Multimedia Systems manuscript No.
(will be inserted by the editor)

Hybrid and Forward Error Correction Transmission Techniques for Unreliable Transport of 3D Geometry

Zhihua Chen[1], **J. Fritz Barnes**[2], **Bobby Bodenheimer**[1]

[1] Electrical Engineering and Computer Science, Vanderbilt University, Box 1679 Station B, Nashville TN 37235
e-mail: {chenz,bobbyb}@vuse.vanderbilt.edu
[2] Symantec, Cupertino, CA
e-mail: jfb@acm.org

Abstract. The continual improvement in computer performance together with the prevalence of high-speed network connections having high throughput and moderate latencies enables the deployment of multimedia applications, such as collaborative virtual environments, over wide area networks. These applications can serve as simulated environments in scenarios such as emergency response training to catastrophic disasters, military training, and entertainment. Many of these systems use 3D graphics for display and may be required to distribute geometric models on demand between participants. Progressive meshes provide an attractive mechanism for such distribution. Previous uses of progressive meshes have sent data using reliable protocols (TCP). However, such protocols have disadvantages in on-demand settings, in that they: (1) use flow control, which limits performance in wide area networks; (2) add additional bandwidth when there is loss; (3) treat all loss as an indication of congestion; and (4) require feature-rich multicast support, which is not always available. In this paper, we modify progressive mesh models to allow reconstruction even in the event of packet loss. We use these modifications in two transmission schemes, a hybrid transmission that uses TCP and UDP to send packets and a forward error-correcting transmission scheme that uses redundancy to decode the information sent. We assess the performance of these transmission schemes when deployed on network testbeds that simulate wide-area and wireless characteristics.

Key words: Lossy transport – 3D geometry – Progressive mesh

1 Introduction

As the Internet expands, demand is growing for interactive multimedia applications, from games such as Everquest [14] to collaborative virtual environments and purely web-based applications. Such applications use high resolution 3D meshes to embed participants in virtual environments. The growing demand for these data intensive meshes introduces new challenges for storage and transmission. There are two traditional methods for obtaining high resolution 3D meshes in these applications: (1) use a client-server model and assume the client already has the model; (2) wait for the model to be downloaded over the Internet.

Both methods have drawbacks. The traditional client-server model that assumes all the models are local is appropriate for such applications as games[1] where models never change, but is less appropriate for distributed applications where new models may be created by users and incorporated into the environment in real-time. Likewise, downloading may be appropriate in some scenarios, but even if the data is compressed the transmission of a complex model can involve unacceptable delays. For example, online video games typically use the first method to obtain game 3D geometry, because smooth navigation and increased user satisfaction result when there are no delays in rendering caused by not having the model.

Geometric data for multimedia applications is usually stored and transmitted as a complex of polygons (typically triangles) with shared vertices. These complexes are known as polygonal (triangular) *meshes*. An example of a geometric object constructed from a triangular mesh is shown in Figure 1, which is a model of a Cessna airplane and is fully described in Section 5. Accompanying the vertex data in the mesh is often additional information such as shading parameters and normal information that help render the geometric data. Although triangular meshes are the *de facto* standard for computer graphics, they possess disadvantages, the most notable of which is the number of triangles required to render a high quality image of an object.

Hoppe [20] proposed progressive meshes (PMes) as an initial solution for the transmission of geometric data over data networks. In this method, the server initially sends coarse shape information to a client that can be reconstructed and

Correspondence to: bobbyb@vuse.vanderbilt.edu

[1] However, many games allow modifications that extend the initial game and user-defined models that allow for customization of the game.

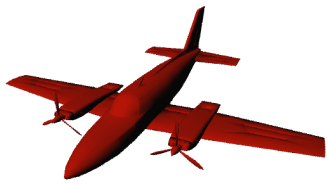


Fig. 1 An example rendering of a triangular mesh, a Cessna airplane.

rendered very quickly. This coarse shape information is called the *base mesh*. Then increasing detail in the model is transmitted to the client, allowing the client to progressively refine the initial model into the full resolution model. If, for example, the object is initially far away, then a user cannot perceive the loss of detail in the model caused by rendering the initial model. More generally, the user is able to see and interact with the coarse model immediately, and thus the delay while the model is being refined is not as perceptually disturbing as delay in the action while the model is fully downloaded. In its basic form, progressive transmission does not *reduce* the amount of data that needs to be transmitted, but it *orders* the data from most important to least important. The PM technique was not intended to be robust to random losses occurring in the transmission channel. As a result, encoding a 3D geometry using the PM scheme and transmitting the file over a lossy network such as the Internet using UDP or Multicast will result in data that cannot be reconstructed into the original form. Previous papers have been more concerned with the efficiency of the compression technique than robustness to packet loss.

Various authors [30, 39, 26] have combined mesh compression techniques with progressive transmission to reduce the amount of data that needs to be sent. Typically, there is a trade-off between compression ratio and the *fragility* of the compressed mesh. For example, if even one packet is lost in a highly compressed mesh then the entire geometry may not be reconstructible from what remains. This fragility mandates the use of a *reliable* transmission method, and the common protocol for such transmission is TCP, which is generally error-free. TCP has four disadvantages in the multimedia setting: it (1) uses flow control, (2) increases bandwidth, (3) treats all loss as congestion, and (4) requires feature-rich multicast.

TCP's flow control was designed to avoid senders that send data faster than the application at the receiver can handle the data [34]. In the case of 3D geometry over wide-area networks, it is unlikely that the sender will send faster than the receiver can receive. However, if the network latency is large, the overall throughput will be decreased because the server does not know if there is any available space for the receiver to receive the next packet.

When packet loss occurs, TCP requires retransmission of lost segments. The retransmission increases the number of packets sent over the link. If the loss was caused by congestion, this retransmission increases the congestion problem in the network.

Many network endpoints are attached to networks using wireless connections. These connections have advanced error correction mechanisms, yet are unable to catch all errors that may occur when transmitting a packet over a wireless network. Endpoints will notice these lost packets and assume that congestion exists in the network. The sender will then halve the rate at which it sends data.

In many collaborative virtual environments [27, 17], a single sender sends data to multiple recipients over the Internet Multicast Backbone (MBONE) [9]. This technique allows a host to scalably transfer information to multiple recipients. However, multicast communication often does not support reliable communication because of the complex and prohibitive cost associated with developing such transport.

One way to improve the performance of the transmission of PMes is to use an unreliable protocol that will not face the same limitations described above. However, alternative transmission protocols such as UDP are unacceptable because they may not deliver all packets to the receiver. In this paper, we consider different mechanisms that can be used to transmit PMes using unreliable channels.

In prior work, we demonstrated that using an unreliable channel can improve transmission times by as much as forty percent with a negligible drop in visual quality [12] and we investigated mechanisms to improve handling of lost packets in the PM representation [11]. In this work, we add an additional transmission scheme that makes use of forward-error correction to transmit portions of the mesh and we perform analysis of both previous and new transmission methods across wireless networks. We perform additional analyses to determine how much of the performance degradation is caused by congestion control in these networks. We find that the use of unreliable data transport mechanisms can improve transmission times across wide-area networks and/or networks with wireless segments and decrease the variability of 3D geometry transmission.

The paper is organized in the following manner. In Section 2, we review prior work in this area and place our work in context. Section 3.1 details the modifications to the basic PM scheme needed when data loss can occur. Section 3.2 discusses our hybrid transmission protocol and the forward error correction codes we use. Section 4 discusses the experimental setup used to conduct our tests, and provides details of the experiments we ran. Section 5 presents the results of actual transmission experiments over a lossy network with an accompanying analysis. Finally, in Section 6 we discuss our findings and future work we intend to investigate.

2 Background

Our work evaluates two methods for transmitting 3-D geometric meshes. We take advantage of the underlying data be-

ing sent to improve both transmission and tolerate the loss of packets with a slight degradation in geometric quality. In this section, we present a brief overview of the bulk transmission and the progressive mesh representation, which we modify to add tolerance for packet losses.

2.1 Transfer of Bulk Data

Many papers have explored how to reliably send data through unreliable networks [34, 19, 16, 38]. Essentially, there are two approaches: automatic repeat-request (ARQ) and forward-error correction. In ARQ, one adds additional information to packets such that the sender and receiver can determine if packets have been lost and the receiver can request a retransmission of any lost data. The drawbacks of using an ARQ protocol are that it increases the time of transmission of data and does not scale for use in multicast transmissions.

Recent work [15, 23, 25] has looked at modifications of TCP, a common ARQ protocol used to reliably transmit data over the Internet, to improve the transfer of bulk data with large transmission windows. In particular, these schemes modify TCP's additive increase, multiplicative decrease (AIMD) algorithm to increase the congestion window more aggressively and decrease it less drastically when the congestion window is large. We take a different approach: we assume that some packets can be lost and therefore do not need to be transmitted. In some ways, our work is orthogonal to the above work in that their techniques could be added to control the send rate of our UDP transmitted packets.

In forward-error correction (FEC) methods, one adds redundant information to packets such that the receiving application can reconstruct lost packets from data contained in other packets that this application has already received. The primary disadvantage of FEC is the addition of redundant data in the transmission.

The FEC codes used in our work are the Vandermonde-based Reed-Solomon codes [37]. Byers et al. developed Tornado Codes [8], a FEC technique that provides a faster encoding and decoding speed. Byers et al. applied Tornado Codes to enable efficient reliable distribution of bulk data. Their work differs from ours in that they employ the same protection of all data, they assume a different model of transmission, and they do not use information about the underlying data being transmitted.

Multimedia transmission research [32, 35] has considered techniques that dynamically adjust the amount of information encoded in the multimedia stream to meet the bandwidth available between the sender and the receiver. In many cases, these schemes cannot afford to drop packets or use forward-error correction mechanisms. This approach differs from the transmission of 3D geometric information that we study in this paper. Alternative research in multimedia delivery systems has explored the use of transmission over partially ordered transport protocols [13] and the increased use of buffers for supporting long-lived multimedia streams [2].

Additional research has investigated the question of whether one can guarantee the quality of network transmissions. Guaranteed quality has an obvious and important impact on mesh transmission. Specific work includes integrated and differentiated services [40, 7, 6] within the Internet. These proposals provide mechanisms that could be used to control the number of packets lost in the network. As a result, the use of these mechanisms could eliminate concerns about lost packets. Unfortunately, deployment and use of integrated or differentiated services is not currently available.

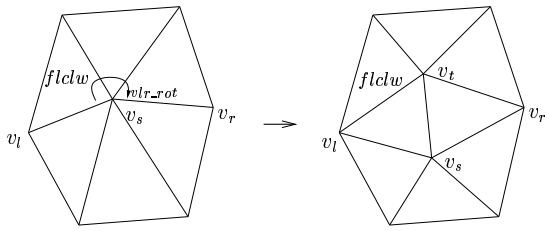


Fig. 2 Vertex split transformation.

2.2 Progressive Mesh Transmission

The PM scheme was devised by Hoppe in [20, 22], and we have implemented the basic versions without the modifications of later work, e.g., [21, 33]. The PM representation of a mesh M is stored as a coarse mesh M^0 and a sequence of n detail records called *vertex splits*. These vertex splits indicate how to incrementally refine M^0 so that after the n vertex splits have been processed, the original mesh M is recovered. In fact, the PM representation defines a sequence of meshes $M^0, M^1, \ldots, M^n$ which provide increasingly accurate approximations of M.

A vertex split is a basic transformation that adds a vertex to the mesh. The basic progressive mesh scheme is implemented, but for the purposes of this paper, a vertex split does not contain normal, texture, or material information. This simplification is conservative in that the loss of a packet containing only vertex splits will be more damaging to visual fidelity than a packet containing vertex splits and additional information. Each vertex split is a 30 byte quantity consisting of a face index, $flclw$, an index vs_index ($0 \leq vs\_index \leq 2$), an encoding vlr_rot, and two vertex position deltas, vad_l and vad_s. In our experiments, these vertex splits are packed into 1400 byte packets. Thus, each packet contains roughly 46 vertex splits.

Figure 2 illustrates a PM vertex split transformation. Each vertex split operation introduces a new vertex v_t and two new faces, as shown. The location of a vertex split is parameterized by v_s, v_l, and v_r. By default, the PM data structure does not include incidence information in the vertex to face direction. The vertex values are determined through the fields $flclw$, vs_index, and vlr_rot. To determine the vertex being split (v_s), the three vertices of the face $flclw$ are sorted by their index values and stored into an ordered list. They are indexed by vs_index. Vertex v_l is the next vertex clockwise on the face $flclw$. The vertex v_r is determined by vlr_rot, the number of clockwise rotations about v_s from v_l to v_r.

Other techniques for mesh decomposition with an idea toward robust transmission exist. A more complicated scheme involving decomposing a mesh into a set of overlapping ellipsoids and point sampling the shape has been proposed by by Bischoff and Kobbelt [5]. Chen and Nishita [10] have constructed a streaming mesh format for progressive transmission with quality of service (QoS) control. This QoS control gives them advantages in anticipating network congestion, but their transmission technique is still built upon TCP and suffers the drawbacks of it. Using forward error correction for *compressed* progressive meshes [30] has been investigated by Al-Regib and Altunbasak [1]. Conceptually, this work is similar to our own. However, Al-Regib and Altunbasak consider the mesh as being decomposed *a priori* into discrete levels of detail. If the bit stream of one of these levels of detail is lost, there is an amount of distortion introduced into the rendered mesh corresponding to these levels of detail. Our method, in contrast, is more opportunistic in that it may lose more data, but renders what it can at a finer granularity, determined by the amount of data lost. An additional difference is that we have tested our method on a wide area network with simulated background traffic, whereas their results come entirely from simulations.

In the context of progressive transmission much of the work has focused on methods for better compression of geometric data rather than on robust transmission [39, 30, 26, 3, 18]. These techniques are complementary to our own, in that they could fit within the techniques proposed here to achieve higher transmission rates.

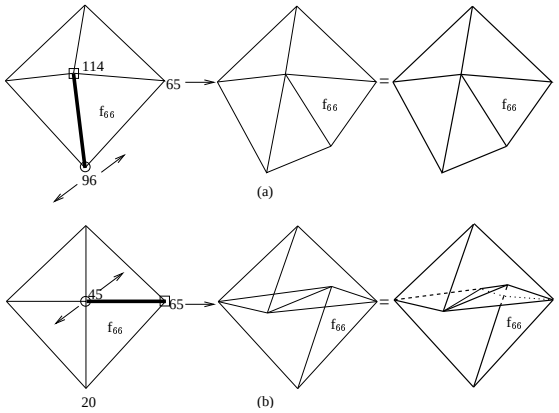


Fig. 4 Illustration of self-intersection occurrence caused by vs_index: (a) when there is no packet loss, intermediate vertex splits change the three vertices of face f_{66} to 65, 96 and 114; (b) when packets containing the intermediate vertex splits are lost, the three vertices of face f_{66} remain unchanged as 65, 20, 45.

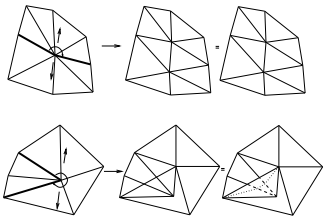


Fig. 5 Illustration of self-intersection occurrence caused by vlr_rot: (a) when there is no packet loss, intermediate vertex splits change the number of faces around v_s to 8; (b) when packets containing the intermediate vertex splits are lost, the number of faces around v_s remain unchanged at 6.

3 Unreliable Channel Distribution

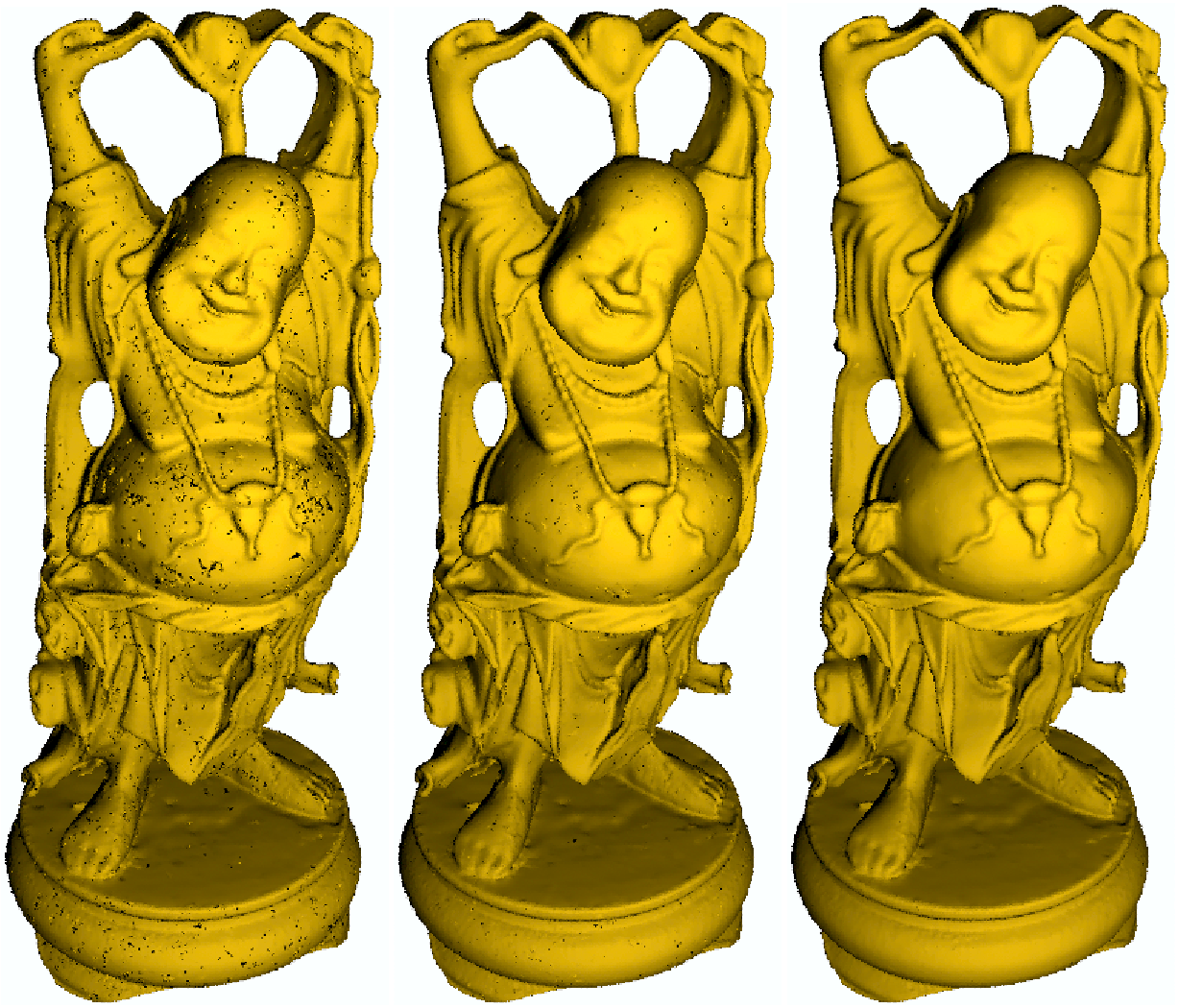
Careful modification of the PM method described above can make the PM representation tolerant of lost packets. Improving the robustness allows the use of unreliable transmission mechanisms. In this section, we discuss changes we made to the PM representation to handle loss. We follow this presentation with a description of a hybrid transmission scheme and a forward error-correction scheme that make use of the fact that packet loss can be tolerated when the receiving application attempts to reconstruct the original 3D geometry.

3.1 PM Extensions to Handle Loss

When the transmission channel is lossy, vertex splits will be lost when packets are lost. As a result, the geometry of the reconstructed model can be corrupted. This corruption most often takes the form of surface self-intersection, i.e., parts of the surface intersecting with the surface itself. The self-intersections destroy the manifold property that the surfaces in the original model have and create visual artifacts in the reconstructed models, as shown in Figure 3 (a). This figure shows a standard 3D model called "the happy Buddha," the details of which are described in Section 5. We now discuss how lost vertex splits cause self-intersections to occur and describe modifications to the PM data structure that improve the situation. These changes to the PM data structure do not increase the size of the information associated with each vertex split.

First, in addition to a packet containing only vertex split information, each packet contains a 4 byte header that stores the index number of the first face of the mesh that will be introduced by the first vertex split in this packet. The client renderer uses this number as a "poor man's" error correction, to give faces generated by vertex splits after lost packets their index number in the full resolution mesh. This process is done so that future splits whose face index, $flclw$, reference these faces can find them. This header is a monotonically increasing number, and can also be used to detect out-of-order packets.

One cause of self-intersection is that the vs_index field in a vertex split record is the $relative$ numerical order of the index value of the split vertex v_s among the three vertices on $flclw$ (see Figure 2). When vertex splits are lost, the index values of the vertices on $flclw$ may become different from what they were when the progressive mesh was generated. Thus, the ordering of the vertices may be wrong. This change of ordering may cause the PM reconstruction to find the wrong v_s, which results in geometric corruption. For example, in Figure 4 (a), without packet loss, intermedi-



(a) (b) (c)

Fig. 3 The Buddha model after transmission over a lossy network: (a) no change to PM data; (b) using absolute vs_index; (c) using both absolute vs_index and altered vlr_rot field. The black spots indicate areas where the triangle mesh was corrupted and thus renders incorrectly.

ate vertex splits change the three vertices of face f_{66} to 65, 96 and 114. A subsequent vertex split with $flclw$=66 and vs_index=1 splits vertex 96 because its index is the second smallest. When the intermediate splits are lost, as shown in Figure 4 (b), the three vertices of face f_{66} remain unchanged as 65, 20, 45. Now vertex 45 has the second smallest index among the vertices of f_{66}. The same split with $flclw$=66 and vs_index=1 then splits vertex 45 and causes the faces connected with the added vertex to pierce the top right face.

The PM data does not store how the three vertices of a face to be reconstructed were ordered in the vertex list of the face in the original model. Therefore the ordering of vertices in the vertex list of a face in the reconstructed model may be different from that in the original model. This fact is part of the reason why the vs_index was defined as a $relative$ order of index value rather than the $absolute$ placement of v_s in the vertex list of $flclw$. To alleviate this problem, a second pass over the PM data is made that replaces each $relative\ vs\_index$ with the $absolute$ placement of v_s at the time of the vertex split. The time the second pass takes is negligible compared with the time to generate the PM. When this change is incorporated a significant number of self-intersections can be eliminated, as shown in Figure 3 (b).

The vlr_rot field in a vertex split record can also cause self-intersections to occur. The vlr_rot field, as originally defined, is the number of clockwise rotations from v_l to v_r about v_s. When vertex splits are lost, the number of rotations from v_l to v_r may be different from what it was in its original context, or v_r may not exist at all. This difference can cause the reconstruction program to pick the wrong v_r, resulting in geometric corruption as shown in Figure 5. To help the reconstruction program detect this case without changing the size

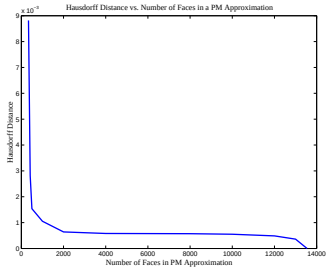


Fig. 6 The Hausdorff distance of the Cessna model for different PM approximations by number of faces in the model.

of the PM data structure, we modify the vlr_rot field. Instead of using 16 bits for the vlr_rot field, we use 8 bits, and use the remaining 8 bits to encode a quantity called $cycle\_size$. The $cycle\_size$ field stores the total number of faces around v_s. The reconstruction program checks the $cycle\_size$ at run time and discards splits whose $cycle\_size$ do not match those in the partially reconstructed meshes. When this change is incorporated, more self-intersections can be eliminated, as shown in Figure 3 (c).

3.2 Transmission Schemes

The progressive mesh format creates an alternative representation of the 3D geometry. One significant advantage of this representation is that some packets containing mesh data are more important than other packets. An example of this difference in packet priority appears in Figure 6. This figure compares the Hausdorff distance [24] between the original 3D model and a model derived by performing a subset of the vertex splits on the base model. We calculate the surface deviation using the algorithm described in Aspert et al. [4]. Notice in the graph that the initial splits provide significant improvements in the Hausdorff metric, while later splits provide decreasing benefits.

3.2.1 Hybrid Transmission Our hybrid technique leverages the inherent differences in the effect that vertex splits have on the resulting visual accuracy of the reconstructed mesh. Thus, in this scheme, we begin by transmitting data using the TCP protocol. Part-way through the transmission, the hybrid sender closes the TCP connection and transmits the remaining data using UDP. The sender can thus reliably transfer the base mesh and some of the initial splits and then use a more aggressive technique to transfer less important splits.

TCP controls the rate that packets are sent to maintain flow control and minimize congestion. As a result, TCP does not provide the end-user control of the sending rate. When we consider using UDP for transmission of data, the end-user has significant influence over the send rate. Therefore, our application must carefully select the send rate. If the send rate exceeds the capacity of the channel, we will increase the number of packets lost. However, if we decrease the send rate, we will not lose any packets but the transfer of information will take much longer than a TCP connection might take. We experimentally determine UDP send rates to use in applications.

An issue of concern in implementing this idea is that the sender can begin sending UDP data to the receiver while the receiver is still reading data from the TCP connection. This behavior causes a problem since the initial data will fill up the operating system's buffer for incoming network data and start dropping UDP packets that arrive. Therefore, in the hybrid protocol the sender waits for the receiver to close the connection before it starts to send the UDP data. This delay guarantees that the underlying TCP stack will have successfully transmitted all data.

3.2.2 FEC Transmission Initial experimental experience with the hybrid transmission scheme indicated that even transmitting the base model of the mesh using TCP creates significant delays and variability in model transmission during network congestion. An alternative is to send the entire geometric model using only UDP. However, this alternative alone would be disastrous when transmitting over an unreliable channel because the loss of any packets in the base mesh results in an inability to reconstruct the mesh. To remedy this drawback we use forward-error correction (FEC) to encode redundant information into the transmission of some of the packets sent to the destination.

We split the data to be sent into two segments. The first segment consists of the base mesh of the PM representation of the model being sent plus a percentage of the vertex splits, and the second consists of the remaining vertex splits. We use FEC codes to transmit the first segment of the data. Specifically, we use an erasure code based on Vandermonde matrices [37]. The encoding allows us to encode k packets as n packets where $n > k$. As long as we receive at least k of the n encoded packets, we can reconstruct the initial data sent. Importantly, in the encoding that we use, the first k of the n encoded packets are identical to the original data. This means that even if we receive less than k packets, we can still reconstruct some of the packets sent, but will have lost some information.

In our prototype application, we must decide on the values of k and n to use in the transmission of protected data. We could set k equal to the size of the base mesh plus all protected vertex splits. However, this would require the solution of a large matrix equation before sending the data. This matrix solution would add significantly to the time to transmit and in many multimedia applications it would be unreasonable to assume we could precompute these values. Therefore, we chose k to be sixteen packets. We experimentally determined the amount of protection, the value of n, needed to reliably transmit the k original data packets. For our experiments, we assume that we can use *a priori* measurements to determine the necessary protection required to transmit the mesh. This is a reasonable assumption for wireless networks. Loss characteristics for a wireless network will remain constant as long as wireless receivers and transmitters do not move to new locations. *A priori* measurements are unreasonable for WANs, which have fluctuating performance. We will

consider better methods to determine n for WANs in future research. Although it is unreasonable to assume an application could use this mechanism to determine n, our results can be used to determine whether FEC transmission schemes have the potential to improve the performance of the bulk transmission of 3D geometry.

Similar to the hybrid transmission scheme, the application has control over how fast to send information using UDP. The UDP send rate affects the number of dropped packets, because higher transmission speeds increase the congestion on the network. A faster send rate will therefore require a larger value of n to guarantee enough packets are received to reconstruct the sent set of k packets.

4 Experimental Design

This section describes the design of an experimental testbed to explore the use of TCP and several hybrid protocols (TCP/UDP) to transmit 3D geometries of two different models across a congested network. We use a network testbed that allows us to simulate the network behavior of congested links. The testbed consists of a bridge machine and four hosts used to generate background noise traffic across the bridge.

4.1 Testbed Setup

Figure 7 shows the setup of our testbed. Our test environment consists of two user machines (PSnd, PRcv), four background load machines ($NSnd_1$, $NRcv_1$, $NSnd_2$, $NRcv_2$), one bridge machine (BRG) and two 100 Mbps switches (SW_1, SW_2).

	Processor	Memory	OS
PSnd, PRcv	800 MHz AMD[1]	256 MB	Redhat 7.3
$NSnd_1$, $NRcv_1$	800 MHz AMD	256 MB	Redhat 7.3
$NSnd_2$, $NRcv_2$	800 MHz AMD	256 MB	Redhat 7.1
BRG	1 GHz Pentium III	512 MB	FreeBSD 4.3

Table 1 Specifications for machines used in experiments.

The bridge runs the FreeBSD operating system and its kernel is compiled with options Bridge, Dummynet, HZ=1000, and NMBCLUSTERS=10,000. The Bridge and Dummynet options allow the use of this machine to simulate a WAN characteristics for packets sent between SW_1 and SW_2. The HZ option improves the resolution of the operating system's timer. The NMBCLUSTERS option is set to prevent the kernel from running out of mbuf clusters.

The Linux TCP send and receive buffers are initially allocated 16 KB and 85 KB of space respectively. During the course of transmission, the Linux kernel automatically adjusts the send buffer size between a minimum of 4 KB and a maximum of 128 KB, and the receive buffer size between 4 KB and 170 KB, based on actual needs.

[1] AMD/Duron Processor

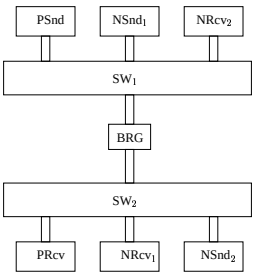


Fig. 7 Configuration of Machines in the Testbed.

As shown in Figure 7, PSnd, $NSnd_1$ and $NRcv_2$ are connected with SW_1. PRcv, $NRcv_1$ and $NSnd_2$ are connected with SW_2. The bridge routes packets between SW_1 and SW_2.

4.2 Simulated WAN

The setup described above simulates a WAN. The WAN is simulated since it provides an environment in which repeatable experiments can be conducted. If our experiments were conducted over a true wide-area network, two different experiments could have significantly different amounts of packet loss and queuing delays. An alternative approach would be to use a model to analytically evaluate the performance of different transmission methods. The analytic technique suffers from the disadvantage that many network models fail to adequately capture the underlying characteristics of real networks and the interactions of complex protocols such as TCP.

Dummynet [36] provides a standard implementation for simulating WANs. It allows users to create pipes of communication with specified bandwidth limits and propagation delays. These pipes are defined by matching header information such as source and destination IP addresses.

In all experiments, Dummynet limits the bandwidth and adds propagation delays to packets sent between switches SW_1 and SW_2. Specifically, we create two pipes, each with a bandwidth limit of 50Mbps and a 25ms propagation delay. The traffic from PSnd, $NSnd_1$ and $NRcv_2$ shares one pipe and the traffic from PRcv, $NRcv_1$ and $NSnd_2$ shares the other. Two pipes are necessary to simulate the out-bound and in-bound bandwidth between the sender and receiver. If the noise generated is not greater than the channel capacity, then packets will not be lost on the simulated network. In other words, a congested network requires more data to be sent than there is channel capacity available.

Our simulated WAN exhibited some of the same behaviors as actual WANs. In particular, by creating a bridge with a limited bandwidth, it simulates a bottle-neck router on the Internet. We observed periods of bursts periods where multiple packets were lost in a row similar to the behavior of actual WANs. The propagation delays used in our simulation were derived from network propagation times sampled from the Internet.

4.3 Simulated Wireless Network

Wireless networks transmit packets using radio signals. These signals are vulnerable to noise and other errors in transmission that may cause packet loss. Typical packet error rates are 2-3% [28] in a wireless LAN.

We experimentally verified the loss rates found by Nguyen [28] by using a wireless computer and sending packets every second to measure the loss rate. Distance from the base station and topology of the building had effects on the loss rate. We saw loss rates from 1-5%.

We simulated a wireless network by using Dummynet and setting the bandwidth and percentage loss rate for the link. This results in random losses rather than the bursts of losses created when the wide-area network was simulated. There was no cross-traffic when running experiments on a simulated wireless network.

4.4 Background Load

The background load generator consists of two UDP components, a sender and a receiver. The background load generator simulates the network characteristics of a large number of external applications, communicating over a shared data link. Previous research has established that the time between packets arriving at a router matches a Pareto distribution [31]. This distribution is *heavy-tailed*, in that most of the inter-arrival times are small but there are periods of significant delay. The Pareto distribution creates self-similar behavior in the network, i.e., there are time scale independent bursts of packet activity.

Each of the two pairs of background load machines has a noise sender and a noise receiver. The noise sender sends UDP packets of size 1400 bytes to the the noise receiver, sleeping for a random amount of time before sending the next packet. The random time it waits follows a Pareto distribution, which favors shorter wait times. This effect causes the packets to have a higher probability of being sent in bursts than evenly spaced.

5 Experimental Results

The network testbed described in the previous section provided a platform that can evaluate the performance of 3D geometric model transmission. While considering PM transmission we focused on several questions about the different schemes:

1. What is the latency cost of transmission?
2. Does TCP transmission increase the variability of the time required to transmit a data set?
3. How much additional bandwidth is required to transmit the mesh with different schemes?
4. Is TCP congestion one of the primary limitations of TCP transmission?

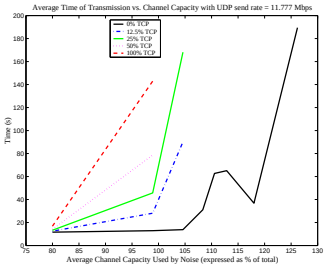


Fig. 8 The average time of transmission of the Buddha model versus noise in the channel for the hybrid transmission scheme.

5. When packet loss occurs and some data is lost, how well does the reconstruction match the original mesh?
6. Do the transmission schemes apply to different meshes?
7. Do the transmission schemes perform similarly in wireless environments?

In the rest of this section, we will answer these questions.

We report results for two models, the “happy Buddha” and a model of a Cessna airplane. The Buddha model contains 1.08M faces in its full model and 1998 faces in its base mesh. The Cessna model has 13546 faces and 338 in its base mesh. Experiments involving the Buddha model consist of six runs of the experiment. Experiments involving the Cessna model consist of ten runs of the experiment. The results are discussed in terms of the average values across a set of runs. Some experiments timed out, i.e., they did not complete, so the average is the average of the experiments that completed in under 200 seconds. In the experiments, we measure the time taken to transmit the progressive mesh, the actual number of vertex splits received by the receiver, and the number of faces in the final mesh reconstructed by the receiver. The number of faces is important because some of the vertex splits that arrive may be unusable because of a previously dropped packet. We also compute the Hausdorff distance from the reconstructed meshes to the original full mesh. Finally, the suite of experiments was run using different send rates for the UDP portion of the model transmission.

5.1 Performance of Transmission Schemes

The primary method to measure performance for mesh transmission in a multimedia system is the time to transfer the geometric model. If too much time is required, the application will not have the model sufficiently soon enough to render the model when needed. In this section, we evaluate the hybrid and FEC schemes and contrast them with the performance of a TCP transmission.

Figure 8 shows these measurements for the hybrid transmission scheme. The graph displays results when TCP was used to transfer the base mesh and 0, 12.5, 25, 50, and 100% of the vertex splits. In these figures, the UDP send rate was 11.777 Mbps. This send rate was chosen because it provides a good trade-off between performance and visual quality of the final mesh. Observe that the lines in the graph have different lengths, because some experiment runs at a given noise level

may have timed out. The important result in Figure 8 is that using TCP solely to transmit the Buddha model takes longer than the hybrid transmission schemes. Moreover, as the percentage of packets sent by TCP decreases, the transmission time improves for all given levels of noise.

The time of transmission for the 0% TCP fluctuates significantly when the background noise applications consume more than 110% of the capacity. Only a single run completed at 100% noise, which is why that value is noticeably outside the curve. The fluctuation is caused entirely by the base mesh that was sent via TCP. This result should not be surprising because experiments [29] have found that when loss rates exceed 10% TCP communication may incur significant delays and may fail.

Using TCP takes longer because of flow control, increased use of bandwidth, and congestion control. The longer latency between sender and receiver combined with the 64 KB maximum receive window places a maximum on achieved throughput. In this case, the theoretical limit is 10.486 Mbps, which means the fastest the Buddha model can be transmitted is 12.476 seconds. TCP's reliability means lost packets will require retransmission by TCP that will increase the total amount of data transmitted over the link. Congestion control might also play a limiting factor in performance. However, typically congestion control is considered a useful attribute so that the congested link does not suffer congestion collapse.

5.2 Variability of Transmission

Of particular concern in interactive and multimedia applications is how much jitter, or variance, occurs in transmission times. The lower the variance, the more accurately the application can predict at what time the full model will be available. In this section, we consider the distribution of running times for the hybrid and TCP transmission of geometric models.

Table 2 provides information about the mean, standard deviation, minimum, maximum, and median of 10 experimental runs, using the same noise and send rates as in Section 5.1. In this set of experiments, we waited for completion of all transmissions. Our results show that there is significant variability in just transmitting the base mesh. However, the UDP portion of the transmission occurs with low variability, which is to be expected since the only change in transmission times between packets will be from queuing delays. Furthermore, if a packet is delayed it will not affect when the next packet is sent.

5.3 FEC Performance

Using TCP to transmit the base mesh can be a significant component of the transmission time as shown in the previous section. This variability led us to explore the FEC transmission scheme that eliminates TCP entirely.

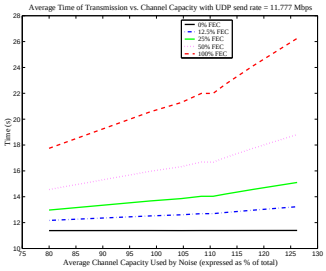


Fig. 9 The average time of transmission of the Buddha model versus noise in the channel for the FEC transmission scheme.

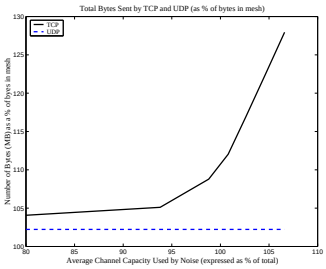


Fig. 10 Total bytes sent by TCP versus noise in the channel. Note that the number of bytes sent by TCP increases as it must resend data due to lost packets.

Figure 9 shows the time to transfer the Buddha model using the FEC scheme. The graph displays lines for varying percentages of the model protected using FEC. The send rate was 11.777 Mbps for all runs. The important result of this experiment is that for high noise levels the FEC scheme significantly decreases the time to send the Buddha model as compared to full TCP or the hybrid transmission scheme (compare Figure 8). Furthermore, all experimental runs completed within 200 seconds when using the FEC scheme. The time to transmit the Buddha model increases as the capacity of the network used by noise increases because we are using more packets to provide protection from packet loss. However, the benefits provided by the FEC transmission scheme come at the cost of additional bandwidth required to send the full mesh. In the next section, we consider the amount of additional bandwidth required for transmission.

5.4 Bandwidth Overhead

Transmitting the mesh requires more bandwidth than the actual size of the mesh. The additional bandwidth is required for IP headers, TCP or UDP headers, and application headers. Reliable protocols use additional bandwidth for retransmissions of lost data and acknowledgments of when data arrived. Figure 10 contrasts the overhead of transmitting the mesh with TCP versus the hybrid transmission scheme where none of the vertex splits are sent via TCP. As the bottleneck link becomes congested, TCP incurs increasing overheads, mostly caused by lost packet retransmissions. This result is not surprising, but serves to reinforce the point that alternate forms of robustness should be considered.

Experiment Type	Mean	Std. Dev.	Min	Max	Median
Full TCP time (s)	642.7591	33.3770	591.9140	692.3270	636.9105
0% TCP, TCP time (s)	9.4525	13.8711	0.7151	39.1077	2.7756
0% TCP, UDP time (s)	11.3319	0.0014	11.3300	11.3349	11.3317

Table 2 Variability of transmission times. The noise level was at 104.6% of channel capacity.

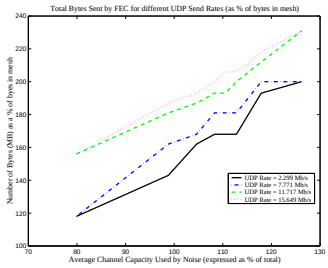


Fig. 11 Total bytes sent by the FEC transmission scheme versus noise in the channel. Overhead is caused by additional protection included in the FEC.

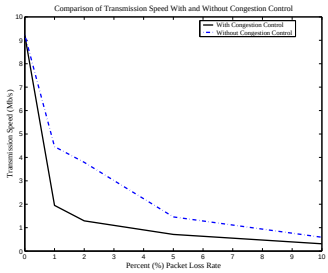


Fig. 12 Effective average send speed before and after kernel modifications are applied to remove congestion control from TCP.

Figure 11 shows the overhead of the FEC transmission scheme. In particular, we show the worst case where 100% of the vertex splits are transmitted with FEC protection. In this case, protection requires significantly more bandwidth than the TCP or hybrid schemes. However, if only 25% of the vertex splits are protected with a FEC, the overhead is reduced to 121.75%, approximately the same as the TCP overhead of 122.38% (at 105% link capacity with 11.777 Mbps UDP send rate).

5.5 Considering Congestion

One of our concerns regarding the hybrid and FEC schemes was how much of the improvement in transmission speed was due to a lack of congestion control. Congestion control is an important component of the TCP protocol that ensures that multiple TCP flows will share bandwidth over a congested link in a fair manner.

We evaluated the effects of congestion control by modifying the Linux kernel's TCP stack. We modified the kernel to remove the congestion window constraint on whether a packet could be sent. We then use Dummynet to introduce packet loss and measured the throughput with and without congestion control. Dummynet also added a 25 ms delay. Figure 12 shows the results of this experiment. The graph indicates that disabling congestion control does provide some improvement in the transmission speed when packets are lost. However, regardless of whether congestion is enabled the curves resemble each other. Thus, congestion control is not the primary cause of the decrease in performance. Flow control's limit on performance should not be affected by packet loss on the network. Therefore, we believe that the cost of retransmitting packets that have been lost significantly contribute to the performance degradation of TCP over congested networks.

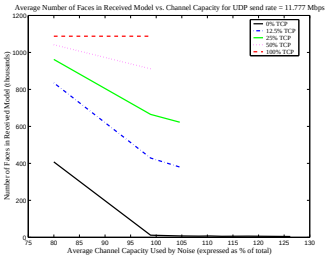


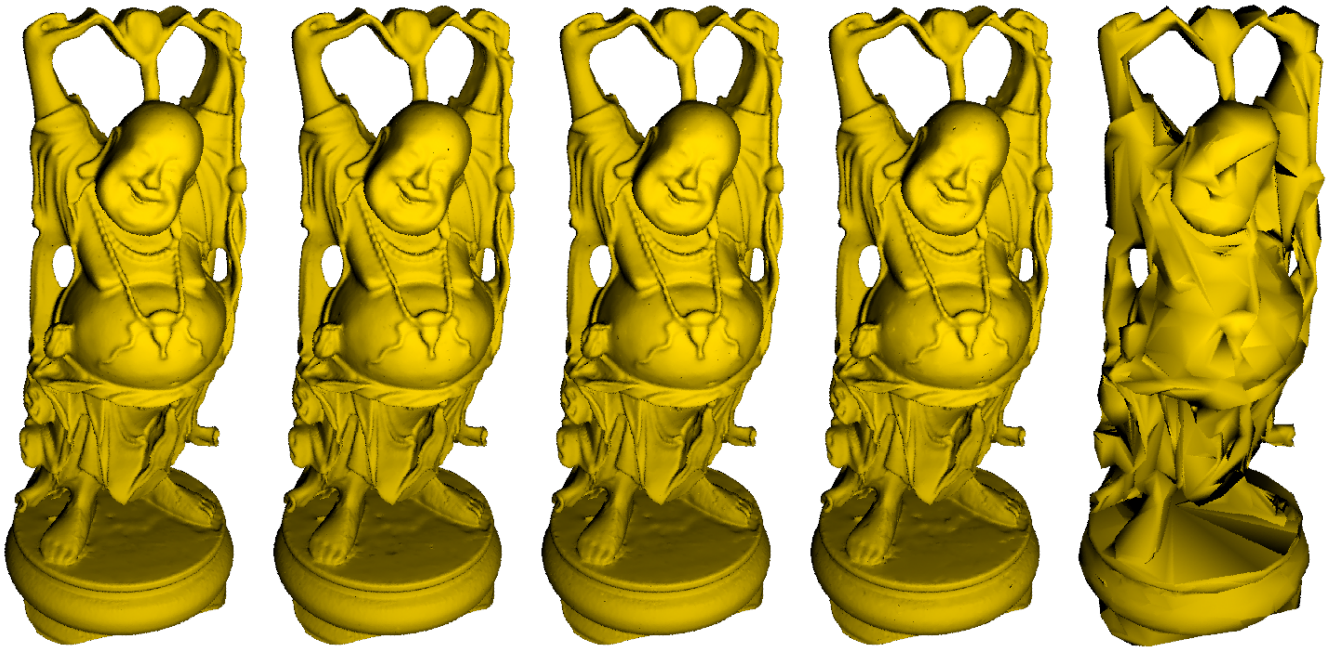
Fig. 13 The number of faces of the Buddha model received versus channel capacity for the hybrid transmission scheme.

5.6 Visual Accuracy of Unreliable Transmission

The previous experiments considered time and bandwidth as the measures of performance. However, we could achieve low transmission times and low bandwidth if nothing was sent. For our application, a transmission scheme that may lose packets must trade-off performance for visual accuracy. In this section, we consider the visual quality of the Buddha model when transmitted using the hybrid and FEC schemes.

Figures 13 illustrates this trade-off. The figure plots the number of received faces in the reconstructed model for different percentages of TCP transmission. Observe that full TCP reconstructs all faces because it is a reliable protocol. When a portion of the packets are sent using UDP, significant packet loss can occur, and this loss becomes worse as the noise increases. When a packet is lost, all splits described in that packet are lost. Also splits that are transmitted later may depend on the lost split and therefore end up being discarded. Starting the UDP transmission earlier will result in fewer faces reconstructed.

Figure 14 shows the visual quality of the FEC transmission scheme. The results are similar to the results of the hybrid transmission scheme because most packets sent with FEC protection will be in the reconstruction. Therefore, loss will only occur for unprotected packets. In Figure 14, more results



(a) (b) (c) (d) (e)

Fig. 15 The Buddha model after transmission over a lossy network: (a) the full model as sent by TCP; (b) model received when 50% transmitted by TCP; (c) model when 25% sent by TCP; (d) model when 12.5% sent by TCP; (e) model when fully sent via UDP.

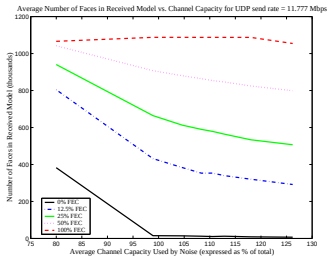


Fig. 14 The number of faces of the Buddha model received versus channel capacity for the FEC transmission scheme.

are available because the FEC scheme does not have the TCP time-out problem during the experimental runs.

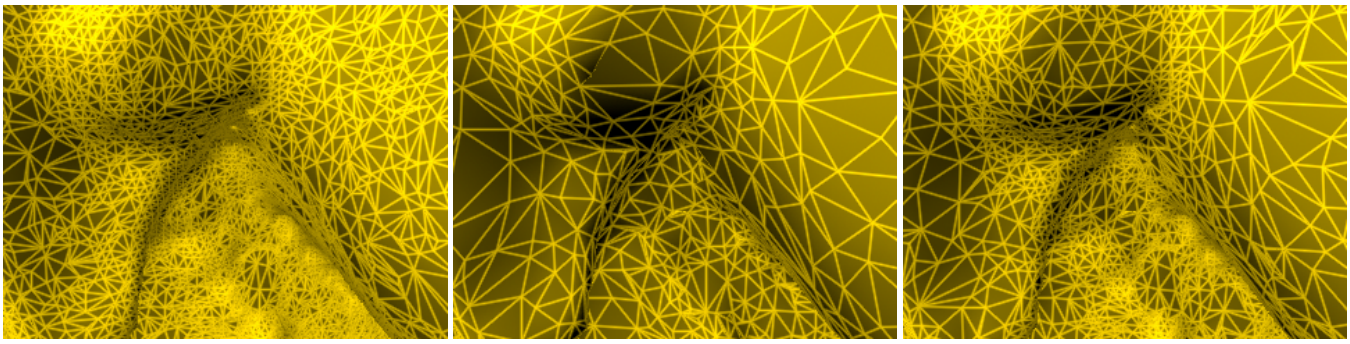
All faces in the Buddha model do not have an equal effect on the user-perceived quality of the output. Any consideration of visual quality must evaluate the visual representations. Figure 15 shows the visual degradation in the Buddha model for the hybrid transmission scheme. These figures were obtained by transmitting across the network testbed when the crosstalk noise application consumed 98.9% of the channel capacity on average (UDP send rate was 11.777 Mbps). Only the transmission of the model with 0% of the splits sent by TCP appears flawed, therefore using 12.5% would transmit an acceptable model yet require only 28.089 seconds instead of the full 142.941 seconds required to send using TCP. Similar results would be found if we looked at the results for the FEC transmission scheme since both schemes have similar numbers of faces in the reconstruction.

How important is the lossy transmission of data using TCP? Figure 16 compares transmitting only the first 12.5% of vertex splits reliably, shown in Figure 16 (b), with the situation where the last 87.5% of vertex splits have been sent using UDP, shown in Figure 16 (c). Using UDP to transmit the remaining vertex splits improves the detail of the mesh significantly. However, the loss of packets means that one does not receive as much detail as the original mesh, shown in Figure 16 (a).

5.7 Different Geometric Models

This experiment addresses whether the transmission scheme performs similarly for different geometric models. We consider the Cessna model, which has fewer faces, as a contrast. The Cessna model should take significantly less time to transmit. When using lossy transmission schemes, loss may have a more significant effect on the visual results.

Figure 17 shows the average transmission time for the Cessna model when using the hybrid transmission scheme. Figure 18 shows the number of faces in the reconstructed model for the same experiment. These results are similar to the results found for the Buddha model. However, there is considerably more variance in the results. This variance is largely due to the significantly smaller number of TCP packets transmitted, because the Cessna model is roughly two or-



(a) (b) (c)

Fig. 16 Close-up wireframe view of the Buddha model: (a) the full model as sent by TCP; (b) model reconstructed after receiving the beginning 12.5% of the vertex splits by TCP; (c) model reconstructed after the rest 87.5% of the splits are sent by UDP.

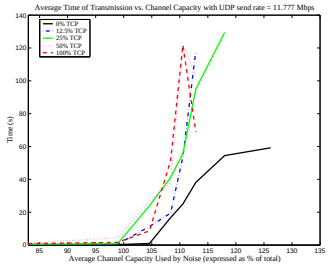


Fig. 17 The average time of transmission of the Cessna model versus noise in the channel for the hybrid transmission scheme.

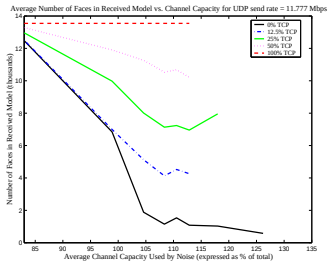


Fig. 18 The number of faces received of the Cessna model versus channel capacity for the hybrid transmission scheme.

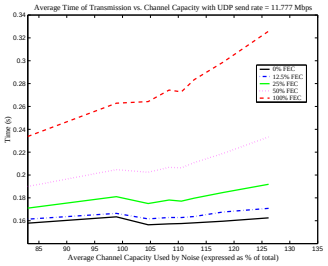


Fig. 19 The average time of transmission of the Cessna model versus noise in the channel for the FEC transmission scheme.

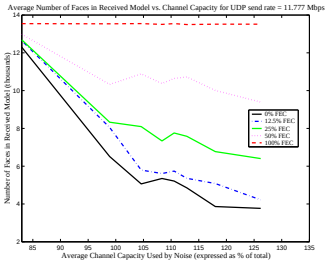


Fig. 20 The number of faces received of the Cessna model versus channel capacity for the FEC transmission scheme.

ders of magnitude smaller than the Buddha model. As a result, the base mesh for the Cessna model only consists of three or four packets. Therefore, if the initial SYN[2] request is lost, the results can be significantly skewed. This skewing occurs because the Linux TCP implementation significantly increases the packet time-out value if one of the SYN packets is lost. Thus, when an additional packet is lost during transmission, a long time-out occurs (on the order of five to fifteen seconds) that significantly affects the results. If the initial packet is not lost, the time-out will only be on the order of 400 ms. Note that we do not count the connection setup time in our results; therefore, our results do not include the SYN time-outs.

[2] TCP uses the SYN packets to initiate a request using a three-way handshake.

The results for the FEC transmission scheme appear in Figure 19 and 20. The significant difference is that since TCP is not used in this situation, the initial fluctuation and increased transmission time do not occur. The number of reconstructed faces is similar to that for the hybrid transmission scheme.

5.8 Wireless: Visual Accuracy

Since we envisioned using the transmission schemes described in this paper for a variety of networked topologies, we need to consider how alternative network conditions will affect the transmission schemes. In this section, we configured the experimental testbed to behave similar to wireless networks. In particular, the round-trip time between sender and receiver

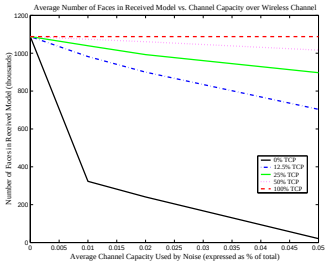


Fig. 21 The average time of wireless transmission of the Buddha model versus the set packet loss rate for the hybrid transmission scheme. The UDP send rate was 10.098 Mbps.

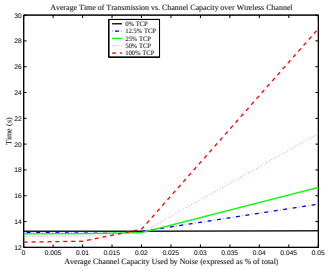


Fig. 22 The average time of wireless transmission of the Buddha model versus the set packet loss rate for the hybrid transmission scheme. The UDP send rate was 10.098 Mbps.

will be significantly decreased, and instead of packet loss occurring in bursts it will be uniformly distributed.

The difference in loss distributions may cause significant differences in the ability to reconstruct the mesh. We first consider the visual quality of the reconstructed meshes when sent over the emulated wireless network. These experiments are run without network crosstalk with an assumption that the bandwidth supported by the network is 11.000 Mbps.

Figure 21 shows results when using the hybrid transmission schemes. We use a send rate of 10.098 Mbps, which is 92% of the available bandwidth over the wireless link. The results are similar to those for wide area networks, with slightly greater faces reconstructed. As long as 12.5% or more of the vertex splits are sent via TCP, a visually accurate model can be reconstructed. The visual quality for the FEC transmission scheme is similar because the same number of packets will be unprotected from packet loss.

5.9 Wireless: Performance

In this experiment, we evaluated the performance as measured in the time to transmit the Buddha model over a wireless network. The same parameters were used in this experiment as in the previous section. Figure 22 shows the transmission time for the hybrid transmission scheme when 0, 12.5, 25, 50, and 100% of the vertex splits are sent via TCP. If the loss rate is below 2% of packets, then TCP outperforms the hybrid transmission scheme because it uses all the available bandwidth on the wireless network. As loss increases, TCP takes increasingly longer to transmit the model and using UDP to transmit some of the vertex splits can reduce the transmission time.

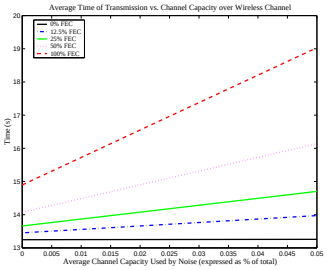


Fig. 23 The average time of wireless transmission of the Buddha model versus the set packet loss rate for the FEC transmission scheme. The UDP send rate was 10.098 Mbps.

The results for the FEC scheme appear in Figure 23. As the packet loss rate increases, more protection is needed for each packet, which increases the time required to receive the model.

6 Conclusions and Future Work

This paper assessed two methods for handling the transfer of 3D geometry across wide area and wireless networks. Hybrid and FEC mechanisms can improve transmission performance when transmitting geometry over lossy channels. This improvement in performance comes at the cost of lost packets. The progressive mesh representation allowed us to minimize the visual impact when packets are lost, although some surface corruption is visible.

We presented results from experiments that compare and contrast using TCP, a hybrid transmission scheme, and a FEC transmission scheme to transmit 3D geometry. We found that reducing the amount of data transmitted by TCP can significantly reduce transmission time and variability. Using FEC requires significantly more overhead than TCP to transmit the mesh in the worst case where we encode the entire mesh. Protecting only a portion of the mesh with FEC transmission reduces the overheads so that the bandwidth used is similar to TCP. A concern is that using UDP eliminates congestion control. We found that TCP behaves poorly even when congestion control is disabled. We also investigated whether the technique was generally applicable and found similar results when used with different models and in different network situations. In future work, we will look into deploying the UDP transmission with support for the Datagram Congestion Control Protocol.

The FEC and hybrid transmission schemes presented use the progressive mesh scheme and do not optimize for the packet grouping that occurs when transmitting using TCP or UDP. As a result, the schemes presented here do not provide the optimal least number of vertex splits discarded due to a loss of a packet.

Another limitation of these schemes is the requirement that the application select the transmission speed. This requires hand-tuning for different environments. The hand-tuning

should select an appropriate UDP send rate relative to the available bandwidth. Future work will look at ways to determine the proper UDP send rate.

There remain several open questions and challenges for future work. One potential problem of the hybrid and FEC transmission schemes is that UDP is used to transfer a portion of the geometric model. Using UDP avoids the congestion control provided by TCP. Extending the hybrid and FEC transmission mechanisms to handle congestion would be a valuable enhancement. The work in this paper does not attempt to determine whether the cause of packet loss is congestion in a bottleneck network link or between a wireless device and a wireless access point. An additional modification of the algorithm is to incorporate feedback from the receiver to conditionally resend packets that will force the algorithm to discard future packets that will be sent, or even alternatively, avoid sending these packets that will be discarded anyways.

References

1. Al-Regib, G. and Y. Altunbasak: 2002, 'An unequal error protection method for packet loss resilient 3-D Mesh Transmission'. In: *Proceedings of INFOCOM 2002*.
2. Allen, A. D.: 2001, 'Optimal Delivery of Multi-Media Content over Networks'. In: *Ninth ACM Multimedia Conference*. Ottawa, Ontario, Canada, pp. 79–88.
3. Alliez, P. and M. Desbrun: 2001, 'Progressive Compression for Lossless Transmission of Triangle Meshes'. In: *Proceedings of ACM SIGGRAPH 2001*. pp. 195–202.
4. Aspert, N., D. Santa-Cruz, and T. Ebrahimi: 2002, 'MESH: Measuring Errors Between surfaces using the hausdorff distance'. In: *Proceedings of the IEEE International Conference in Multimedia and Expo (ICME)*. pp. 705–708. http://mesh.epfl.ch.
5. Bischoff, S. and L. Kobbelt: 2002, 'Streaming 3D Geometry over Lossy Communication Channels'. In: *Proceedings of the IEEE International Conference on Multimedia and Expo*.
6. Blake, S., D. L. Black, M. A. Carlson, E. Davies, Z. Wang, and W. Weiss: 1998, 'An Architecture for Differentiated Services'. Request for Comments, RFC 2475.
7. Braden, R., D. Clark, and S. Shenker: 1994, 'Integrated Services in the Internet Architecture: an Overview'. Request for Comments, RFC 1633.
8. Byers, J. W., M. Luby, M. Mitzenmacher, and A. Rege: 1998, 'A Digital Fountain Approach to Reliable Distribution of Bulk Data'. In: *ACM SIGCOMM'98 Conference. Applications, Technologies, Architectures and Protocols for Computer Communication.*, Vol. 28(4) of *Computer Communication Review*. Vancouver, BC, Canada.
9. Casner, S. and S. Deering: 1992, 'First IETF Internet Audiocast'. *ACM Computer Communication Review* **22**(3), 92–97.
10. Chen, B.-Y. and T. Nishita: 2002, 'Multiresolution Streaming Mesh with Shape Preserving and QoS-like Controlling'. In: *Proceeding of the 7th International Conference on 3D Web Technology*. pp. 35–42.
11. Chen, Z., B. Bodenheimer, and J. F. Barnes, 'Extending Progressive Meshes for Use over Unreliable Networks'. To appear at ICME 2003.
12. Chen, Z., B. Bodenheimer, and J. F. Barnes: 2003, 'Robust Transmission of 3D Geometry over Lossy Networks'. In: *Proceedings of the 8th International Conference on 3D Web Technology*. St. Malo, France, pp. 161–173.
13. Conrad, P. T., A. Caro, and P. Amer: 2001, 'ReMDoR: Remote Multimedia Document Retrieval over Partial Order Transport'. In: *Ninth ACM Multimedia Conference*. Ottawa, Ontario, Canada, pp. 169–180.
14. Everquest. http://everquest.station.sony.com.
15. Floyd, S.: 2003, 'HighSpeed TCP for Large Congestion Windows'. Internet draft.
16. Floyd, S., V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang: 1997, 'A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing'. *IEEE/ACM Transactions on Networking* **5**(6).
17. Frécon, E., C. Greenhalgh, and M. Stenius: 1999, 'The DIVEBONE — An Application-Level Network Architecture for Internet-Based CVEs'. In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*. London, UK, pp. 58–65.
18. Gandoin, P.-M. and O. Devillers: 2002, 'Progressive Lossless Compression of Arbitrary Simplicial Complexes'. *ACM Transactions on Graphics* **21**(3), 372–379. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).
19. Garlick, L. L., R. Rom, and J. B. Postel: 1977, 'Reliable host-to-host protocols:Problems and techniques'. In: *Proceedings of the Fifth Data Communications Symposium. Applications, Technologies, Architectures, and protocols for Computer Communication*. Snowbird, Utah, USA.
20. Hoppe, H.: 1996, 'Progressive Meshes'. In: *Proceedings of SIGGRAPH 96*. New Orleans, Louisiana, pp. 99–108. ISBN 0-201-94800-1.
21. Hoppe, H.: 1997, 'View-Dependent Refinement of Progressive Meshes'. In: *Proceedings of SIGGRAPH 97*. Los Angeles, California, p. I198. ISBN 0-89791-896-7.
22. Hoppe, H.: 1998, 'Efficient implementation of progressive meshes'. *Computers & Graphics* **22**(1), 27–36. ISSN 0097-8493.
23. Jin, C., D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, and S. Singh: 2003, 'FAST TCP: From Theory to Experiments'. *Submitted to IEEE Communications Magazine, April 1, 2003*.
24. Kelley, J. L.: 1955, *General Topology*, No. 27 in Graduate Texts in Mathematics. New York: Springer-Verlag.
25. Kelly, T.: 2002, 'Scalable TCP: Improving Performance in Highspeed Wide Area Networks'. Submitted for publication. See `http://www-lce.eng.cam.ac.uk/~ctk21/scalable/#publications`.
26. Khodakovsky, A., P. Schröder, and W. Sweldens: 2000, 'Progressive Geometry Compression'. In: *Proceedings of ACM SIGGRAPH 2000*. pp. 271–278. ISBN 1-58113-208-5.
27. Macedonia, M. R., M. J. Zyda, D. R. Pratt, D. P. Brutzman, and P. T. Barham: 1995, 'Exploiting Reality with Multicast Groups'. *IEEE Computer Graphics and Applications* **15**(5), 38–45.
28. Nguyen, G. T., R. H. Katz, B. Noble, and M. Satyanarayanan: 1996, 'A Trace-Based Approach for Modeling Wireless Channel Behavior'. In: *Proceedings of the Winter Simulation Conference*. Coronado, Canada.
29. Padhye, J., V. Firoiu, D. Towsley, and J. Kurose: 1998, 'Modeling TCP Throughput: A Simple Model and its Empirical Validation'. In: *ACM SIGCOMM'98 Conference. Applications,*

Technologies, Architectures and Protocols for Computer Communication., Vol. 28(4) of *Computer Communication Review*. Vancouver, BC, Canada.

30. Pajarola, R. and J. Rossignac: 2000, 'Compressed Progressive Meshes'. *IEEE Transactions on Visualization and Computer Graphics* **6**(1), 79–93. ISSN 1077-2626.
31. Park, K., G. Kim, and M. Crovella: 1996, 'On the Relationship Between File Sizes, Transport Protocols, and Self-Similar Network Traffic'. In: *Proceedings of the Fourth International Conference on Network Protocols (ICNP'96)*.
32. Pejhan, S., T. hao Chiang, and Y.-Q. Zhang: 1999, 'Dynamic Frame Rate Control for Video Streams'. In: *Seventh ACM Multimedia Conference*. Orlando, FL, USA.
33. Popović, J. and H. Hoppe: 1997, 'Progressive Simplicial Complexes'. In: *Proceedings of SIGGRAPH 97*. Los Angeles, California, pp. 217–224. ISBN 0-89791-896-7.
34. Postel, J.: 1981, 'Transmission Control Protocol'. RFC 793, `http://www.rfc-editor.org/rfc/rfc793.txt`.
35. Rejaie, R., M. Handley, and D. Estrin: 1999, 'Quality Adaptation for Congestion Controlled Video Playback over the Internet'. In: *ACM SIGCOMM Conference. Applications, Technologies, Architectures and Protocols for Computer Communications.*, Vol. 29(4) of *Computer Communication Review*. Cambridge, MA, USA.
36. Rizzo, L.: 1997a, 'Dummynet:a simple approach to the evaluation of network protocols'. *ACM SIGCOMM Computer Communication Review* **27**(1).
37. Rizzo, L.: 1997b, 'Effective Erasure Codes for Reliable Computer Communication Protocols'. *ACM Computer Communication Review* **27**(2), 24–36.
38. Sinha, P., T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan: 2002, 'WTCP: A Reliable Transport Protocol for Wireless Wide-area Networks'. *Wireless Networks* **8**(2/3).
39. Taubin, G., A. Gueziec, W. Horn, and F. Lazarus: 1998, 'Progressive Forest Split Compression'. In: *Proceedings of SIGGRAPH 98*. Orlando, Florida, pp. 123–132. ISBN 0-89791-999-8.
40. Zhang, L., S. Deering, D. Estrin, S. Shenker, and D. Zappala: 1993, 'RSVP: A New Resource ReSerVation Protocol'. *IEEE Network* **7**(5), 8–18.