The Design of a Transport Protocol for On-Demand Graphical Rendering

Albert F. Harris, III[*]
University of Illinois Urbana - Champaign
aharris@uiuc.edu

Robin Kravets
University of Illinois Urbana - Champaign
rhk@cs.uiuc.edu

ABSTRACT

In recent years there have been significant advances in 3D scanning technologies that allow the creation of meshes with hundreds of millions of polygons. A number of algorithms have been proposed for the display of such large models. These advances, coupled with the steady growth in the speed of network links, have given rise to a new area of interest, the streaming transmission and visualization of three-dimensional data sets. We present a number of issues that must be addressed when designing a transport protocol for three-dimensional images as well as a method of transport. We then evaluate an implementation of an On-Demand Graphic Transport Protocol (OGP) that addresses these issues.

Categories and Subject Descriptors

C.2.2 [**Computer**]: Communication Networks

General Terms

Design

Keywords

transport protocol, partial reliability, partial order, 3-D models, streaming

1. INTRODUCTION

With the enormous increase in computing power of today's home computers, it is now possible for complex three-dimensional models to be used in home applications. The growth of the Internet introduces the possibility of using such three-dimensional models on websites and online applications. The size of the models, with the corresponding long download times, prohibits transferring the models completely before rendering begins; therefore, efficient methods for streaming three-dimensional models must be developed if the models are to be used in online applications. The use of a streaming protocol allows the user to begin viewing a portion of the model before transmission is completed. This type of

streaming is, however, distinct from the problem of video streaming. Videos can be loss-tolerant and are linearly ordered. By contrast, three-dimensional images are not linearly ordered. Given this information, we observe that while video streams have time-dependencies between video segments, three-dimensional images have space-dependencies between picture segments. This leads to radically different methods for optimal transport.

A common scheme for rendering three-dimensional models is based on a bounding volume approach. The data structure containing the models is a tree with certain properties, such as loss-tolerance yielded by the partial ordering of the data structure, that can be leveraged to facilitate efficient streaming. The nodes of the tree represent bounding volumes of the model with the roots of the tree representing spheres that encompass the entire model. Nodes lower in the tree represent successively smaller volumes, the leaves being the smallest resolution points. The dependencies in the tree run along the branches. Each node is dependent only on its parent node and there is no ordering among siblings in the tree. This allows a certain flexibility in the reliable transport of the model because losses only affect subtrees rooted at the lost nodes. Another facet of the tree is that greater resolution is achieved by running down the tree, therefore it is possible to achieve a lower resolution version of the model by only streaming down to a certain level within the tree. The ability to send only part of the tree for rendering can be leveraged to control the quality of the model based on the bandwidth available to the user. A further property of the tree is that particular branches in the tree correspond to particular spatial sections of the picture. It is possible to take advantage of this fact and render exclusively, or more completely, the sections of the image that are of interest to the user, perhaps by allowing the user to mouse-over the portion of the model in which they are interested [14].

We therefore contribute an analysis of the issues involved in developing an efficient transport protocol for streaming three-dimensional models and the design of the On-Demand Graphic Transport Protocol (OGP). The standard Internet transport protocols are not appropriate for streaming such models. TCP's [11] full reliability can create unnecessary delays in the stream, but UDP [12] does not provide the rendering application the reliability it requires to maintain the partial ordering [3]. Therefore we present a protocol that maintains the partial ordering required by the rendering applications. We present a good node packing algorithm that can maintain properties such as loss-tolerance constant throughout transport, thereby allowing the transport protocol to continue to exploit these properties. We demonstrate that better performance can be obtained by a transport protocol by taking advantage of these properties.

The rest of this paper is organized as follows. In Section 2, we describe approaches to streaming media in general, discussing their limitations. In Section 3, we present our analysis of the tree structured three-dimensional models and the design of an efficient streaming protocol for three-dimensional models. In Section 4, we present the parameters used in designing an efficient protocol for three-dimensional model transport. We then present the implementation of the On-Demand Graphic Transport Protocol (OGP). In Section 5 we present the results of the evaluation of OGP. In Section 6, we present some conclusions about three-dimensional model streaming.

2. STREAMING MEDIA

The sheer size of media, such as entire videos and three-dimensional models, leads us to consider new representations that enable intelligent partitioning of the data into smaller application-layer data units [4]. Such partitioning supports the streaming of data, providing the benefits of pipelining data transmission with data presentation. This new type of media is often progressive, with each additional piece of data adding to what has already been recieved. Protocols such as RTP [15] make use of application level framing. Many such protocols are integrated directly into the application, which would then sit on top of UDP. This allows the applications to be run without changes to the current Internet technologies. This research in streaming media has mainly focused on video as the media of choice. RTP, for example, is tailored to handle real-time requirements that are inherent in video. The streaming of three-dimensional models introduces different requirements for the transport of model data based on the structure of the data.

2.1 Video

Since a video is the progressive representation of images, streaming video is naturally represented by a sequence of frames or groups of frames for optimization. These frames represent a fully linear sequence. Loss cannot be tolerated within a frame, but each of these frames is an independent entity, enabling streaming video protocols to tolerate the loss of some frames. Essentially, these protocols are memory-less. For example, in MPEG encoding, the I-, P-, and B-frames must be in order to be usable but the stream can tolerate the loss of a B-frame affecting only the subsequent B-frames until the reception of the next I-frame [6]. There are also strict time considerations with video. Once the play-out time for the frame following a lost frame expires, it is too late to do anything with that lost frame. Such partitioning of video divides it into application data units that have temporal locality. Each group of pictures is related to the others in time, not space.

The usefulness of designing image models that allow losses in particular parts of the image, thereby allowing the continued transfer and display of the rest of the image, with the lost piece appearing as a small blank has been shown to help intra-frame loss [16]. This can be seen as a motivating step toward the structuring of models using spatial partitioning as opposed to temporal partitioning. Three-dimensional modeling uses this concept extensively.

2.2 Three-Dimensional Models

The design goal of three-dimensional models is to provide a representation of an object for presentation to the user. In order to provide a complete representation, all data must be available for presentation. In order to support more flexible presentation of data, these models have been designed in a progressive manner; the more of the data that is available, the better the quality of the presentation. The three-dimensional models are divided into application data units that have spatial locality. Each node is related to the oth-

ers with respect to their relative positions in the actual model. Such models inherently require memory of all available data. Model representations based on the bounding volume approach have a tree-based structure. This structure provides a partial ordering on the data. Each node that refines a particular spacial location on the model is dependent on all previous nodes that involve that location. If a node from one part of the model is lost, this does not affect other sections of the model. In this way, the data is non-linear. The partial ordering can be leveraged to make significant performance gains in the face of loss [2, 3, 8]. It is possible to maintain the stream efficiently by carefully keeping track of the dependencies existing between parts of the data structure [9].

When a loss is encounter in a model, any received children of the lost node cannot be rendered. If the rendering is on-demand, then the time delay in rendering caused by the transmission of nodes that cannot be immediately rendered will be perceived. In order to avoid this, nodes that are dependent on the loss should not be transmitted until the lost node is re-transmitted and received. This implies that rapid loss detection is essential to efficiently stream models for on-demand rendering. To this end, there has been some work in finding effective ways to detect losses early. Papadopoulos, et. al. [10] use gap-based loss detection as opposed to timer-based loss detection. This minimizes the latency for loss detection. While the time constraints for three-dimensional models are different than for video, it will still prove to be very useful to minimize the time it takes to detect a loss. By detecting losses early and transmitting only sections of the data structure that can be rendered, the door is opened for the possibility of not re-transmitting lost data immediately if it is not desired. This amounts to varying the level of reliability for the transfer of each packet. Varying the reliability levels dynamically during transmission of data has been shown to be useful in creating better efficiency in the transfer of data that can sustain some loss [7].

One approach to streaming three-dimensional models is to transmit data progressively with lower resolution data being transmitted first and then streaming higher-resolution data to fill in the details. One approach [13] uses the HTTP/1.0 protocol to request data from a standard web server, such as Apache, that would send requested bits of data to the client. We believe this approach is not optimal because it does not make use of the properties of the data structure.

Since node size will not always be the same as packet size, it is important to consider how to pack the models into packets. The problem is essentially one of linearization. The tree in which the three-dimensional models is not linear, it is partially ordered, but transmission over the network requires linearization. If the data is not intelligently linearized, we can end up transmitting data that can not be used.

3. APPLICATION DATA PROPERTIES

Today, the transport of data across the Internet is achieved primarily through the use of two transport protocols, TCP and UDP. TCP provides a fully reliable means of communication that guarantees in-order delivery of all packets sent without duplication. UDP, on the other hand, provides no reliability guarantees at all, packets may come out of order or even not at all. With the widespread use of these two transport protocols, it is tempting to take a very binary, one-dimensional view of transport service. The tendency is to either see the data transmission as reliable and fully ordered or not reliable and unordered. The main problem is that this cuts off a large portion of the possible space. Really, transport service should be seen as a two dimensional continuous function involving guarantee of delivery and ordering as in Figure 1 [2, 3, 7, 8].
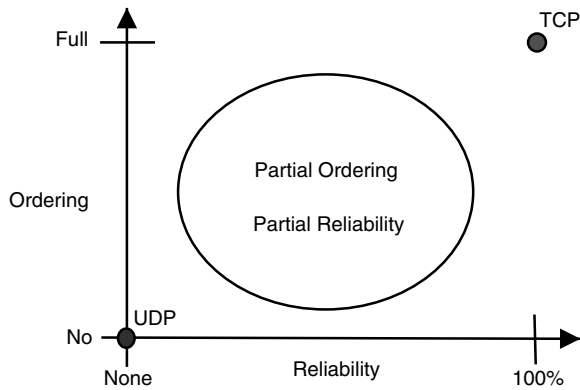
Figure 1: Transport Service Space

Figure 2: Partially Ordered Tree

3.1 Exploring the Service Space

Once the transport service space is expanded, the impact of using the incorrect transport service level can be analyzed. There are three possible relationships between the transport service level used in the transmission of data and the actual need of the application requiring the transmission.

1. The level of transport service is optimal for the application.

2. The level of transport service is too high for the application.

3. The level of transport service is too low for the application.

For the second two relationships, there are two different aspects in which the transport service level could be seen as too high or low for an application.

1. The ordering requirements can be too strict or lax.

2. The loss tolerance can be too strict or lax.

The various combinations of possibilities are derived directly from the expanded transport service space. The effects of the situations, however, must be discussed systematically.

3.2 Mapping Channel Service to Application Requirements

If an optimal transport service level is chosen for the transport of an application's data, it is clear that there will be no wasted resources providing services that the application does not require. It is also clear that the application will be able to function since all of its service requirements would be met by the optimal matching.

The interesting cases to study are the sub-optimal cases. If there is no significant impact on performance when a sub-optimal choice in transport service level is chosen, there would be no reason to consider alternative transport mechanisms to TCP.

If the reliability level required by the application is not met, it will be impossible for the application to produce correct results. The actual effects on the application are scenario specific, but can range from total failure to incorrect functionality [9]. Similar effects can be seen from an inappropriate ordering constraint. If the application's ordering requirements are more strict than the transport layer can provide, then again the application will fail to function correctly. For three-dimensional modeling applications, it is not possible for the rendering engines to deal with loss or complete mis-ordering, therefore it is clear that UDP will not provide a sufficient level of transport service.
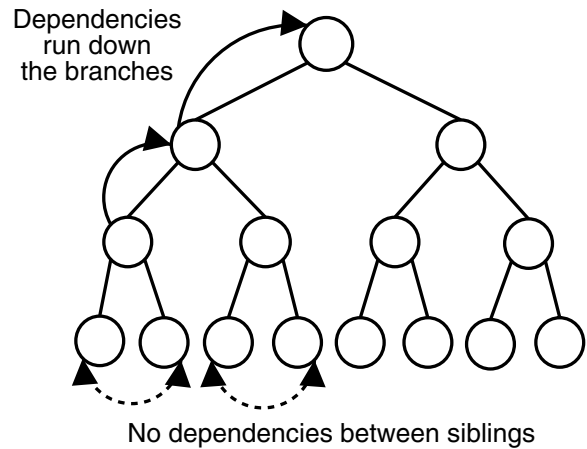
If the transport service level required by the application is surpassed by the network, unnecessary delays and a loss of goodput is experienced. A similar result occurs when the ordering constraints of the network are too strict. If this happens, it is possible that some data will be ready to be sent but will have to wait due to false ordering constraints. In this way, both throughput and goodput will suffer [9].

3.3 Maintaining Application Data Properties

While there has been research into designing protocols that can handle partial orderings [2, 3, 8], the protocol to handle streaming bounding-sphere based three-dimensional models must be specially tailored. Current partial order protocols address the ordering of packets, but not how data will be organized into packets. Given that a node in a three-dimensional model may not fill an MTU, it may be necessary to pack multiple nodes in one packet. Such packing can greatly improve the performance and efficiency of the transmission of the model, but must be accomplished within the constraints of the original ordering. Also, due to the very specific nature of the tree and its partial ordering, a specialized transport scheme can enable on-demand improvement of specific, user-chosen parts of the model.

We now analyze the tree used to store the models in bounding sphere methods of rendering. The only ordering constraint is that for any child received, its parent must have been received first. Essentially, there must be complete ordering only down the legs of the tree, as in Figure 2.

The nodes of the tree represent the smallest data units that can be drawn, with each node relying on its parent. The leaves of the tree represent the individual pixels fully rendered. The nodes of the tree can vary between a few bytes and a few hundred bytes. Because of this small size, more than one node can be put in each packet. A decision must be made on the order in which to pack the nodes together. The main constraint is that transmission must not continue too far down any path in which one of the parent nodes has been lost or not yet been acknowledged. This problem can be avoided by only allowing new nodes to be sent after an acknowledgment has been received for the node's parent. Such information can be integrated into the flow control mechanism as discussed in Section 4.1.

Node packing is very important in order to maintain the partial ordering throughout transmission of the model. We note that any subtree of the tree will have the same data structure properties as

the whole tree. Therefore, in order to be able to leverage the properties that have been discussed throughout transport, nodes must be packed in such a way as to not break the subtree relationships across packets. Packing the nodes of the tree in a subtree-by-subtree order will effectively maintain the partial ordering of the tree throughout transmission. The details of node packing are discussed in Section 4.1.

4. AN ON-DEMAND GRAPHIC TRANSPORT PROTOCOL (OGP)

There are fundamental design decisions that must be made in order to optimize a protocol for the transport of three-dimensional models. The inherent loss tolerance and partial ordering of the tree should be leveraged in order to efficiently stream the models. The models can sustain no loss along any branch. However, due to the fact that there are no dependencies between siblings in the tree, if a loss occurs, streaming of other branches in the tree may continue without interruption. The inherent loss tolerance of the data structure can be leveraged to allow the transfer of data to continue in the face of loss. The partial ordering of the data structure can be leveraged to allow the protocol to continue to transfer the data smoothly in the face of lost or reordered packets. The partial ordering can also be used to allow the protocol to take into account user focus information and to transfer only the part of the model that is currently in focus. The protocol must also contain some method of deciding when it is appropriate to retransmit lost packets and which packets to retransmit. Finally the protocol must package the nodes in such a way as to maintain the properties of the data structure throughout the transfer. In the following subsections, we present our implementation of a three-dimensional model streaming protocol, OGP. We use OGP to illustrate how these issues in protocol design can be resolved.

OGP is a self-clocked protocol that makes use of TCP congestion control algorithms [1]. Self clocking is achieved by allowing the receipt of an acknowledgment to trigger the transmission of new packets. In the current version of OGP, every packet is acknowledged, therefore, the loss of an acknowledgment implies the loss of a packet. This problem could be fixed by using selective-acknowledgments instead of single-packet acknowledgments. Congestion control is managed through the use of a window mechanism that monitors the number of outstanding packets, as discussed in Section 4.2.

Because it is possible for multiple nodes to be sent in one packet, the sender maintains a mapping between nodes and packets. In this way, the receiver can acknowledge packets and the sender can then map these acknowledgments onto acknowledgments of nodes, which are used by the packing algorithm as described in Section 4.1.

4.1 Node Packing

In order to take advantage of the properties of the tree, the partially ordered data structure must be linearized and sent across the network while maintaining the inherent properties. In order to do this, the partial ordering relations of the tree, as well as the spacial relations of the tree nodes, should be used. OGP begins by packing the largest possible subtree at the root. After this first packet has been acknowledged each node sent in that packet can now be used as a root of a new subtree to send. OGP continues by packing the children of the acknowledged nodes as roots of largest subtrees in breadth-first order [Figure 3]. Essentially, the largest possible next subtree is being packed in each successive packet. In the event of a loss, OGP avoids sending any of the children dependent on the lost nodes until the lost nodes have been successfully re-transmitted.

The goal is to maintain a subtree-by-subtree transmission pattern in order to keep the partial ordering and reliability properties of the entire model constant throughout transfer.

In order to keep track of what nodes have been sent, markers are added into the data structure that can be set as *sent* and *ackd*. As the tree is traversed breadth-first while packing nodes, the nodes are marked as *sent*. When a packet is acknowledged, each node contained in the packet is marked as *ackd*. Pointers are kept to the first node in the tree that is not acknowledged and the current node needing to be sent. In this way nodes needing to be marked and nodes needing to be re-transmitted can be found efficiently while still walking forward in the tree.

The packing algorithm works as follows. Get_Next_Node takes the current_sub_tree as the root of a tree and increments node in breadth-first order using a standard tree walking algorithm. Get_Next_Subtree increments current_sub_tree to the next unmarked node in the tree using a standard breadth-first tree walking algorithm.

```
current_sub_tree = root_of_tree;
node = root_of_tree;
while(node != end_of_tree)
{
    while(!Packet_Full(packet))
    {
        Append(node,packet);
        Mark(node);
        Get_Next_Node(current_sub_tree,node);
    }
    Send(packet);
    Empty(packet);
    Get_Next_Subtree(current_sub_tree);
    node = current_sub_tree;
}
```

To illustrate the algorithm, consider an example of a three-dimensional model with node sizes of 150 bytes. The tree has a branching factor of 4. Consider a link over the Internet between a client and server with a delay of 20ms and a bandwidth of 256Kbits. The Ethernet frame size is 1500bytes. 10 nodes fit per packet and there can be at most 5 packets in flight between the last acknowledged and the last sent packets. Given this information, if the first nodes are packed in breadth-first order in the first packet, there will be 15 nodes that can be rendered but have not yet been sent. If nodes are continued to be packed breadth-first, there are possible cases where, after a loss, data will still be sent that cannot be rendered because the loss will not yet be noticed on the sender side. This problem will be exacerbated by smaller nodes sizes, such as Qsplat's 4 byte node sizes [14]. If however, the node packing is done intelligently, the branching factor of the tree can be leveraged to prevent the possibility of transmitting data that cannot be immediately rendered, see Figure 3.

4.2 Congestion Control

The loss tolerant nature of the data structure allows OGP to continue transmitting data in the face of losses. The only invariant that must be maintained is that no node whose parent has not yet been received should be sent. The packing algorithm in Section 4.1 insures that the ordering properties of the tree will be maintained throughout transmission. A flow control algorithm is needed, however, to insure that nodes are always available whose parents have already been acknowledged when it is time to send. OGP should also react to congestion appropriately so that it is TCP friendly.
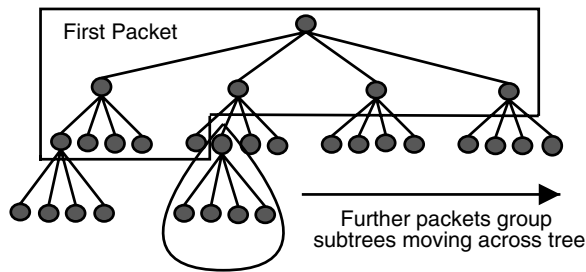
Figure 3: Intelligent Packing Order

For congestion control, OGP reacts much the same way as TCP New-Reno, making use of the slow-start, congestion avoidance, fast recovery, and fast retransmit algorithms common to TCP [1]. Changes have been made to account for the fact that it does not need to actually retransmit a lost packet. OGP does require losses to be identified by the sender, however, so that nodes can be marked correctly. OGP uses a TCP like sequence numbering scheme. The receiver always acknowledges the sequence number received and not the sequence number of the next packet expected. This is necessary because packets are not retransmitted so a lost sequence number will never be received.

OGP has a congestion window, `cwnd`. The `cwnd` is started at 1 and is increase according to the TCP New-Reno algorithms. According to these algorithms, `cwnd` is never increased by more than one at any received acknowledgment. The `cwnd` is reduced due to loss according to the TCP New-Reno algorithms.

A loss is detected when the acknowledgment received at the sender is not the next acknowledgment expected. In order to relate a packet loss to a node loss, the sender maintains a mapping between packets and nodes that are in flight. When a loss occurs, the sender refers to this mapping to decide which nodes were lost in the lost packet. The nodes in the lost packet are then marked *unsent*. It is possible that packets that are reordered will be considered lost. This is not a real problem, however, because OGP does not immediately retransmit lost packets, therefore when the packet is eventually received and acknowledged, the nodes will be marked *ackd*. OGP also has a time-out mechanism to allow it to recover in the face of losses of acknowledgments like TCP New-Reno.

Because packets may be lost and never retransmitted, a standard sliding window, such as the one TCP uses, cannot be used. Instead only the number of packets still in flight need be considered, which can be calculated based on the last packet acknowledged and the last packet sent. This number is compared with the value of `cwnd`. When the number of packets in flight is less than the congestion window, OGP can send. Each time an acknowledgment is received, OGP will send as many packets as it can.

An important issue is to make sure that there are always nodes whose parents have been acknowledged with which to build the next subtree to send in a packet. Consider that a packet is only sent after an acknowledgment is received. Also consider that the congestion window is never opened by more than 1. Then, in the face of no losses, there will never be more than 2 packets sent per acknowledged packet. Therefore, as long as the data structure has a branching factor of 2 and there is at least 1 node per packet, there will always be nodes to send. In the face of losses, it is possible that more than 2 nodes may be sent after an acknowledgment. However, the congestion window is halved when a loss is encountered for each round-trip-time. Therefore, there will not be more packets to send until a number of new acknowledgments have arrived. Therefore, either the branching factor of the structure must be at

least four, or there should be at least three nodes per packet. The largest node size that we have observed in the three-dimensional modeling schemes using bounding volumes is 300 bytes and the branching factors are typically 4. It is not unreasonable to assume that 4 nodes fit per packet, giving the needed 4 nodes to send per acknowledgment even assuming only a branching factor of 2.

4.3 Exploiting the Partial Ordering

The partial ordering of the data structure combined with the node packing algorithm allows OGP to insure that the packets in flight are not dependent on one another. Because of this, OGP does not need a standard congestion-window and can focus on the number of nodes in flight. Therefore, OGP can continue to send in the face of packet reordering and loss. The only reason OGP should slow transmission is in response to congestion. Therefore, OGP should see more throughput than TCP. This result is achieved as shown in Section 5.

The partial ordering of the data is also used to allow the protocol to send data that is relevant to the user's focus. Each node in the tree represents a particular bounding volume that covers a particular section of the model. The user's focus can be given to the protocol as the identity of the node that covers the section of interest. It is then possible, due to the partial ordering of the data, to only send the line of nodes down the branch that leads to the node covering the area of the model on which the user is focused. The subtree of the model rooted at this user focus defined node can then be sent using the normal transport algorithm.

4.4 Deciding What to Transmit

Aside from sending without the retransmission of lost packets as described so far in the paper, the protocol also has the ability to use some its throughput to retransmit some or all of the lost data. We implemented full-reliability by marking lost nodes as *lost* in the data structure and first packing those nodes into packets to be sent. While these nodes will not normally reveal new nodes that can be sent, this is not a problem for the protocol as there were nodes that would have been sent that were not, had the protocol been functioning with no reliability.

Another parameter to consider in transmission is user demand. The reliability level of a certain part of the tree can be changed depending on user focus. If there is a particular part of the model that the user is interested in, then OGP can immediately re-transmit any lost nodes from the corresponding section of the data structure. This mechanism works the same way that full-reliability does, except that only nodes in the part of the model that are of interest to the user are marked *lost* and are therefore retransmitted.

User focus could also be used to partition out one part of the model to be filled in faster than the other parts, using all available bandwidth to concentrate on the refinement of relevant parts of the model if this is desired. Again, the part of the model that is of interest to the user can be viewed as a subtree of the data structure. Therefore, OGP begins by packing nodes in depth-first order to the point where the relevant data begins, and then uses the standard packing algorithm. We implemented this functionality by assuming that the user focus is represented by the node that surrounds the volume of the model that is of interest to the user. This node is then considered as the root of the subtree that should be sent.

5. EVALUATION

In evaluating OGP, we found it useful to consider data-specific properties of the protocol, namely, node size and branching factor. We also evaluated the performance of the protocol in terms of throughput and fairness to TCP.
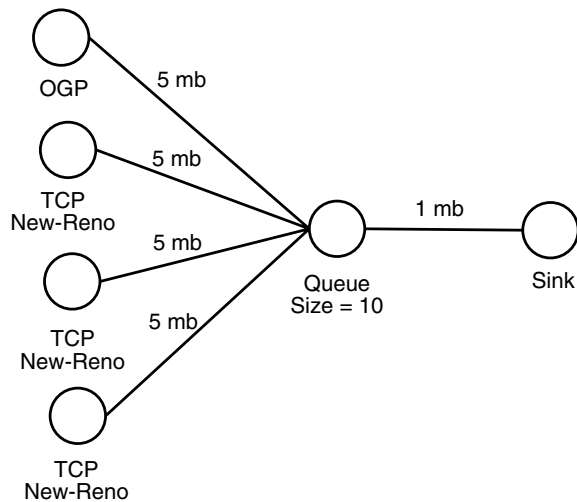
Figure 4: Simulation Set-Up

Figure 5: Simulation Results: queue size = 10, bottleneck = 1mb

	Packets Received	Packets Dropped
OGP	3898	135
TCP New-Reno 1	2871	175
TCP New-Reno 2	2866	175
TCP New-Reno 3	2857	175

Table 1: Simulation Results: queue size = 10, runtime = 100s, bottleneck = 1mb

5.1 Simulation Set-Up

In the simulations we used the ns2 [5] network simulator. The simulation layout we used is depicted in Figure 4. For the simulation runs, we used a node size of 200bytes and a branching factor of 4. The bottleneck queue size was varied between 5 and 40 packets. For the simulation runs, we had 3 TCP New-Reno flows and one OGP flow. The link speeds are given in Figure 4.

5.2 Data-Specific Evaluation

We first evaluated the effect of node size on OGP. As mentioned in Section 4.1, as node sizes decrease and more nodes are included in each packet, a breadth-first packing algorithm will encounter more occurrences of packets being sent that cannot be rendered. TCP however does not have this problem as it guarantees in-order, fully-reliable transport. We found however, that smaller node sizes merely increase the number of nodes that may be used as next sub-tree roots for each packet that is acknowledged. Therefore small node sizes have no real effect on OGP. We notice similar results from larger branching factors. We do not show these results here as they were all roughly the same.

In our simulations, we found that OGP always achieved 100% goodput, meaning that all packets received can be rendered. TCP also achieves 100% goodput due to its full reliability. One interesting point to notice, however, is that the packets sent by OGP can be immediately rendered, there is never a need to wait for retransmissions or re-ordering.

5.3 Performance Evaluation

The two main results that we wanted to evaluate were TCP fairness and good throughput. Our simulations show that these goals have been met. As the length of the simulations as well as the queue size on the bottleneck link were varied, we found that OGP
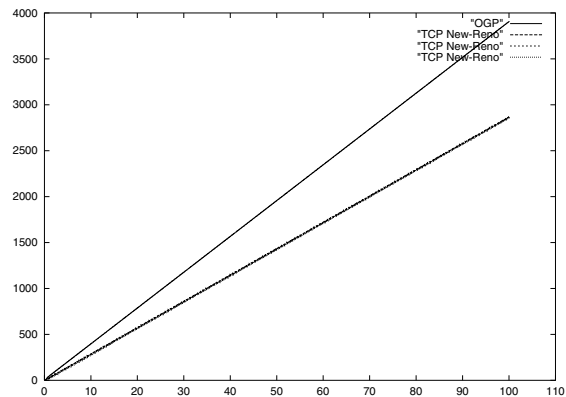
achieved very similar results to itself. The results for simulations with a queue size of 10 are displayed in Table 1 and Figure 5.

OGP reacts quicker to congestion, and therefore has fewer packets in flight during a congestion period. This is due to the fact that OGP detects losses not by the number of duplicate acknowledgments but instead by the difference between the current acknowledgment and the expected acknowledgment, i.e. gap detection. This rapid detection of losses allows OGP to cut its congestion window more rapidly than TCP. OGP, however, gets more throughput due to the fact that it does not worry about re-ordered packets and it does not wait to re-send lost packets before opening its flow control window. TCP New-Reno and OGP did reach an equilibrium point where all flows were getting bandwidth. OGP achieves approximately 26% better throughput however, due to the mechanisms already mentioned.

6. CONCLUSION

Given the greater processing power and better graphics cards in desktop computers, it is now possible for very complex three-dimensional models to be used in everyday applications. This leads to the desire to send such models across the network. Due to the potentially large size of the models, it is necessary to find a way to stream them efficiently over the network. In order to do this, the properties of the data structures used to store three-dimensional models must be exploited. The tree that is used in three-dimensional models has the property of being partially ordered, i.e. not completely linear. This led us to analyze the problems of having too narrow a view of the transport service space. Once we see the space as expanded, we can understand the problems caused by having too much reliability provided by the network. We therefore show which parameters must be considered in designing an efficient protocol for the transmission of on-demand three-dimensional models. We discussed the approach to linearize the non-linear partially ordered tree. We then exploit the approaches to design the On-Demand Graphic Transport Protocol (OGP). OGP is shown to gain performance benefits over TCP, making it clear that the inherent properties of the three-dimensional modeling data structures should be used to develop efficient transport mechanisms.

in the protocol. We would also like to thank the entire Mobius Group for their continued help and support.

7. REFERENCES

[1] M. Allman, V. Paxson, and W. Stevens. Tcp congestion control. Request for Comments (Standards Track) RFC 2581, Internet Engineering Task Force, April 1999.

[2] P. Amer, P. Conrad, E. Golden, S. Iren, and A. Caro. Partially-ordered, partially-reliable transport service for multimedia applications. In *Telecommunications / Information Distribution Research Program Annual Conference*, 1997.

[3] P. D. Amer, C. Chassot, T. J. Connolly, M. Diaz, and P. Conrad. Partial-order transport service for multimedia and other applications. *IEEE/ ACM Transactions on Networking*, 2(5):440–456, 1994.

[4] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In *Proceedings of the SIGCOMM '90 Symposium*, pages 200–208. 1990.

[5] K. Fall and K. Varadhan. Ns notes and documentation. LBNL, August 1998.

[6] I. S. O. (ISO). Information technology – generic coding of moving pictures and associated audio information. International Standard ISO/DIS 13818, ISO, 1995.

[7] R. Kravets, K. L. Calvert, P. Krishnan, and K. Schwan. Adaptive variation of reliability. In *the Seventh IFIP Conference on High Performance Networking (HPN'97)*. 1997.

[8] R. Marasli, P. Amer, and P. Conrad. Optimizing partially ordered transport services for multimedia applications. *Multimedia Modeling: Towards the Information Superhighway*, 1996.

[9] R. Marasli, P. D. Amer, and P. Conrad. Retransmission-based partially reliable services: An analytical model. In *Proceedings of IEEE INFOCOM 96*. 1996.

[10] C. Papadopoulos and G. Parulkar. Retransmission-based error control for continuous media applications. In *The 6th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV) '96*. 1996.

[11] J. Postel. Transmission control protocol. Request for Comments RFC 761, Internet Engineering Task Force, January 1980.

[12] J. Postel. User datagram protocol. Request for Comments RFC 768, Internet Engineering Task Force, August 1980.

[13] S. Rusinkiewicz and M. Levoy. Streaming qsplat: A viewer for networked visualization of large, dense models.

[14] S. Rusinkiewicz and M. Levoy. Qsplat: a multiresolution point rendering system for large meshes. SIGGRAPH, 2000.

[15] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. Rtp: A transport protocol for real-time applications. Request for Comments (Standards Track) RFC 1889, Internet Engineering Task Force, January 1996.

[16] C. J. Turner and L. L. Peterson. Image transfer: An end-to-end design. In *Proceedings of the SIGCOMM '92 Symposium*, pages 258–268, 1992.