Wireless video telephony performance enhancements: Towards applied network engineering approach[1]

Justin Madigan
Rochester Institute of Technology
justin.madigan@gmail.com

Fei Hu
Rochester Institute of Technology
fxheec@rit.edu

ABSTRACT

The aim of this network engineering research is to improve video transmission performance (such as end-to-end delay) in video telephony applications in cellular network scenarios. There exists a definite need for quality-of-service (QoS) and guaranteed throughput when using streaming media. However, in wireless environments where resources are precious, this is often not possible, especially with a large user base that is communicating simultaneously. Not only does traffic load pose a problem, but also the inherent instability of the radio links makes the task considerably more difficult. This presents the need for a new method for undertaking the task of wireless streaming media in resource-constrained cellular environments. Introduced and studied here is the effectiveness of a new transport-layer method based on UDP-lite, which is shown to significantly improve the performance of streaming media over noisy wireless networks in the Universal Mobile Telecommunications System (UMTS) cellular environment.

INDEX TERMS

Streaming video, UMTS, Video telephony, Wireless networks

I. INTRODUCTION

The field of streaming media has become increasingly important and relevant in the past twenty years. With a growing market push for on-demand media, the foundational substructure of current cellular networks must incorporate streaming media functionality. Streaming media is the term used specifically for media that is immediately consumed upon deliverance to its intended party [1]. This is defined in contrast to traditional media, which is consumed after a complete download of the media is finished. With the advent of streaming media, the possibilities for instant delivery of desired content as well as for live telecasting to mobile devices takes hold. With these exciting ideas comes a new market for streaming media and, consequently, a challenge for the existing protocols and substructures underlying the communications media of the current era. Modern cell phones, such as the Motorola Razr v3 [2], can play MP3s, view live and recorded videos, take pictures with a built-in high-resolution digital camera, surf the internet, play games, serve as a personal digital assistant, and more. With the functionality of these devices expanding so rapidly, the underlying network structure must be able to support Quality-of-Service (QoS) requirements of live media.

The main difficulty with streaming media is the immediate consumption of the data. This means that a reliable throughput between receiver and transmitter must be attained in order for the video to appear smooth and correct to the end user. This is a difficult to maintain, especially over long distances or in cases of high noise and/or lossy environments such as wireless networks. Wireless networks are inherently unstable and prone to data loss, corruption, and other undesirable issues. To avoid this, or at least mitigate it to an acceptable extent, the current methodology behind streaming media is to merely avoid the issue entirely by a process known as *buffering*. Buffering is the process by which a receiver buffers a certain amount of streaming video before showing it to the end user. By doing this, the receiver

1 This research has been supported by Sprint Inc. 2006 video-telephone project.

has a "backup" store of video which it can show to the user if the stream is somehow interrupted, packets are lost, or corruption is seen in the incoming video. This method works relatively well; however, it often requires long delays in order to buffer enough video to be an acceptable method. Additionally, it is only a "cover-up" method which, in the event of packet corruption, causes the user to still experience unwanted delays in his or her streaming video. This is the main cause of user frustration in streaming media, especially in cell phones. It also makes the possibility of live teleconferencing significantly more difficult, since by delaying the packets due to buffering, the "live" aspect of a telecast is lost.

Current protocols such as TCP [3] have robust implementations that are resistant to packet loss and packet error. However, their overhead has shown to be simply unbearable in the case of streaming media, especially in resource-constrained environments and wireless environments, as discussed in [4]. Even UDP, the simplest transport protocol [5], is not entirely ideal for streaming video. This will be discussed in depth later on, but there are certain fundamental issues underlying the typically seen protocol stack. The main purpose and drive of the protocol stack as it stands is to deliver content reliably and correctly at the expense of bandwidth and delays. Although the balance is not excessively tipped, and indeed mediation can occur, the main focus is reliable content delivery at whatever cost is necessary. The opposite approach is required for streaming media. Users are far more content with a slightly distorted image in a video that lasts for milliseconds than with a stoppage of video while rebuffering can occur every ten seconds or so.

Clearly, a new approach is needed. The ideal approach for use in a resource-constrained environment would improve performance without requiring significant additional overhead. It would also strive to be as backwards-compatible as possible as to not disturb the pre-existing architecture which is being used for other purposes as well (as is the case in cell phones, etc.). This paper explores the possibility of a new *transport-layer* protocol which significantly improves performance in wireless resource-constrained networks without significant impact to overhead.

The rest of this paper will be organized as follows: Section 2 briefly introduces UMTS that serves as our experimental cellular network platform. Section 3 introduces the basic MPEG video characteristics. A comprehensive live video performance analysis under different UMTS conditions will be discussed in Section 4. In Section 5, we will propose our improved UDP protocol to enhance video telephony performance. Sections 6 and 7 provide our experimental results. Section 8 concludes the entire paper.

II. UMTS: AN ADVANCED CELLULAR NETWORK

Currently, the European and Asian cellular telephone markets are saturated with a technology known as the Global System for Mobile Communications, or GSM. GSM is capable of 9.6 kbps of user data rate for services such as videotext, facsimile, and teletext [6]. This kind of data transfer rate is unacceptable for streaming video, which requires significantly more in terms of available bandwidth. With GSM technology nearing the end of its lifecycle, a newer technology has come to replace the aging GSM, and this technology is called the Universal Mobile Telecommunications System, or UMTS. UMTS is an exciting new technology, capable of 1920 kbps, a data rate which far outweighs the GSM capability [7]. Although users can currently expect rates closer to 384 kbps, Japan is implementing a 3 Mbps upgrade to its currently active UMTS networks [8]. Clearly, this newly available bandwidth opens the door for a myriad of new uses for the cellular telephone network in Europe and Asia.

Additionally, UMTS is touted as being globally capable and intends to replace the existing dichotomy of CDMA-based services in the US and Canada and GSM-based services in Europe and Asia. This will allow global communication as well as the ability for users to take their cell phones to virtually any country and still maintain functionality.

UMTS also is permanently melded with the internet (called Wireless Internet) both in form and design. This makes UMTS a valuable asset and important in streaming media, since it easily opens the possibility for streaming media straight from the internet to be viewable on a cellular device. Since the upcoming UMTS system is so noticeably apt for streaming content, it is used as the platform for evaluating the performance of the proposed protocol herein.

III. MPEG-4: A POPULAR VIDEO FORMAT

MPEG-4 is a popular format for many handheld devices, such as the Apple video iPod [9] and many popular cellular phones (such as the Motorola Razr v3 [2]). Each individual picture in an MPEG-4 video

is referred to as a *frame*. Each frame is made up of a series of 8x8 pixel squares called *blocks*. A row of blocks is known as a group of blocks, or *GOB*. A GOB also contains a header in MPEG-4. (Although this is optional, it is typically used if any errors in video transmission are expected.) The header can aid in resynchronization if a section of MPEG data is corrupt or unusable. Because MPEG-4 videos are encoded with a variable bit-rate, if a piece of a frame is corrupt, the frame would ordinarily be unusable (see illustration in Figure 1).

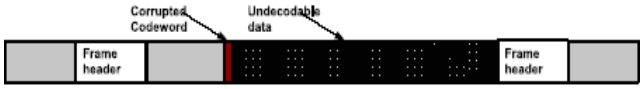


Figure 1. Case 1: A piece of frame data is corrupt (image from [10])

However, to combat this, a known data pattern is inserted every so often as a "marker" to allow resynchronization if errors occur. This allows the decoder to "pick up" the frame as soon as a bit marker is identified. This idea is illustrated in Figure 2.

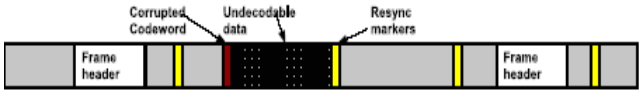


Figure 2. Case 2: Corrupt data with Resync Markers [10]

Although this method leaves undecodable data, it is clearly preferable to the option of simply losing an entire frame. This method is actually improved upon in the MPEG-4 specification when reversible variable length encoding is used by allowing *backward* decoding. That means when a resynchronization marker is found, the frame can be regenerated backwards starting from the first good resynchronization marker. This process is illustrated in Figure 3.

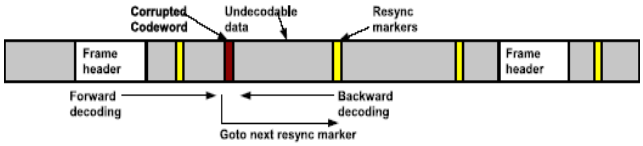


Figure 3. Case 3: Corrupt data with bidirectional resynchronization markers and reversible variable length encoding [10]

Clearly, from the diagrams shown, frame loss is avoidable even with corrupt data. This means that when faced with the decision to accept a corrupt packet or reject it entirely due to unknown corruptions within the packet, the advantageous choice is to accept the packet and allow the built-in error correction mechanisms to take hold. It should be duly noted that this is not the only error-correcting method available to MPEG-4, and other methods exist to layer atop this method which make the MPEG-4 encoding even more robust. Nonetheless, the point remains entirely valid that clearly *packets are better off used than discarded in MPEG-4, even with corruption.* Our improved UDP scheme (to be discussed in Section 5) utilizes this principle.

IV. UMTS VULNERABILITY AND STABILITY ANALYSIS

Ideally, any network would exhibit the same performance under any level of load on the system. In reality this is not possible, since computing resources (such as CPU capability) are always limited. Thus attention must be paid to maximizing the performance seen by the video telephony users at all times even under times of heavy video strain on the physical wireless system. To best accomplish this, the system cannot be constrained by any one system component—that is, there should be no "bottleneck" in the system.

The approach we take here is to attempt to identify core components of the UMTS system that may cause a bottleneck in video telephony performance. If such performance-critical components could be identified, a best-effort approach could be taken to ensure that their utilization does not exceed some "critical mass"—that is, a utilization level beyond which network performance suffers dramatically. This could significantly improve performance of the wireless networks in times of heavy video load and could be used with great success in high-traffic cellular network areas.

To correctly investigate any phenomenon or idea that is difficult to implement physically, a proper simulation is required. Without a robust and objective simulation to compare other results against, it is impossible to gauge the performance of one algorithm or idea against another. The industry-standard network simulation tool, OPNET Modeler [16], is an ideal choice for the investigation of UMTS traffic operation and its parameters for many reasons.

The parameters in this case were chosen based on items that are likely to be overutilized during heavy wireless network congestion. Resultantly, CPU usage, IP packet buffers, and background system utilization are of particular importance. The

Table I. Specific parameters and the explored permutations

PARAMETER	DEFAULT	NEW VALUES	RATIONALE
Background System Utilization	0%	10%, 50%, 90%	Test the effects of modifying background utilization to determine if the modification of the usage of any one component will adversely affect system performance more than the others.
Number of CPUs in UMTS base-staion	1	20, 32	Determine if multiprocessing capability in the UMTS base-station would significantly improve performance in heavy traffic. Expected result would be improved performance of the multiprocessor system due to its ability to handle multiple tasks / connections simultaneously.
IP Buffer Space in both cell phone and base-station	16 MB	64 MB, 128 MB, 256 MB	Ascertain the effects of additional buffer space for incoming packets. Expected result would be fewer dropped packets and hence better performance.
Datagram Forwarding Efficiency Protocol	Disabled	Enabled	Determine if using a datagram forwarding efficiency algorithm provided in OPNET will improve performance.
Number of CPUs *and* IP buffer space	1 CPU, 16 MB buffer space	32 CPUs, 256 MB buffer space	Determine the effects of having a multiprocessor system with a large buffer on performance in heavy traffic. Expected result would be a performance boost similar to that found when increasing just one of the attributes. This will ascertain if the effect of both is additive or the same as one or the other alone.
Datagram Forwarding Buffer	10 packets	1000 packets	Test the effect of increasing the base-station's ability to forward packets, potentially resulting in fewer dropped packets.

variables and their altered values may be seen in Table 1 along with a summarized rationale.

The simulation setup (as shown in Figure 4) was a simple two-node UMTS system exchanging heavy amounts of normal resolution video traffic. This encompasses the same type of traffic seen under a many-node scenario. (In this case, however, two nodes form multiple connections and exchange data rather than multiple nodes; however, the effect seen from the RNC [i.e. UMTS base-station control center] and similar components is identical.)

The first explored topic was the *background system utilization,* as per the aforementioned table. The purpose here was to ascertain whether or not the performance degradation seen from *background system utilization* on any particular UMTS device would be greater or less when compared with other devices under a similar strain. In other words, this was a simple way to root out "bottlenecks" in the system by letting OPNET simulate background system utilization. The obtained graphs from the simulation may be viewed in Figure 5 (a)~(c).

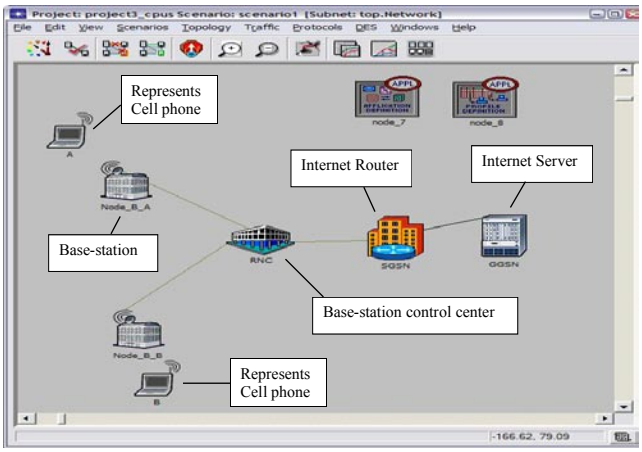


Figure 4. OPNET simulation setup for parametric testing

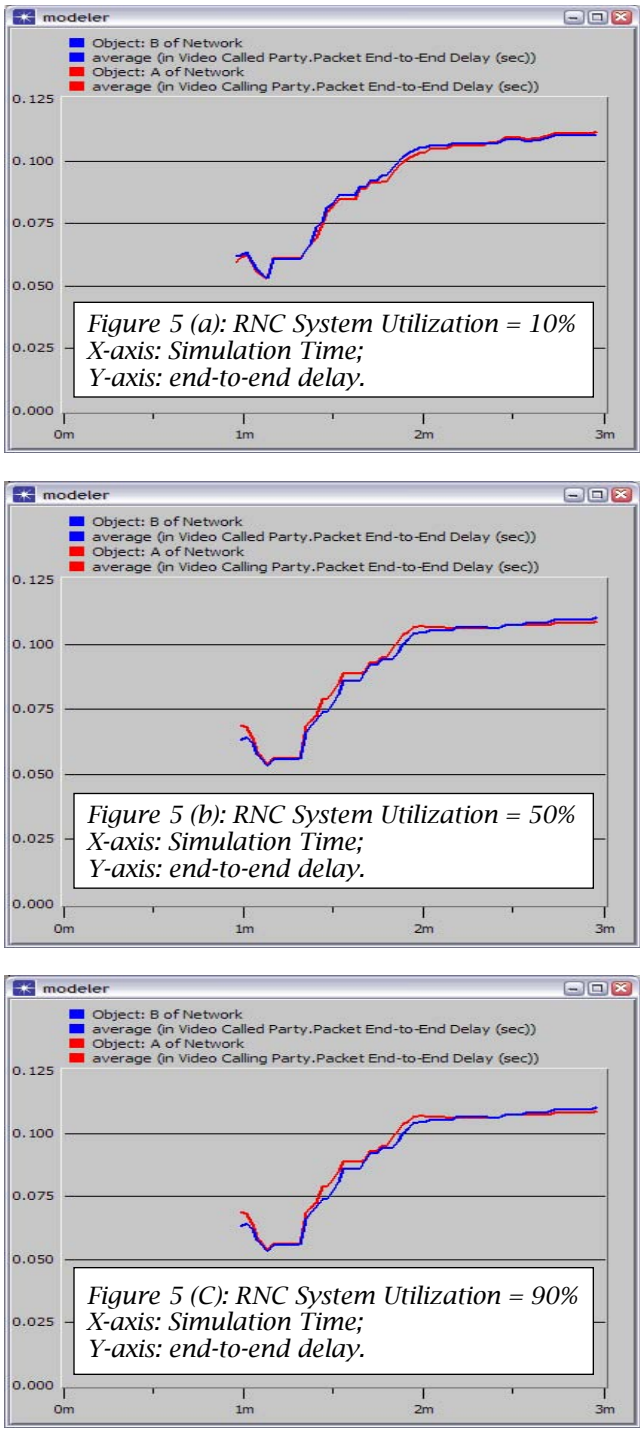
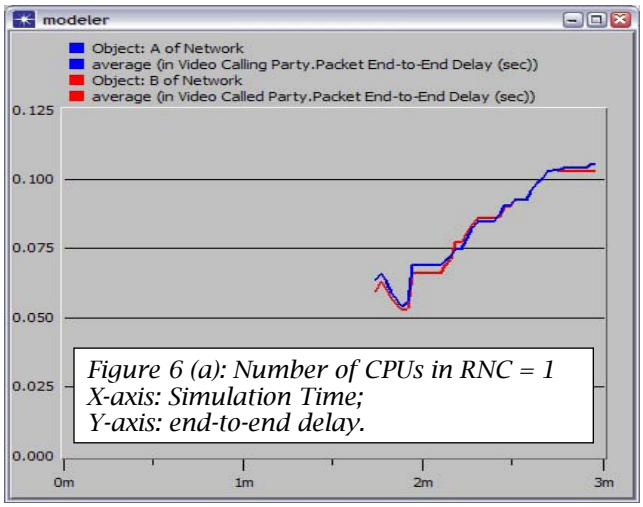
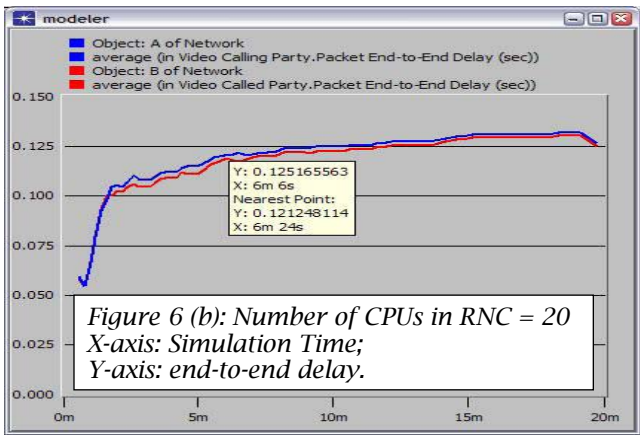


Figure 5. End-to-end delay under system utilization of 10%, 50%, and 90% respectively

These results clearly indicate, when looking top to bottom in Figure 5, the occurrence of a single cutoff value – specifically, performance seems relatively stable outside the range of 10% system utilization. In other words, after about 10% of system utilization, the heavy system load no longer sees performance degradation from the further utilization. This indicates that the system can adequately handle the traffic with only 90% of the system resources and will not see further degradation given the system resources until after 90% utilization. Thus, the system utilization itself is not enough to constitute a "bottleneck" in the UMTS system; *however, an important lesson is revealed through this experiment: system utilization affects performance even at low usage levels.* Although these performance impacts are not drastic, they do exist and, clearly, this indicates that in even heavier traffic loads the performance impact would be exacerbated. System utilization should be kept at a minimum wherever possible, although with high traffic areas this may be an unrealistic expectation.

The next varied parameter is the *number of CPUs* in the RNC. These results, shown graphically in Figure 6, indicate the differences. According to the data, an optimal situation is found in the single CPU scenario. The 20-CPU scenario sees actual performance degradation, and the 32-CPU scenario seems to recover slightly. This is most likely because the level of exploitable parallelism in the given simulation scenario is reached in the single CPU scenario. In other words, there is very little parallelism to exploit herein. Perhaps in a truly multi-client scenario, one in which there were many simultaneous connections coming from autonomous clients in varying locations with different demands, this scenario might play out differently under the various numbers of available computing units. Nonetheless, at this time it seems the single core RNC performs the best in the given scenario.





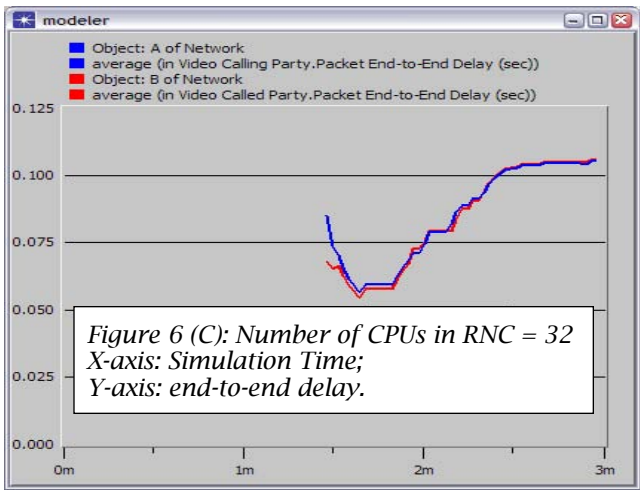


Figure 6. RNC multiple CPU simulation results (a) 1 CPU (b) 20 CPUs (c) 32 CPUs

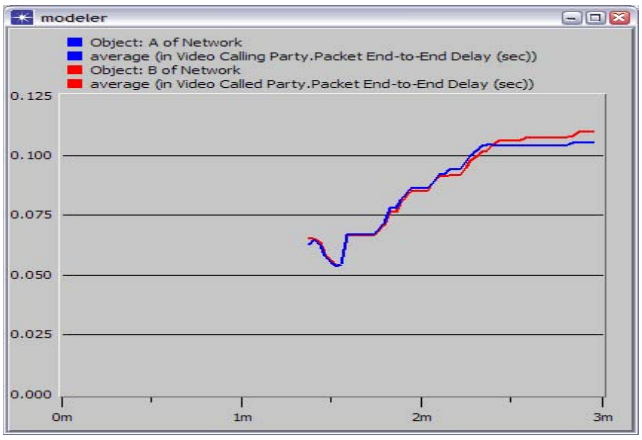


Figure 7. Simulation performance results with all components using 32 CPUs

The next step was to outfit all devices with the maximum possible amount of CPU power, that is, all communication components (including cell phones, base-station, and control center) were given 32 CPUs, and the results are displayed above (Figure 7).

In this scenario, the clear result is virtually identical to the single CPU scenario. Thus, it is very clear that at no level is there exploitable parallelism for the explored case. Hence in situations of single-stream video traffic between communicating nodes, it does not appear that there are significant, if any, gains from multiple processing at any level, since there does not appear to be any exploitable parallelism.

If packets are coming too quickly at any component that is required to forward them, it is conceivable that such a component could become clogged and be forced to drop packets accordingly. Therefore, it stands to reason that increasing the *buffer size* could result in better performance due to fewer dropped packets. This would translate to the final end-to-end delay, which incorporates packet loss. However, the results in this case (see Figure 9) indicate no measurable gains whatsoever from the IP buffer increase. It would seem that in this case there is little forwarding to be done. Consequently, there is more buffer space but effectively no use for it, and hence no performance impact is seen. Nonetheless, it is entirely possible in a scenario where more forwarding were required that these results might be different. A more powerful scenario environment and processing elements would be required to properly test this phenomenon.

Perhaps it is not the buffer space that is an issue but the *forwarding protocol* that might be used.

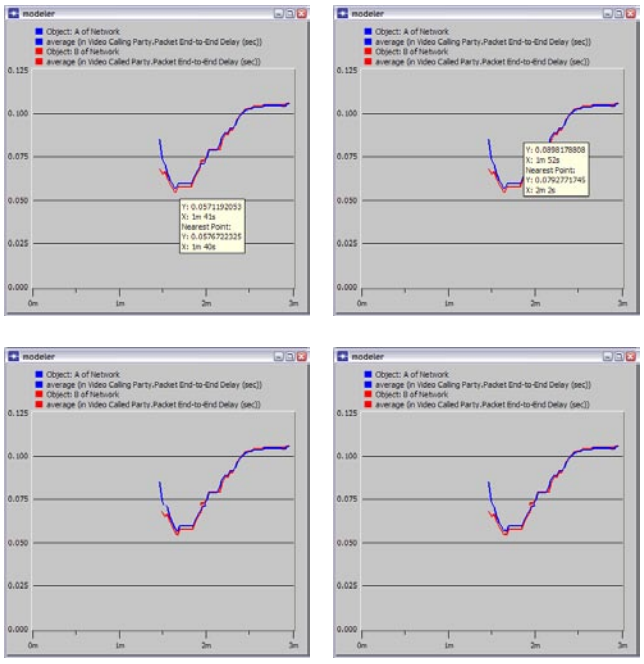


Figure 8. IP Buffer modification simulation results, top left to bottom right, 16 MB, 64 MB, 128 MB, 256 MB IP buffer memory

OPNET has a built-in *forwarding efficiency protocol* which can be used. Although no specific details are given in the OPNET documentation, the results supposedly can enhance forwarding performance in packet-switched networks. The results of implementing this protocol are shown below (see Figure 9). In this case we actually see degradation in performance, albeit a slight one. This protocol should not be used; however, we do find an interesting phenomenon here—performance seems to be relatively hinged on the forwarding protocol. Further exploration into this phenomenon would be useful since such sensitivity was found in this simple test. There is the possibility that the forwarding protocol could hold significant ability to improve performance and yield the desired "bottleneck" this research is searching for.

The next investigation centers upon the concept of the combined impact of two previously explored elements: *the multiple CPU option coupled with the IP buffer option.* Perhaps a source of performance degradation in the multiple CPU scenarios was not simply nonexistent parallelism. Instead, perhaps the increased speed of the 32 CPU processing was producing packets too fast for the buffers to handle and thus packets were being dropped. In such an event, a bottleneck would have been in-

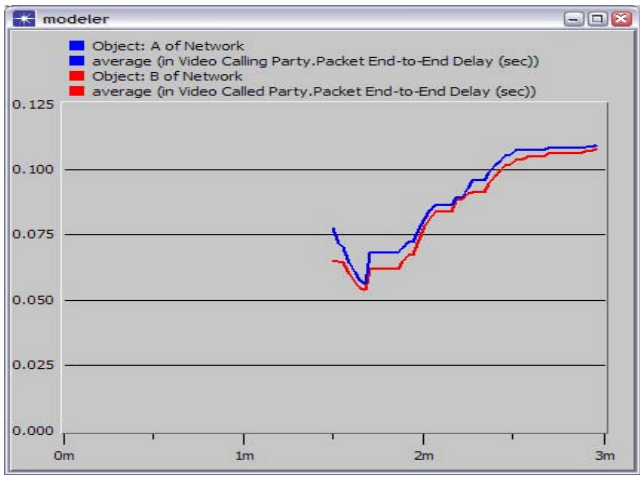


Figure 9. Forwarding efficiency protocol simulation results

advertently created. To test this, giving the entire scenario a complement of 32 CPUs and 256 MB of IP buffer space should show performance improvement if such a bottleneck had been created. The results in Figure 10 show an actual slight performance improvement over the single CPU, 16 MB IP buffer space scenario. However, this improvement is not enough to warrant a bottleneck discovery, although further research into this phenomenon may yield intriguing results. The probable cause of the performance improvement here is simply that the faster processing ability, coupled with the buffer space improvements, have removed a few packet drops somewhere in the simulation. The results are not drastic enough to be convincing.

The final phenomenon investigated is the potential for a *datagram forwarding buffer* of 1000 packets as opposed to the default value of 10. *Instead of buffering at the IP level, this would buffer at the datagram, or transport layer.* In this case the idea is presented that at the UDP layer, perhaps corruption or packet loss is being seen, and thus by increasing the available buffer space for UDP packets, this could be partially alleviated. The results are given in Figure 11 and show a very good matching between the sender and receiver end-to-end delay. There is also a slight performance improvement evident in the graph. This shows that perhaps at the UDP layer there are some exploitable performance enhancement techniques.

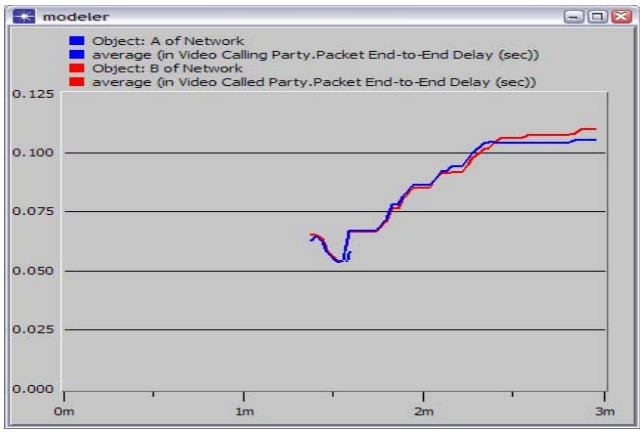


Figure 10. Simulation results of all components with 32 CPU and 256 MB IP buffer

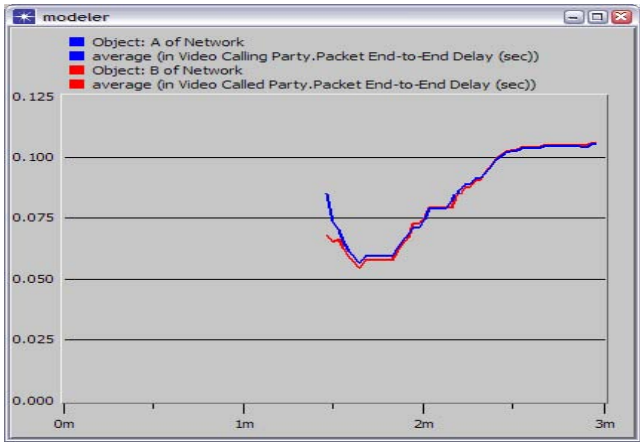


Figure 11. Simulation results from datagram forwarding buffer at 1000 packets

In conclusion, a comprehensive set of parametric tests were performed on the UMTS network under a video traffic load. Several conceptual bottlenecks were explored, including the concept of a CPU-bound bottleneck, memory-bound bottleneck, and a combination thereof. No singular bottleneck was found for the case that was explored, although the possibility of bottlenecks in different scenarios was suggested during various parts of the simulations.

The most significant result was the UDP datagram forwarding protocol enhancements which yielded a slight but noticeable change; besides a small performance improvement, an end-to-end delay match was found between sender and receiver. This means that there is no packet loss or other phenomenon hindering one side more than the other (sender or receiver). The following section will explore a potential way to exploit the ramifications of this result toward better streaming video performance in UMTS networks.

V. IMPROVED TRANSPORT PROTOCOL DESIGN

Many different methods have been attempted to improve performance of videos across congestion-prone networks. Among these methods is an approach known as UDP Lite [11], a modified version of which is the focus of this chapter. UDP lite is a conceptually straightforward method of improving performance through the reduction of packet loss. The concept behind UDP lite is the specification of data as being partially free of corruption. If such a "corruption threshold" is met, the packet is accepted regardless of the fact that some corruption may indeed exist in the packet payload. This method has been implemented with success toward cellular video, with significant performance improvements being seen in [12].

In many different algorithms for the resource-constrained transmission of voice and data, packet loss is difficult to recover from. For example, in many video compression algorithms, "key frames" are sent periodically to update the exact video frame information, but generally only changes in the video from frame to frame are transmitted. This means that only some metadata about the current video status is being sent from end to end rather than entire frames. This can be envisioned as a simple set of sequential numbers, each depending on the previous number. If a number in the sequence is lost, the sequence becomes unintelligible to the receiver since each piece depends on the correctness of the previous number. It is similar in video – a lost packet may manifest as visual corruption or loss of synchronization. In this sense, it is important to minimize packet loss whenever and however possible in video transmission schemes.

The figure below (Figure 12) shows the UDP and IP pseudo headers [13] as seen in RFC 768. In order to accomplish the stated goal of improved performance in streaming video without significantly impacting the amount of overhead required, a change must be made in the existing header structure. To maintain compatibility with other applications, this change should not alter the underlying substructure of the header format; rather it should be a change which uses the existing fields in a perhaps more clever or optimized way in regard to real-time streaming video.

The change suggested here is similar to that of UDP lite: change the length field in the IP header to a "checksum coverage" field and alter the checksum to be a partial checksum (see Figure 13). It is true according to [11] that real-time video and audio protocols are more sensitive to packet loss than to partial packet corruption. Furthermore, many protocols contain header information or important details at the front of the packet that are important. Thus it stands to reason that, if the first few bytes of the packet might be deemed trustworthy insofar as the data it contains, the packet itself might be salvaged by the error correction and fault tolerance built into the higher layers (namely, the video or audio decompression scheme in use in the application layer). The proposed scheme, then, would provide a checksum coverage field indicating how much of the packet (beginning from the start of the UDP layer header) is covered by the indicated checksum. The checksum, then, is replaced by this partial checksum. By specifying the packet information as being partially error-free, a corrupt packet would be allowed to proceed to the higher layers, and the higher level fault tolerance and error correction capabilities would be relied upon to mend the damage in such a way that would be transparent to the end user.

IP pseudo-header	Source address		
	Destination address		
	Zero	Proto	UDP length
UDP header	Source port		Destination port
	Length		Checksum

Figure 12. The UDP/IP pseudo-header from RFC 768 [13]

Pseudo header	Source address		
	Destination address		
	Zero	Proto	UDP Length
UDP Lite header	Source port		Destination port
	Coverage		Checksum

Figure 13. UDP lite headers from [11], closely related to the proposed approach (field names are the same)

To understand this approach, a flowchart is shown in Figure 14 with the specific process highlighted. This method is a strictly transport-layer protocol, thus the transport layer methodology is described herein. Other protocols are certainly usable above or below this one. The UDP-Lite Pseudo header block diagram is also given, as the fields are very similar to the proposed approach.

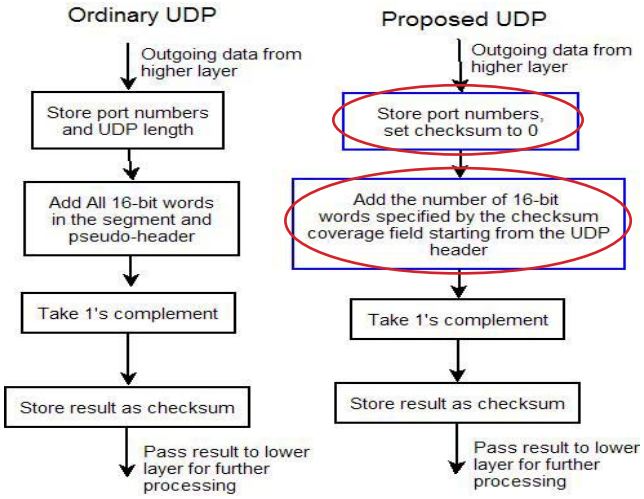


Figure 14. Ordinary UDP versus proposed UDP, differences highlighted

The unique aspects of this new protocol will now be outlined and discussed. For outgoing information, first the port numbers are stored as they will be used. This includes the source and destination port, just as in regular UDP. Next the checksum is set to 0. This is a requirement of the protocol and prevents the checksum from being erroneously included in any checksum calculations. The checksum coverage field is used to determine the number of bytes of the UDP packet to checksum. The checksum coverage field specifies the number of bytes, beginning with the first byte of the UDP header. A legal nonzero coverage value *must* be greater than 7, since the UDP header is 8 bytes and the checksum *must* cover the header. This is done to ensure the proper header values. Essentially, this signifies that the header itself is considered universally vital to operation regardless of the specific application. A zero coverage value indicates the entire packet is included in the checksum, making the approach identical to ordinary UDP in this case. The checksum is computed through 16-bit additions of each 16-bit word until the checksum is complete, pursuant to the aforementioned rules. The one's complement of this sum

is taken, and the result of that operation is stored as the checksum. With this, the packet may be passed to a lower layer. For incoming data, the reverse process is done. The checksum is stored and the checksum field set to zero; the checksum is calculated using the specified rules and the result (without the one's complement) is added to the checksum. If the result is all ones (0xFFFF), then there are no errors in the section specified by the checksum coverage field, and the data may be passed on to the higher layer for further processing.

The difference between the protocols is subtle, but its impacts on performance can be profound. As discussed earlier, specifying data as partially insensitive to errors is often a better choice than specifying data as wholly insensitive to errors, as UDP currently does. This change allows higher layers to use their error detecting and correcting protocols while not entirely discarding the lower layer of protection provided at the UDP layer. In the standard UDP lite, which has slight differences compared to the proposed model, performance has increased significantly in real-time audio-visual data transmission with sources reporting improvements in end-to-end delay of 26% and 50% less packet loss than traditional UDP [14]. Hence. although the differences between the protocols may seem trivial, its impacts are certainly far from it.

VI. OPNET EXPERIMENTS SETUP

To properly test the ideas of Section IV, the network simulator OPNET [16] was chosen for its renowned reliability and accuracy in network simulation. The ideas exactly as previously specified were implemented in OPNET code.

Since OPNET's built-in video conferencing functionality is not sufficient to properly test an idea of this magnitude accurately, actual trace footage from an MPEG-4 video was used. The video trace used was a verbose trace from the movie *Jurassic Park* [15]. This should be an acceptable video since it has wildly varying colors, landscapes, and other interesting landmarks that would be seen in real streaming videos and would properly "stress" the MPEG-4 codec's ability to recognize shapes and patterns and compress them adequately.

Of course, the vital aspect here is the actual implementation of the code in OPNET. First the length is computed from the packet size. Next the checksum is computed based on the checksum length specified either by the user default (which is 20 bytes, including the 8-byte UDP header) or by a 0, which indicates checksumming the whole packet. The checksum length to actually compute is found by taking the previously determined value and subtracting 8 (bytes), which are the UDP header and *must* be part of the checksum. Next the checksum is computed and the one's complement is taken so that it can be passed to the next layer.

On the receiving end, the code is essentially identical save for the fact that it is entirely reversed. In the same way as in the transmitter, the checksum is computed, this time after the received checksum is saved. Note that any checksum coverage values of 1 to 7 are deemed invalid and the packet is immediately discarded. This applies also to packets with checksum coverage larger than the packet itself. Provided the packet passes these "sanity checks," after the checksum is computed it is added to the received checksum. If the result is all 0's with an overflow, then the data is verified up until the point that the checksum coverage specifies. At this point the data can be considered partially insensitive to errors up until the point of the checksum coverage. It is then accepted and sent to the next higher layer for processing (IP layer).

VII. SIMULATION AND RESULTS

Two main parameters were focused on during the simulation: *end-to-end (video transmission) delay* and *video jitter. End-to-end delay* is the amount of time seen from the sending of a packet initially and its correct reception by a receiver. *Video jitter* is the amount of delay and unwanted interruptions or "skipping" seen by the user in the video experience. A reduction in both is ideal. A simple two-node simulation was done with high noise to attempt to yield some corruption which could bring out the improvements offered by this new approach.

A high update interval (once every 100,000 events) was used to get a fine-grained view of the overall performance improvement of the new protocol. Numerous simulations ensure accuracy in the resulting data, although all simulations resulted in the same performance. The final results of the experiment can be seen in Figures 15 and 16.

Averaging the above data and comparing them yields an overall average improvement of 10% in end-to-end delay and 5% in jitter. The jitter improvement is difficult to see directly, because jitter

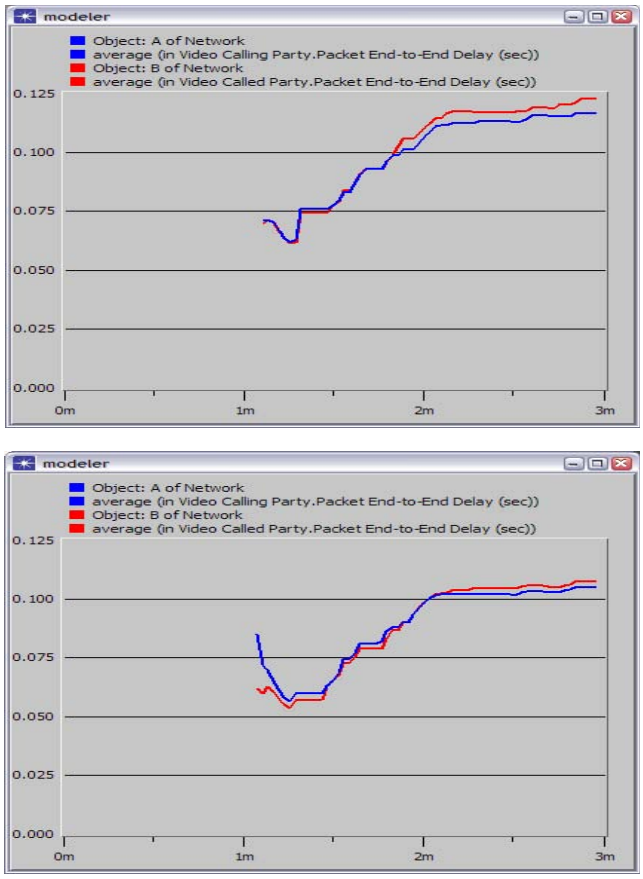


Figure 15. End-to-end delay –
(a) original UDP (b) proposed UDP

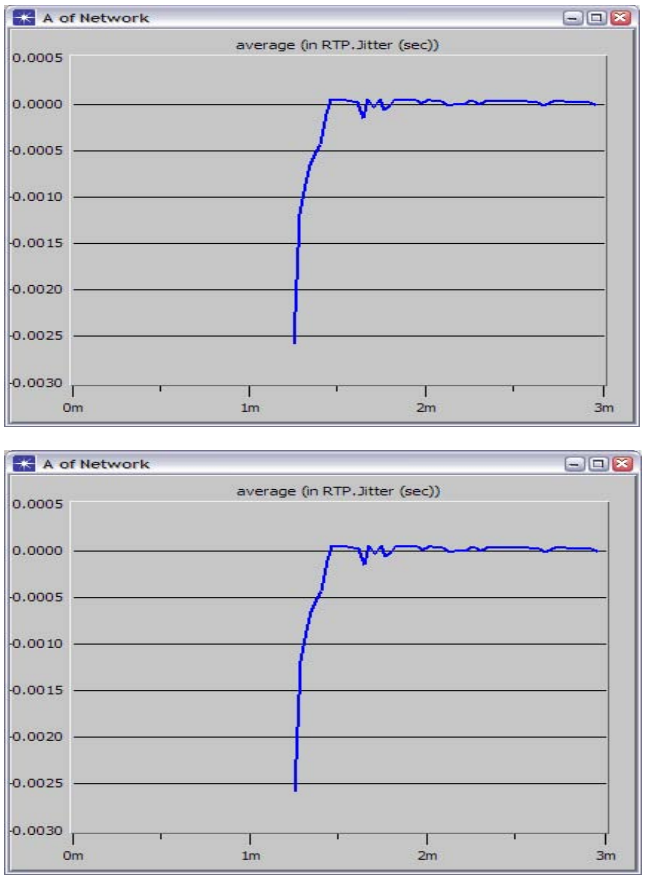


Figure 16. Video jitter–
(a) original UDP; (b) proposed UDP

is already low due to the frame buffering done by the receiver and the relatively low amount of traffic in the simulation. Unfortunately, a wide-scale simulation required significantly more resources than were available; however, this scale simulation is a good indication, especially in terms of end-to-end delay, that the new transport protocol is a success.

Thus the results of the simulation show that indeed there is an improvement in the performance of the proposed scheme over the traditional UDP approach. The purpose and stated intent of the new protocol is end-to-end delay reduction in high-noise environments, and this has been achieved with significant results even in a non-congested network. These results indicate end-to-end delay reductions averaging 10%. Additionally, since this baseline case shows improvement, further research with this protocol in high-noise, high-traffic scenarios is warranted. Unfortunately, that type of widespread simulation entails highly complex parametric calibration utilizing real-world scenarios and further research which is outside the scope of this paper. Nonetheless, with these impressive results highlighting the bare essential baseline case, there is much cause for interest in the further pursuit and exploration of this protocol.

VIII. CONCLUSIONS

Streaming media over wireless and cellular networks is an exciting and fascinating upcoming field with diverse challenges and fertile ground for new ideas. The boom of the telecommunications industry with regard to cellular telephones and a general shift from household to portable electronics has fueled a trend toward interactive portable multimedia devices. With this new technology comes the challenges of reliable content delivery, acceptable end-user experience, and maintenance of high video quality even under stressful network conditions. With the immense user base that cellular telephones already have, and the network conditions that already exist, providing an entirely new multimedia experience for the end user is an important and difficult new problem. This paper proposes a modification to the UDP transport layer which will provide improved end-to-

end delay and decreased jitter. Complex simulation results yielded a maximum 10% drop in end-to-end delay and a maximum 5% drop in jitter. The topic of streaming media is currently very active and an incredibly rich and diverse area of research. This new protocol constitutes an exciting and innovative new approach that should prove useful in the continual development of this amazing field.

IX. ACKNOWLEDGMENTS

This work has been supported by Sprint Inc. 2006 Video telephony project, Sensorcon 2005 Wireless Research Program, and Cisco Inc. URP (University Research Program) 2005 Grants. The research results presented here do not reflect Sprint, Sensorcon, or Cisco's opinions.

X. REFERENCES

[1] Media Services Multimedia Glossary, *Media Services Division of the Information Technology Department,* University of California Santa Cruz, 2005.

[2] Motorola Razr v3 User's Manual, *Motorola, Inc.* Available: http://www.motorola.com/mdirect/manuals/v3_manual9491A47O.pdf.

[3] J. B. Postel, editor. Transmission Control Protocol. Internet Request For Comments RFC 793, September 1981.

[4] J. Chung, M. Claypool, and Y. Zhu. Measurement of the Congestion Responsiveness of RealPlayer Streaming Video Over UDP. *In Packet Video Workshop* (PV), April 2003.

[5] J. B. Postel, editor. User Datagram Protocol. Internet Request For Comments RFC 768, August, 1980.

[6] Radio-Electronics, "A technical overview or tutorial of the GSM (Global System for Mobile Communications)," Cellular Telecoms. Available: http://www.radio-electronics.com/info/cellulartelecomms/gsm_technical/gsm_introduction.php

[7] Wikipedia contributors, "Universal Mobile Telecommunications System," *Wikipedia, The Free Encyclopedia.* Available: http://en.wikipedia.org/w/index.php?title=Universal_Mobile_Telecommunications_System&oldid=61696538

[8] Ibid.

[9] Wikipedia contributors, "MPEG-4," Wikipedia, The Free Encyclopedi. Available: http://en.wikipedia.org/w/index.php?title=MPEG-4&oldid=61359434

[10] Department of Computer Science, University of Verona, "Error Resilience Tools in H.263, H.264, and MPEG-4", Available: http://www.di.univr.it/documenti/OccorrenzaIns/matdid/matdid981158.pdf

[11] L. Larzon, M. Degermark, and S. Pink. UDP lite for real time multimedia applications. Technical Report HPL-IRI-1999-001, HP Labroratories, April 1999.

[12] A. Singh. et al., "Performance evaluation of UDP Lite for cellular video." NOSSDAV'01, 2001.

[13] Postel, J., User Datagram Protocol, Internet Request for Comments RFC 768, August 1980.

[14] A. Singh, A. Konrad, and A. D. Joseph, "Performance Evaluation of UDP Lite for Cellular Video," Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), June 2001.

[15] Universal Pictures, Ambien Entertainment, *Jurassic Park,* released June 11, 1993.

[16] OPNET: a commercialized network engineering simulation tool. Available: http://www.opnet.com

ABOUT THE AUTHORS

JUSTIN MADIGAN is a Computer Engineer from the Kate Gleason College of Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA.

FEI HU received his first Ph.D. degree from the Shanghai Tongji University in 1999 and the second Ph.D. degree from Clarkson University in 2002. He is currently an assistant professor in the Computer Engineering Department at Rochester Institute of Technology, Rochester, NY. Dr. Hu is a full Sigma Xi member, a member of the IEEE, and an IEEE chapter officer. His research interests are in ad hoc sensor networks, 3G wireless and mobile networks, and network security.