \end{aligned}$$

which is a multiple of $B \times L$. □

Note that the above interleaving pattern illustrates the minimum number of packets that satisfies the adjacent distance restriction. A higher number of packets can be transmitted by partitioning each block horizontally and by dividing each packet (PKT_i) into smaller packets.

8.2.3 Computation of Control Parameter L

We put B packets in a block, inside which the adjacent distance between corresponding elements at the same j position in consecutive packets is L. For example, in Figure 8.4, PKT0 and PKT1 belong to a block, while PKT2 and PKT3 belong to another block. When a burst error occurs, the maximum loss is one block or two consecutive packets. The parameter L is computed so that vertices in the same neighborhood are distributed into different blocks. If a burst error occurs and PKT0 and PKT1 are lost, then so is vertex 3, but vertex 3 can be reconstructed based on the geometric information of its neighbors which are not lost. However, if its neighbors are transmitted in the same block, then a larger piece of the surface geometry is lost, and reconstruction based on interpolation will be less accurate. To computer the proper L, we have the following theorem.

Theorem 8.1: If all of the adjacent distances between any neighboring vertices in the meshes are not a multiple of L, the vertices in the same block are not neighboring in the meshes.

Proof: According to **Lemma 8.1**, the adjacent distances between the vertices in the same block are a multiple of L. If all of the adjacent distances between any neighboring vertices in the meshes are not a multiple of L, then the vertices in the same block cannot be neighboring in the meshes. □

To avoid transmitting neighbors in the same block, we define a confidence level Γ to guarantee that at least this percentage of adjacent elements will not appear in the same block. We choose an appropriate L value based on the mesh geometry to achieve the desired Γ by using **Algorithm 8.1**.

Algorithm 8.1: Compute parameter L

1. Given a mesh, obtain its histogram H as shown in Table 8.1;

2. Set t = 0 and while (H.CumPercent[t] < Γ) t++;

3. Set *MaxDist* = the longest adjacent distance in H;

4. Given B, we search for the minimum L in the range from 2 to $\left\lceil \frac{MaxDist}{B} \right\rceil$ that can achieve the confidence level Γ:

5.L = 2;

6.while ($B * L <= MaxDist$) do {

7. found = true;

8. for (j = 0 to t) do {

9. if (L is divisible by H.distance[j] {

10. found = false;

11. break; // increase L

}

} // for j;

12. if (found) // found satisfying L

13. break;

14. L++;

} // while;

15. return L; □

According to **Lemma 8.1**, a constraint is imposed by our interleaving technique:

- The adjacent distance between elements within a block is a multiple of L.

This constraint ensures that neighboring vertices are not in the same block. Based on our mathematical model, we could distribute the vertices into network packets and retrieve vertices from network packets in a proper order (**Algorithm 8.2** and **Algorithm 8.3**).

Algorithm 8.2: Compute (*i, j*) coordinates for a vertex

- Given a traversal order ℓ of a vertex, the position j of the vertex in a packet is given by: $j = \lfloor \ell / (B \times L) \rfloor$ according to Formula (8.1);
- The packet number i of a vertex is given by the following:

```
// get the packet distribution with j equal to 1 in network packets by
      using Formula (8.1).
1. for (k = 0 to B * L -1) do {
2.      Packet[k] = (k mod B) * L + ⌊k / B⌋
  } // for k

3. nMod = v mod  (B * L);
4. for (i = 0 to B * L – 1) {
5.      if (nMod == Packet[i]) {
6.            break;
        } // if
  } // for i
7. return i, j; □
```

Algorithm 8.3: Compute vertex traversal order *v* given (*i, j*)

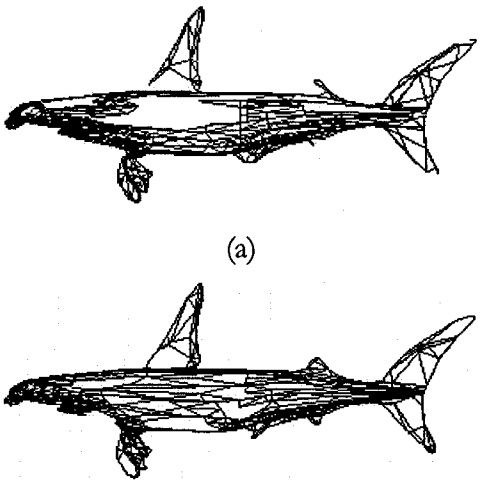
```
// according to Formula (8.1)
1.return (j * B * L + (i mod B) * L + ⌊i / B⌋); □
```

In **Algorithm 8.1**, we find the minimal L which is not divisible by the Γ percentage of the adjacent distances between the neighboring vertices in order to transmit the data in as few packets as possible for networking efficiency. If there are too many vertices for transmission in $L \times B$ packets, the number of packets can be increased to a multiple of $L \times B$. For example, if the number of vertices is V, and the maximum number of vertices that could be transmitted in a network packet is v, the number of packets used is $\left\lceil \frac{V}{L \times B \times v} \right\rceil \times L \times B$. In this case, to map the vertices into network packets, we just replace B by $\left\lceil \frac{V}{L \times B \times v} \right\rceil \times B$.

8.3 Experimental Results and Analysis

In order to verify our probabilistic mesh-dependent interleaving technique, we conducted experiments by using different arbitrary 3D meshes. A confidence level of 80% was used throughout our experiments. Like the authors of [132][133], we simulated the burst packet loss channel by a two-state Markov model known as the Gilbert model [7]. The overall loss probability of the channel was set to 25%, 50%, and 75%. The average burst length $\overline{B}$ of packet loss was set to 2, 3, 5, 8, and 16 packets.

Figure 8.5 shows the reconstructed HammerHead fish mesh after transmitting without (a) and with (b) interleaving using a burst packet loss channel with overall loss probability equal to 25% and burst length equal to 5. Note that when no interleaving was applied, a larger area of the geometry was lost. When our mesh-dependent interleaving was applied, the lost vertices were distributed more evenly.



(a)

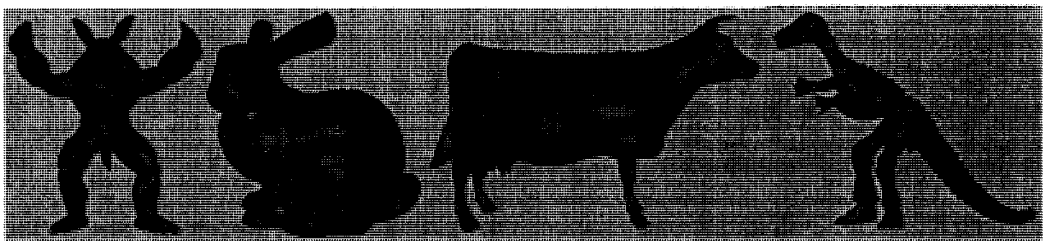
(b)

Figure 8.5: Comparison of results: (a) no interleaving; (b) mesh-dependent interleaving.

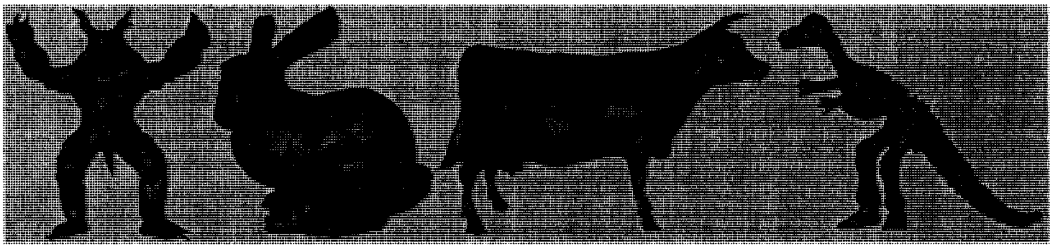
More examples are shown in . The meshes were transmitted with 50% and 75% packet loss, respectively, and with a burst length equal to 16. The results show that the smoothness on the object surface was preserved even at 50% packet loss. At 75% packet loss, the smoothness on the object surface deteriorated, but the overall shape of the objects was preserved. The visual quality of the reconstructed mesh is affected by the original density of the mesh. The Dinosaur mesh had a high density of 5000 vertices, and the visual quality of the reconstructed mesh at 75% packet loss was satisfactory. The other meshes with lower density - Armadillo, bunny and cow - showed obvious degradation in visual quality after reconstruction at 75% packet loss.

We use the metro tool [CRS98] to measure the errors between the original models and the reconstructed models, based on the Hausdorff distance. The metro tool applies surface sampling and point-to-surface distance computation. It samples vertics, edges and faces by taking a number of samples that is approximately 10 times the face number. The reconstruction errors of 6 models after transmitting without interleaving, with the parameter-fixed interleaving (L=64), and with our model-dependent interleaving using a burst packet loss channel, are shown in . The lost geometry was interpolated as described in Section 7.3. In the all 90 cases, no interleaving had the smallest reconstructed error in 6 cases, the parameter-fixed interleaving method had

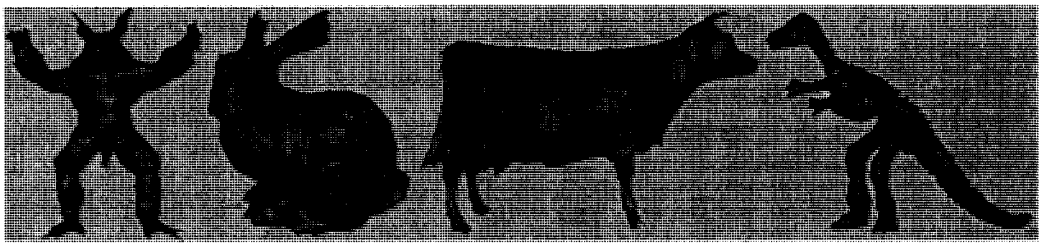
the smallest reconstructed error in 27 cases, and our mesh-dependent interleaving technique had the smallest reconstructed error in the remaining 57 cases. The comparison results verify that, in general, the interleaving methods performed better than a method with no interleaving transmission, and that our probabilistic mesh-dependent interleaving technique performs better than the parameter-fixed interleaving technique.



Original model



Reconstructed model at 50% packet loss (average burst length = 16)



Reconstructed model at 75% packet loss (average burst length = 16)

Figure 8.6: Different models: Armadillo (1st column, 1,752vertices); bunny (2nd column, 2,503vertices); cow (3rd column, 2,904vertices) and dinosaur (4th column, 5,000 vertices).

Table 8.2: Reconstruction error after transmitting through burst packet loss channels by using a confidence level of 80%.

		Reconstruction Error (Hausdorff distance of metro tool[CRS98])								
		Overall Loss Probability =0.25			**Overall Loss Probability =0.50**			**Overall Loss Probability =0.75**		
Model (Vertex Number)	**Average burst length $\overline{B}$, and B**	**No Inter-leaving**	**Fixed Inter-leaving**	**Our Inter leaving**	**No Inter-leaving**	**Fixed Inter-leaving**	**Our Inter-leaving**	**No Inter-leaving**	**Fixed Inter-leaving**	**Our Inter-leaving**
Armadillo (1,752)	$\overline{B}$ =2,B=4	6.41454	5.072178	3.653101	6.41454	5.07217*	6.51283	9.99011	9.81494	8.79573
	$\overline{B}$ =3,B=6	7.91999	5.754521	3.694067	8.22781	7.50195	5.74949	23.77672	11.8564	11.1682
	$\overline{B}$ =5,B=10	8.64229	3.817449	3.374231	9.78420	6.51283*	10.22729	12.58359	18.9428	7.83661
	$\overline{B}$ =8,B=16	7.31479	7.430123	3.969433	10.1119	8.70413	7.04173	9.04029	11.1294	9.59031
	$\overline{B}$ =16,B=32	9.77221	2.925999	2.613735	9.07491	4.70990*	7.43012	14.35158	12.9794	10.6530
Bunny (2,503)	$\overline{B}$ =2,B=4	0.00380	0.00329*	0.004887	0.00529	0.00324*	0.00417	0.00752	0.00642*	0.00910
	$\overline{B}$ =3,B=6	0.00533	0.00224*	0.00267	0.00533	0.00421	0.00305	0.01015	0.00691*	0.00913
	$\overline{B}$ =5,B=10	0.00595	0.00272	0.00232	0.00421	0.00402	0.00344	0.008195*	0.00851	0.00864
	$\overline{B}$ =8,B=16	0.00655	0.00272	0.00201	0.01180	0.00604	0.00390	0.006706*	0.01042	0.00783
	$\overline{B}$ =16,B=32	0.00718	0.00271	0.00243	0.00911	0.00445	0.00327	0.01253	0.00578	0.01318
Cow (2,904)	$\overline{B}$ =2,B=4	0.02855	0.02842	0.01523	0.02571*	0.02836	0.02836	0.028803*	0.03352	0.03717
	$\overline{B}$ =3,B=6	0.02836	0.01407	0.01149	0.05210	0.02857*	0.02836	0.037924*	0.03958	0.06166
	$\overline{B}$ =5,B=10	0.02854	0.02835	0.01217	0.03572	0.02234	0.02163	0.03649	0.03660	0.03638
	$\overline{B}$ =8,B=16	0.03054	0.01671*	0.01750	0.03997	0.02842	0.02340	0.06997	0.03572	0.03870
	$\overline{B}$ =16,B=32	0.04930	0.01870	0.01085	0.07465	0.02935	0.02842	0.08079	0.04888	0.03667
Dinosaur (5,000)	$\overline{B}$ =2,B=4	1.72555	1.00325*	1.05735	2.23990	1.74481	1.51935	2.27519	2.28647*	2.58011
	$\overline{B}$ =3,B=6	2.48860	0.79743*	1.03273	2.48860	1.48978*	1.60988	14.73258	3.13662	2.60489
	$\overline{B}$ =5,B=10	1.87135	0.94130	0.89237	2.68841	1.64225*	1.67793	3.77594	2.70886	2.21918
	$\overline{B}$ =8,B=16	2.49805	1.14487	0.86998	2.95668	2.04629	1.46601	4.05626	3.99035	2.77679
	$\overline{B}$ =16,B=32	4.05297	0.84624	0.74600	2.24394	1.91984	1.61729	3.82511	3.33772*	4.61276
Hammer Head (752)	$\overline{B}$ =2,B=4	0.04221	0.03047	0.02748	0.06154	0.04671	0.04559	0.058907*	0.06814	0.08028
	$\overline{B}$ =3,B=6	0.05999	0.01954*	0.02540	0.06338	0.06304	0.03360	1.11424	0.04851*	0.06973
	$\overline{B}$ =5,B=10	0.03280	0.02902	0.02344	0.03225	0.06211	0.03628	0.08034	0.07598	0.05242
	$\overline{B}$ =8,B=16	0.08034	0.02502*	0.02741	0.11718	0.03770*	0.04770	0.05016	0.06559	0.03746
	$\overline{B}$ =16,B=32	0.08034	0.01930	0.01570	0.02788	0.06810	0.02902	0.03927	0.07023	0.06848
Venus (8,268)	$\overline{B}$ =2,B=4	0.01442	0.00877*	0.00957	0.01862	0.01166*	0.01273	0.02019	0.01928	0.01652
	$\overline{B}$ =3,B=6	0.01970	0.01091	0.00792	0.01998	0.01033*	0.01395	0.02320	0.01970*	0.02302
	$\overline{B}$ =5,B=10	0.02090	0.00797	0.00845	0.01465	0.01046*	0.01418	0.02117	0.01944*	0.01992
	$\overline{B}$ =8,B=16	0.01709	0.00836	0.00673	0.02276	0.00970*	0.00985	0.02080	0.02770	0.02607
	$\overline{B}$ =16,B=32	0.01799	0.01101	0.00841	1.04193	0.02176	0.01678	0.04012	0.02204	0.02140

For the following reasons, our probabilistic mesh-dependent technique cannot always perform better than techniques using either no interleaving transmission or parameter-fixed interleaving transmission. Since we adopted a probabilistic approach (80% confidence in our experiments), a small percentage of adjacent vertices could be distributed in the same block. Moreover, the lost geometry became more obvious when it represented prominent features on the mesh surface. As an example, note that in Figure 8.7 (a), our mesh-dependent interleaving could not restore the lost feature geometry (the smaller fin near the tail). When the lost geometry appeared on a flatter

region (Figure 8.7 (b)), the reconstructed surface was satisfactory even without interleaving. In our future work, we will study the surface curvature during interleaving.

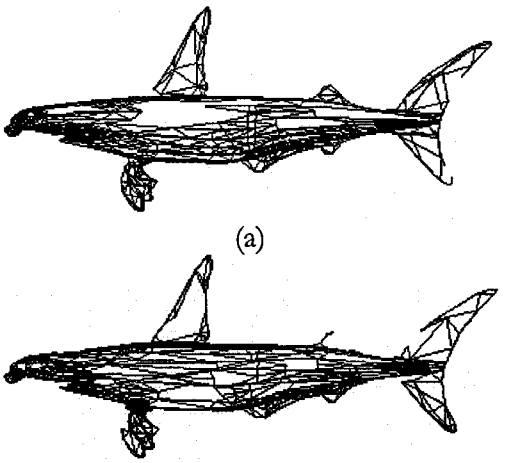


Figure 8.7: Analysis of why no interleaving can have lower reconstruction error than our mesh-dependent interleaving: (a) no interleaving; (b) our mesh-dependent interleaving.

8.4 Conclusions

Interleaving is an efficient way to minimize the adverse effect of burst errors. The existing interleaving algorithms designed for 1D sequences and 2D images are not suitable for arbitrary 3D meshes because of the irregular traversal order dictated by the mesh geometry. In this chapter, we proposed probabilistic mesh-dependent interleaving to minimize the risk of burst packet loss affecting a large neighborhood. The proposed method adjusts the interleaving parameters based on the burst length to be considered and the statistical property of the 3D geometry. A confidence level is defined to ensure that at least a certain percentage of adjacent vertices are not transmitted in the same block of packets. We verified the proposed algorithm over simulated Gilbert channels at different packet loss levels and burst lengths. Our experimental results show that our method has smaller reconstruction error in general compared to the error resulting from the use of no interleaving and the parameter-fixed interleaving techniques. However, when the lost geometry appears on dense feature regions of the mesh, the reconstruction error becomes obvious. Thus, in our future work, we need to consider curvature variations on the 3D surface.

Chapter 9

CONCLUSIONS

9.1 Summary of the Thesis

In this thesis, we addressed the issue of the multimedia delivery over unreliable multi-source networks. We improved optimal bandwidth monitoring for multi-server retrieval, maximizing the expected size of a multimedia object by balancing between the time needed for more accurate bandwidth testing and the time needed for transmitting real multimedia object data. By adopting collaborative retrieval, our improved collaborative retrieval strategy utilized all the channels better than the previous method [58] does, and by using our extended statistical model to estimate the sum of the bandwidth of the servers directly, the estimated object size was larger than can be obtained by using the previous model [58]. We extended the idea of dropping some unreliable channels to achieve the better-estimated size by introducing a reliability factor that can vary between 0 and 1, compared to only 0 or 1.

For the application of P2P Internet VOD, we proposed a new scheme that places a storage cache on peers and replaces the caches based on the ratio of the number of peers demanding to the number of peers with caches instead of traditionally depending on a user to manually decide how much cache space is shared and which video is to be kept for sharing. In simulations, pcVOD showed a significant improvement compared to P2P VOD, in terms of sharing storage managed by users. An optimum performance could be achieved for a reasonably small cache size. In order to reduce the protocol load, pre-fetching was proposed to fetch as much as possible at one time while guaranteeing the data for current playback were ready. We implemented a real-world prototype as well.

Unlike most studies in the literature, which focus on selecting peers with low latency, high uploading bandwidth, and high serving stability by making time-consuming end-to-end measurements, our study proposed traceroute-based fast peer selection

without an offline database and time-consuming probing to narrow the scope of the candidate peers to the close peers either from the same location or within low Round-Trip Time (RTT) from each other. This method of peer selection based on the observation that close peers share routers along the traceroute path to the same destination. Our method is useful when peers from different ISPs and different locations need to be distinguished in order to quickly pre-select peers in a networked environment where the quality of the links varies greatly and to reduce the traffic between different ISPs and different locations. Through the real traceroute records collected by our real-world pcVOD prototype, we obtained the proper parameters and verified the efficiency of our method.

Unlike previous researchers, we carried out a comprehensive study of the impact of packet size, sending rate, sending interval, and buffer size on packet loss by carrying out real-world experiments in a competing WLAN environment, which had packet losses and congestion. Unlike previous researchers who used metrics such as throughput, goodput, transmission range, and energy efficiency, we proposed an approach for estimating the optimal packet size to maximize the expected payload.

The existing research uses either retransmission or ECC to dealing with packet loss. We proposed to transmit the geometry data of 3D meshes over lossy channels without retransmission and ECC. We proposed the strategy of distributing neighboring vertex information into different packets to minimize the risk of packet loss affecting a large neighborhood, and we described an interpolation scheme with different neighbor levels in the order of decoding. With this strategy, the lost vertices are distributed evenly over the model and can be better constructed by interpolation. Experiments on models with different densities show that smoothness on the mesh's surface deteriorates only at a packet loss larger than 75%, and that the object is still recognizable. We also compared the triangle-based linear spline, triangle-based cubic spline and 'v4' interpolation methods. The triangle-based cubic spline interpolation method performed best overall. Compared to the subdivision-based approach, our interpolation scheme performed significantly better.

Due to the arbitrary distribution of vertices on a 3D mesh, a predefined distance or pattern of interleaving is not effective because it may work on one geometry but fail on

another. To combat burst packet loss when transmitting 3D meshes, we proposed probabilistic mesh-dependant interleaving to minimize the risk of burst packet loss affecting a large neighborhood. The proposed scheme adjusts the interleaving parameters based on different mesh geometries (different adjacent distance histograms) and different maximum burst lengths. Our experimental comparison results verified that, in general, our technique performs better than the no-interleaving technique and the parameter-fixed interleaving technique.

9.2 Publications, Commercialization and Future Work

The publications based on this thesis are as follows:

1) L. Ying, A. Basu, and S. Tripathi, "Multi-Server Optimal Bandwidth Monitoring for Collaborative Distributed Retrieval," *IEEE 2004 International Symposium on Circuits and Systems*, Vancouver, 2004.

2) L. Ying and A. Basu, "pcVOD: Internet Peer-to-Peer Video-On-Demand with Storage Caching on Peers," *The Eleventh International Conference on Distributed Multimedia Systems DMS'05*, Banff, Canada, 2005.

3) L. Ying and A. Basu, "Traceroute-Based Fast Peer Selection without Offline Database," *IEEE International Symposium on Multimedia (ISM2006)*, December, 2006.

4) I. Cheng, L. Ying and A. Basu, "Packet Loss Modeling for Perceptually Optimized 3D Transmission," *IEEE Int'l Conference on Multimedia*, July, 2006.

5) I. Cheng, L. Ying and A. Basu, "A Perceptually Driven Model for Transmission of Arbitrary 3D Models over Unreliable Networks," *3rd Intl Symposium on 3DPVT*, June 2006, North Carolina, USA.

6) I. Cheng, L. Ying and A. Basu, "Packet Loss Modeling for Perceptually Optimized 3D Transmission," *Journal of Advances in Multimedia*, Special Issue on Multimedia Networking, 2007.

7) L. Ying, I. Cheng and A. Basu, "Probabilistic Mesh-Dependent Interleaving for 3D Transmission over Burst Packet Loss Channels," *IEEE Transactions on Circuits and Systems for Video Technology (CSVT)*, submitted, Feb. 2008.

Using prototypes related to the P2P multimedia delivery work, we raised US $1 million investment from IDGVC, a US venture capital firm, and co-founded a start-up company, MPO Inc., in Beijing, China.

To improve our work on bandwidth monitoring of collaborative multi-server retrieval, we will try to develop a more efficient way to calculate the reliability factor β by considering pre-computing a set of lookup tables for various conditions and finding an approximate solution for β's based on a quick table lookup. We will verify the bandwidth-monitoring approach by performing real-world experiments.

In our future work on pcVOD, we will conduct experiments using our real-world system to gather, compare and analyze the performance results. Our system will be compatible with existing systems, such as BitTorrent and eDonkey, and will be able to attract users. With our real-world system, we will track the behavior of peers in a typical Internet VOD application, such as the percentages of online peers using this application at different times during the same day. Then further modeling and quantitative analysis of the behavior of peers and pcVOD, such as calculating the optimal cache size, will be conducted.

To extend traceroute-based fast peer pre-selection, future work could include applying our strategy to a totally decentralized architecture, verifying our approach with more traceroute records from more locations, measuring the traffic reduction among different locations, and comparing the initial buffering time of P2P VOD with or without our peer-selection scheme.

To improve the probabilistic mesh-dependent interleaving for 3D mesh transmission, we will take surface curvature into account. Thus, more information at the flat parts can be lost. We will further analyze the proposed scheme and improve the design of the interleaving parameters by avoiding the neighboring vertices between the different blocks.

With the increasing capabilities of today's 3D model acquisition devices, a 3D model with millions of points can be acquired relatively easily [135] [136]. Due to its simplicity and flexibility without connectivity information and its ability to deal with complex topology, the point-cloud-based 3D model has been receiving increasing attention in recent years. We will extend our work to transmit 3D point clouds besides using the 3D mesh model.

Bibliography

[1] D.Tse and P.Viswanath, *Fundamentals of Wireless Communication*, Cambridge University Press, 2005.

[2] G.Wittenburg and J.Schiller, "A Quantitative Evaluation of the Simulation Accuracy of Wireless Sensor Networks," *In Proceedings of 6. Fachgesprach "Drahtlose Sensornetze" der GI/ITG-Fachgruppe "Kommunikation und Verteilte Systeme"*, Aachen, Germany, July 2007.

[3] R.J. Punnoose, R.S.Tseng, D.D.Stancil, "Experimental Results for Interference between Bluetooth and IEEE 802.11 b DSSS Systems," *IEEE 54th Vehicular Technology Conference (VTC)*, 2001.

[4] A.Conti, D.Dardari, G.Pasoliniand and O.Andrisano, "Bluetooth and IEEE 802.11b Coexistence: Analytical Performance Evaluation in Fading Channels," *IEEE Journal on Selected Areas in Communications*, pp. 259- 269, Vol.21-2, Feb. 2003.

[5] F.Mazzenga, D.Cassioli, P.Loreti and F.Vatalaro, "Evaluation of Packet Loss Probability in Bluetooth Networks," *IEEE International Conference on Communications (ICC'02)*, New York, NY, USA, 2002.

[6] F.Mazzenga, D.Cassioli, A.Detti, I.Habib, P.Loreti and F.Vatalaro, "Performance Evaluation in Bluetooth Dense Piconet Areas", *IEEE Transactions on Wireless Communications*, Vol. 3, No. 6, November 2004

[7] E.N. Gilbert, "Capacity of a Burst-Noise Channel," *Bell System Technical Journal*, pp. 1253-1265, 1960.

[8] E.O. Elliott, "Estimation of Error Rates For Codes on Burst-Noise Channels," *Bell System Technical Journal*, pp. 1977-1997, 1963.

[9] Syed A.Khayam and Hayder Radha, "Markov-based Modeling of Wireless Local Area Network," *ACM MSWiM'03*, 2003.

[10] Luis Carvalho, Joao Angeja, and Antonio Navarro, "A New Packet Loss Model of the IEEE 802.11g Wireless Network for Multimedia Communications," *IEEE Transactions on Consumer Electronics*, Vol. 51, No. 3, AUGUST 2005

[11] H. Wang and N. Moayeri, "Finite-State Markov Channel - A Useful Model for Radio Communications Channels," *IEEE Trans. on Veh. Technol.*, vol. 44, N. 1, pp. 163-171, Feb. 1995.

[12] A. Kopke, A. Willig, and H. Karl, "Chaotic Maps as Parsimonious Bit Error Models of Wireless Channels," *IEEE INFOCOM'03*, San Francisco, CA, pp. 513-523, Mar. 2003.

[13] P. Frossard and O. Verscheure, "Joint Source/FEC Rate Selection for Quality-Optimal MPEG-2 Video Delivery," *IEEE Trans. on Image Processing*, vol. 10, N. 12, pp. 1815-1825, Dec. 2001.

[14] K. Stuhlmuller, N. Farber, M. Link, and B. Girod, "Analysis of Video Transmission over Lossy Channels," *IEEE JSAC*, vol. 18, N. 6, pp. 1012-1032, Jun. 2000.

[15] Mohamed Hassan, Marwan M. Krunz, and Ibrahim Matta, "Markov-Based Channel Characterization for Tractable Performance Analysis in Wireless Packet Networks," *IEEE Transactions on Wireless Communications*, VOL. 3, NO. 3, MAY 2004.

[16] Dapeng Wu and Rohit Negi, "A Wireless Channel Model for Support of Quality of Service," *Proc. IEEE Global Communication Conference (GLOBECOM'01)*, San Antonio, Texas, USA, Nov. 25-29, 2001.

[17] G. Alregib, Y. Altunbasak and J. Rossignac, "Error-Resilient Transmission of 3D Models," *ACM Trans. on Graphics*, April 2005. (Early version in ICASSP 02.)

[18] P. Alliez and M. Desbrun, "Progressive Encoding for Lossless Transmission of Triangle Meshes," *SIGGRAPH*, 2001, pages 198-205.

[19] P. Alliez and M. Desbrun. "Valence-Driven Connectivity Encoding of 3D Meshes," *EUROGRAPHICS*, 2001, pages 480-489.

[20] A. Bakre and B. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *Int. Conf. Distributed Computing Systems*, 1995, pages.136–143.

[21] I. Cheng and P. Boulanger, "Just Noticeable Difference for 3D Perception," *Eurographics*, Dubin, August 2005.

[22] I. Cheng and P. Boulanger, "Feature Extraction on 3D TexMesh using Scale-space Analysis and Perceptual Evaluation," *IEEE Trans. on CSVT*, Special Issue on Scale-space Feature Extraction, Oct, 2005.

[23] I. Cheng and A. Basu, "Perceptually Optimized 3D Transmission over Wireless Networks," *SIGGRAPH Webgraphics*, LA, USA, 2005.

[24] Z. Chen, B. Bodenheimer and J. Barnes, "Robust Transmission of 3D Geometry over Wireless Networks," In *Web3D*, 2003, pages 161-172.

[25] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," *IEEE J. Select. Areas Comm.*, vol. 13, pp. 850–857, June 1995.

[26] M. Deering, "Geometric Compression," *SIGGRAPH'95*, 1995.

[27] P. Heckbert and M. Garland, *Survey of Polygonal Surface Simplification Algorithms*, TR, Carnegie Mellon University, 1997.

[28] H. Hoppe, "Progressive Meshes," *SIGGRAPH'96*, 1996.

[29] P. Jaromersky, X. Wu, and Y. Chiang, "Multiple-Description Geometry Compression for Networked Interactive 3D Graphics," In *International Conference on Image and Graphics*, 2005.

[30] K. Lee and S. Chanson, "Packet Loss Probability for Real-time Wireless Communications," *IEEE Trans. on Vehicular Technology*, Nov. 2002.

[31] T. Lewiner, H. Lopes, J. Rossignac and A. Vieira, "Efficient Edgebreaker for Surfaces of Arbitrary Topology," *SIGGRAPH'04*, 2004.

[32] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner, *Level of Detail for 3D Graphics*, Morgan Kaufmann, 2002.

[33] J. O. Limb, "Distortion Criteria of the Human Viewer," *IEEE Transactions on SMC*, 778-793, 1979.

[34] J. L. Mannos and D. J. Sakrison, "The Effects of a Visual Fidelity Criterion on Encoding of Images," *IEEE Trans. on Information Theory*, 1974.

[35] ISO/IEC 14496-2:2000, Amendment 1. Coding of Audio-Visual Objects-Part 2: Visual, Version 2, 2000.

[36] Y. Pan, I. Cheng and A. Basu, "Quality Metric for Approximating Subjective Evaluation of 3D Objects," *IEEE Trans. on Multimedia*, April, 2005. (Short version in IEEE ICIP, Barcelona, 2003.)

[37] F. Pereira and T. Ebrahimi, *The MPEG-4 Book*, Prentice-Hall, Upper Saddle River, New Jersey, 2002.

[38] J. Peng, C.-S. Kim, and C.-C. Kuo, "Technologies for 3D Mesh Compression: A Survey," *Journal of Visual Communication and Image Representation*, 16 (2005) 688-733, 2005.

[39] G. Regib and Y. Altunbasak, "An Unequal Error Protection Method for Packet Loss Resilient 3D Mesh Transmission," *INFOCOM*, 2002.

[40] J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes," *IEEE Trans. on Visualization and Computer Graphics*, pp.47-61, 1999.

[41] J. Rossignac, "Surface Simplification and 3D Geometry Compression," *Handbook of Discrete and Computational Geometry (Second Edition)*, CRC Press, 2003.

[42] D. Tian and G. AlRegib, "FQM: A Fast Quality Measure For Efficient Transmission Of Textured 3D Models," *ACM Multimedia*'04, 2004.

[43] G. Taubin, "3D Geometry Compression and Progressive Transmission," In *Proc. EUROGRAPHICS*, 1999.

[44] C. Touma and C. Gotsman, "Triangle Mesh Compression," In *Graphics Interface*, 1998, pages 26-34.

[44] G. Taubin, A. Guļäeziec, W. Horn and F. Lazarus, "Progressive Forest Split Compression," In *Proc. SIGGRAPH*, 1998.

[45] G. Taubin and J. Rossignac, "Geometric Compression Through Topological Surgery," In *Proc. SIGGRAPH*, 1998.

[46] The Virtual Reality Modeling Language (VRML). ISO/IEC 14772-1, 1997.

[47] D. Wu and R. Negi, "Effective Capacity: A Wireless Channel Model for Support of QoS," *IEEE Trans. on Wireless Communications*, 2002.

[48] X. Xiang, M. Held and J. Mitchell, "Fast & Effective Stripification of Polygonal Surface Models", In *Proc. ACM Symposium on Interactive 3D Graphics* (I3DG'99), Atlanta, USA.

[49] S. Yang, C.H. Lee and C.C.J. Kuo. "Optimized Mesh and Texture Multiplexing for Progressive Textured Model Transmission," *ACM Multimedia*'04, 2004.

[50] Z. Yan, S. Kumar, and C.-C. Kuo, "Error-Resilient Coding of 3-D Graphic Models via Adaptive Mesh Segmentation," *IEEE Transactions On CSVT*, July 2001

[51] D.T. Sandwell, "Biharmonic Spline Interpolation of GEOS-3 and SEASAT Altimeter Data," *Geophysical Research Letters*, 14, 2, 139-142,1987.

[52] P. Cignoni, C. Rocchini and R. Scopigno, "Metro: Measuring Error on Simplified Surfaces," *Computer Graphics Forum*, Blackwell Publishers, vol. 17(2), pp.167-174, Jun. 1998.

[53] I. Cheng, L. Ying and A. Basu, "A Perceptually Driven Model for Transmission of Arbitrary 3D Models over Unreliable Networks," *3rd Intl Symposium on 3DPVT*, June 2006, North Carolina, USA.

[54] Michael Kazhdan, "Reconstruction of Solid Models from Oriented Point Sets," *Eurographics Symposium on Geometry Processing '05*, 2005.

[55] E. Catmull and J. Clark, "Recursively Generated B-spline Surfaces on Arbitrary Topological Surfaces," *Computer-Aided Design*, 10(6):350-355, November 1978.

[56] L Kobbelt, "Sqrt(3)-Subdivision," *SIGGRAPH'00*, 2000.

[57]MathWorks,http://www.mathworks.com/access/helpdesk/help/techdoc/ref/griddata.html

[58] A. Basu, I. Cheng, and Y. Yu, "Multi-server Optimal Bandwidth Monitoring for QoS based Multimedia Delivery," *IEEE ISCAS'03*, Bangkok, Thailand, 2003.

[59] J. Bolliger, T. Gross and U. Hengartner, "Bandwidth Modeling for Network-Aware Applications," *IEEE INFOCOM'99*, March 1999.

[60] R. Braden, L. Zhang, S. Berson and S. Herzog, *Resource ReSerVation Protocol - Version 1 Functional Specification (RFC 2205)*, September 1997.

[61] G. Gopalakrishna and G. Parulkar, *Efficient QoS in Multimedia Computer OS*, Dept. of CS, Washington University, TR, 1994.

[62] R.A. Johnson and D.W. Wichern, *Applied Multivariate Statistical Analysis*, Prentice-Hall, 1988.

[63] A. Vogel, B. Kerherve, G. von Bochmann and J. Gecsei, "Distributed Multimedia and QoS: a Survey," *IEEE Multimedia*, Vol.2 No.2, Summer 1995.

[64] Y. Yu, A. Basu and I. Cheng, "Optimal Adaptive Bandwidth Monitoring for QoS Based Retrieval," *IEEE ISCAS 2002*, Scottsdale, Arizona, USA.

[65] L. Zhang, S. Deering, D. Estrin, S. Shenker and D. Zappala, "RSVP: A New Resource Reservation Protocol," *IEEE Network*, 7(5):8-18, Sep. 1993.

[66] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J.D. Kubiatowicz, "Tapestry: A Resilient Global-scale Overlay for Service Deployment," *IEEE Journal on SAC*, Vol. 22, No. 1, Pgs. 41-53, 2004.

[67] B. Wang, S. Sen, M. Adler and D. Towsley, "Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution," *IEEE INFOCOM'02*, New York, USA, 2002.

[68] B. Cohen, *Incentives Build Robustness in BitTorrent*, http://bitconjurer.org/BitTorrent/, 2003.

[69] D.S. Bernstein, Z. Feng, B.N. Levine, and S. Zilberstein, "Adaptive Peer Selection," *2nd International Workshop on Peer-to-Peer Systems*, Berkeley, CA, USA, 2003.

[70] D. Liben-Nowell, H. Balakrishnan and D. Karger, "Observations on the Dynamic Evolution of Peer-to-Peer Networks," *Int. Workshop on P2P Systems*, 2002.

[71] D. Xu, M. Hefeeda, S. Hambrusch and B. Bhargava., "On Peer-to-Peer Media Streaming," *The 22nd International Conference on Distributed Computing Systems*, Vienna, Austria, 2002.

[72] D.A. Tran, K.A. Hua, and S. Sheu, "A New Caching Architecture for Efficient Video Services on the Internet," *IEEE The 2003 International Symposium on Applications and the Internet*, Orlando, Florida, 2003.

[73] H. Wang, G. Shen, S. Li, and Y. Zhong, "Enhancing Multimedia Streaming Performance Through Peer-Paired Collaboration," *IEEE International Conference on Image Processing*, Rochester, NY, 2002.

[74] L. Ying, A. Basu, and S. Tripathi, "Multi-Server Optimal Bandwidth Monitoring for Collaborative Distributed Retrieval," *IEEE 2004 International Symposium on Circuits and Systems*, Vancouver, 2004.

[75] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-Bandwidth Content Distribution in Cooperative Environments," *Int. Workshop on P2P Systems*, 2003.

[76] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: Peer-to-Peer Media Streaming Using CollectCast," *ACM Multimedia*, Berkeley, CA, 2003.

[77] T.S.E. Ng, Y.H. Chu, S.G. Rao, K. Sripanidkulchai and H. Zhang, "Measurement-Based Optimization Techniques for Bandwidth-Demanding P2P," *IEEE INFOCOM'03*, San Francisco, 2003.

[78] T. Nguyen and A. Zakhor, "Distributed Video Streaming Over Internet," *SPIE Multimedia Computing and Networking*, San Jose, CA, 2002.

[79] X. Jiang, Y. Dong, D. Xu and B. Bhargava, "Gnustream: A P2P Media Streaming System Prototype," *IEEE ICME*'03, Baltimore, Maryland, 2003.

[80] Y. Chu, S.G. Rao, S. Seshan and H. Zhang, "A Case for End System Multicast," *IEEE Journal on SAC*, Special Issue on Networking Support for Multicast, 2002.

[81] Y. Guo, K. Suh, J. Kurose, and D.Towsley, "A Peer-to-Peer On-Demand Streaming Service and Its Performance Evaluation," *IEEE ICME'03*, 2003.

[82] Y. Guo, K. Suh, J. Kurose and D. Towsley, "P2Cast:P2P Patching Scheme for VoD Service," *12th International World Wide Web Conference*, Budapest, Hungary, 2003.

[83] Z. Xiang, Y. Zhong and S.Q. Yang, "PeriodPatch: An Efficient Stream Schedule for VoD," *SPIE Conf. on Internet Multimedia Mgmt. System*, 2000.

[84] Sun Microsystems, Inc., JXTA Project, http://www.jxta.org/

[85] Y.C. Tua, J. Sunb and S. Prabhakar, "Performance Analysis of a Hybrid Media Streaming System," *SPIE Conf. on Multimedia Computing and Networking*, California, U.S. Jan., 2004.

[86] D.Qiu and R.Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," *SIGCOMM'04*, 2004.

[87] A. Al Hamra, E.W. Biersack and G. Urvoy-Keller, "A Pull-Based Approach for a VoD Service in P2P Networks," *7th IEEE International Conference on High Speed Networks and Multimedia Communications*, Toulouse, France, 2004.

[88] C. Loeser, P. Altenbernd, M. Ditze and W. Mueller, "Distributed Video on Demand Services on Peer to Peer Basis," *Int. Workshop on Real-Time LANs in The Internet Age*, 2002.

[89] W.J. Jeon and K. Nahrstedt, "Peer-to-Peer Multimedia Streaming and Caching Service," *IEEE ICME'02*, 2002.

[90] C.L. Chan, S.Y. Huang, M. Jan and J.S. Wang, "Peer-to-Peer Video Delivery Scheme for Large Scale Video-On-Demand Applications," *IEEE ICME'04*, 2004.

[91] T.T. Do, K.A. Hua and M.A. Tantaoui, "P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment," *International Conference on Communications (ICC'04)*, 2004.

[92] S. Jin and A. Bestavros, "Cache-and-Relay Streaming Media Delivery for Asynchronous Clients," *International Workshop on Networked Group Communication (NGC'02)*, 2002.

[93] T. S. Eugene Ng, Y.-H. Chu, S. G. Rao, K. Sripanidkulchai and H. Zhang, "Measurement-Based Optimization Techniques for Bandwidth-Demanding Peer-to-Peer Systems," *IEEE INFOCOM 2003*, 2003

[94] M. Adler, R. Kumary, K. Rossz, D. Rubensteinx, T. Suel and D.D. Yaok, "Optimal Peer Selection for P2P Downloading and Streaming," *IEEE INFOCOM'05*, 2005.

[95] L. Xiao, Y. Liu, and L.M. Ni, "Improving Unstructured Peer-to-Peer Systems by Adaptive Connection Establishment," *IEEE Transactions on Computers*, Vol. 54, No. 9, Sep 2005.

[96] Z. Xu, C. Tang, and Z. Zhang, "Building Topology-Aware Overlays using Global Soft-State," *23rd International Conference on Distributed Computing Systems (ICDCS)*, 19-22 May, 2003.

[97] X. Zhang, J. Liu, B. Li, and T.S.P. Yum, "CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming," *IEEE INFOCOM'05*, 2005.

[98] http://www.pplive.com

[99] http://www.ppstream.com

[100] V.N. Padmanabhan and L. Subramanian, "An Investigation of Geographic Mapping Techniques for Internet Hosts," *ACM SIGCOMM'01*, 2001.

[101] http://www.cz88.net/fox/

[102] http://www.ip2location.com/

[103] V. Jacobson, "Traceroute Software," 1989, ftp://ftp.ee.lbl.gov/traceroute.tar.gz

[104] Z.M. Mao, J. Rexford, J. Wang and R.H. Katz, "Towards an Accurate AS-Level Traceroute Tool," *ACM SIGCOMM 2003*, 2003.

[105] L. Ying and A. Basu, "pcVOD: Internet Peer-to-Peer Video-On-Demand with Storage Caching on Peers," *The Eleventh International Conference on Distributed Multimedia Systems DMS'05*, Banff, Canada, 2005.

[106] S. Floyd, M. Handley, J. Padhye and J. Widmer, "Equation-Based Congestion Control for Unicast Applications," *SIGCOMM'00*, 2000.

[107] A.C. Feng, A.C. Kapadia, W.-C. Feng and G.G. Belford, "Packet Spacing - An Enabling Mechanism for Delivering Multimedia Content in Computational Grids," *The Journal of Supercomputing*, 23, 51–66, 2002.

[108] X. Zhang and H. Schulzrinne, "Voice over TCP and UDP," *Technical Report*, Department of Computer Science, Columbia University, 2004

[109] Y. Gu and R. L. Grossman, "Optimizing UDP-based Protocol Implementations," *Third International Workshop on Protocols for Fast Long-Distance Networks (PFLDNet)*, Lyon, France, Feb. 2005.

[110] R. El-Azouzi and E. Altman, "A Queuing Analysis of Packet Dropping over A Wireless Link with Retransmissions," *Proc. PWC*, Italy, 2003.

[111] V. K. M. Vadakital, M. M. Hannuksela, M. Razaei and M. Gabbouj, "Optimal IP Packet Size for Efficient Data Transmission in DVB-H," *IEEE NORSIG*, 2006.

[112] Y. Sankarasubramaniam, I.F. Akyildiz and S.W. McLaughlin, "Energy efficiency based packet size optimization in wireless sensor networks," *IEEE Sensor Network Protocols and Applications*, 2003.

[113] I. F. Akyildiz. and I. Joe, "A New ARQ Protocol for Wireless ATM Networks," *Proc. of IEEE ICC'98,* Vol.2. pp. 1109-1113, 1998.

[114] P. Letteri and M.B.Srivastava, "Adaptive Frame Length Control for Improving Wireless Link Throughput, Range, and Energy Efficiency," *Proc. IEEE INFOCOM,* pp. 564-571, San Francisco, USA. March 1998.

[115] E. Modiano, "An Adaptive Algorithm for Optimizing the Packet Size used in Wireless ARQ Protocols," *ACM-Boltzer Journal* of *Winless Nerworks 5,* pp. 279-286. 1999.

[116] C. K. Siew and D. J. Goodman. "Packet Data Transmission Over Mobile Radio Channels," *IEEE Trans. on Vehicular Technology,* Vol. *38,* No. 2, May 1989.

[117] J. D. Spragins, J. L. Hammond, K. Pawlikowski, *Telecommunications: Protocols and Design*, Addison Wesley Publishing Company, Inc. 1991.

[118] D. Minoli, "Optimal Packet Length for Packet Voice Communication Communications," *IEEE Trans. on Communications*, Vol.27, No. 3, March 1979.

[119] A. Ganz, Z. Ganz and K.Wongthavarawat, *Multimedia Wireless Networks: Technologies, Standards and QoS*, Prentice Hall PTR, 2004.

[120] R. C. Bose and D. K. Ray-Chaudhari, "On a Class of Error Correcting Binary Group Codes," *Information Control*, vol. 3, pp. 68–79, March 1960.

[121] H. S. Wang and P.-C. Chang, "On Verifying the First-Order Markovian Assumption for a Rayleigh Fading Channel Model," *IEEE Trans. Vehicular Tech*, vol. 45, pp. 353–357, May 1996.

[122] L. Wilhelmsson and L. B. Milstein, "On the Effect of Imperfect Interleaving for Gilbert-Elliott Channel," *IEEE Trans. on Communications*, vol. 47, no. 5, pp. 681–688, May 1999.

[123] J. R. Yee and J. E. J. Weldon, "Evaluation of the Performance of Error Correcting Codes on a Gilbert Channel," *IEEE Trans. on Communications*, vol. 43, no. 8, pp. 2316–2323, 1995.

[124] Y. Q. Shi, X. M. Zhang, Z. C. Ni, and N. Ansari, "Interleaving for Combating Burst of Errors," *IEEE Circuits and Systems Magazine*, pp. 29–42, 2004.

[125] S. Bischoff and L. Kobbelt, "Towards Robust Broadcasting of Geometry Data," *Computers and Graphics*, vol. 26, no. 5, pp. 665–675, 2002.

[126] X. M. Zhang, Y. Q. Shi and W. Q. Xu, "Optimal 2-D Interleaving for Robust Multimedia Transmission," *International Conference on Information Technology: Research and Education (ITRE2003)*, Aug. 2003.

[127] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Englewood Cliffs, NJ: Prentice-Hall, 1995.

[128] UPS Inc. "Maxicode Briefing Document," http://www.maxicode.com, 1996.

[129] K.A.Abdel-Ghaffar, "Achieviag the Reiger Bound for Burst Errors Using Two-Dimensional Interleaving Schemes," *Proceeding IEEE Int. Symp. Information Theory*, p.425, 1997.

[130] M. Blaum, J. Bruck and A. Vardy, "Interleaving Schemes for Multidimensional Cluster Errors," *IEEE Trans. Information Theory*, vol. 44. pp. 730-743, 1998.

[131] W.Q. Xu and S.W. Golomb, "Optimal Interleaving Schemes for Correcting Two-Dimensional Cluster Errors," *Discrete Applied Mathematics*, vol. 155, pp.1200–1212, 2007.

[132] S.H. Yang and J.M. Lin, "Burst-Packet-Loss Concealment for MPEG-2 Video," *Taiwan Telecommunication Workshop* 2002, Dec. 2002.

[133] Q.Qu, Y.Pei and J.W.Modestino, "Robust H.264 Video Coding and Transmission over Bursty Packet-Loss Wireless Networks," *IEEE 58th Vehicular Technology Conference (VTC 2003)*, Oct. 2003.

[134] Y.J. Liang, J.G. Apostolopoulos and B.Girod, "Analysis of Packet Loss for Compressed Video: Does Burst-Length Matter?," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, Apr. 2003.

[135] A. Zakhor and C. Frueh, "Automatic 3D Modeling of Cities with Multimodal

Air and Ground Sensors", in *Multimodal Surveillance, Sensors, Algorithms and Systems*, Z. Zhu and T. S. Huang, Editors, Artech House, Chapter 15, pp. 339-362, 2007.

[136] Y. Huang, J. Peng, C.C.J. Kuo, and M. Gopi, "Octree-Based Progressive Geometry Coding of Point Clouds," *Eurographics Symposium on Point-Based Graphics*, 2006.

[137] P. Mehra, C. D. Vleeschouwer, and A. Zakhor, "Receiver-Driven Bandwidth Sharing for TCP and its Application to Video Streaming," *IEEE Transactions on Multimedia*, Vol. 7, No. 4, pp. 740-752, Aug. 2005.

[138] T. Nguyen and A. Zakhor, "Multiple Sender Distributed Video Streaming," *IEEE Transactions on Multimedia*, Vol. 6, No. 2, pp. 315 - 326, Apr. 2004.

[139] I. Cheng, L. Ying and A. Basu, "Packet Loss Modeling for Perceptually Optimized 3D Transmission," *Journal of Advances in Multimedia*, Special Issue Vol. 2007, 2007.

[140] L. Ying and A. Basu, "Traceroute-Based Fast Peer Selection without Offline Database," *IEEE International Symposium on Multimedia (ISM2006)*, December, 2006.

[141] I. Cheng, L. Ying and A. Basu, "Packet Loss Modeling for Perceptually Optimized 3D Transmission," *IEEE Int'l Conference on Multimedia*, July, 2006.

[142] L. Ying, I. Cheng and A. Basu, "Probabilistic Mesh-Dependent Interleaving for 3D Transmission over Burst Packet Loss Channels," *IEEE Transactions on Circuits and Systems for Video Technology (CSVT)*, to be submitted.