International Journal of Computer Applications (0975 8887)
Volume 87 - No. 15, February 2014

Delivering Low Latency Video using TCP with Network Coding over Wireless Network

Gokul Bhat
University of Florida
Gainesville, FL 32611

Janise McNair
University of Florida
Gainesville, FL 32611

ABSTRACT

Network coding (NC) techniques for lossy wireless networks have been used for fault-tolerant and timely delivery of streaming video data. Recent research on inter-session NC notwithstanding, reliable transmission of high quality media over wireless networks continues to be a challenge. The effects of traffic and network dynamics on coding block size and thereby on latency added at playback were studied, and the interaction of congestion control on the coding technique to remedy this were examined. Simulations show an inherent latency in video playback when TCP with random linear NC is employed as the receiver needs to wait for a certain number of packets to arrive before they are decoded. This paper presents an adaptive NC algorithm based on the nature of video streaming traffic and the available transmission opportunity to improve streaming performance with lower latency and reduced jitter in case of streaming TCP traffic. This algorithm is constructed under the constraint of available transmission opportunities and arriving traffic. The simulation results corroborate that the proposed adaptive NC algorithm reduces observed latency at playback by more than 90% over traditional TCP and more than 60% over simple NC technique. Additionally, the observed jitter reduced by 70% over only TCP and about 60% over fixed bucket size NC. To demonstrate the utility of our approach, the proposed algorithm was compared with TCP's performance for a real-world video trace. Results from this experiment indicated an 80% reduction in end-to-end latency.

General Terms:

Reliable multimedia streaming, Low latency media delivery, Network coding

Keywords:

Bucket size, congestion control, finite field, intra-session network coding, latency, linear network coding, ns-3, streaming media, video aware network coding

1. INTRODUCTION

Demand for multimedia streaming has been rising exponentially with the rise in on-demand and streaming video applications like NetFlix, Hulu and Youtube as well as multi-party gaming. Although, several techniques have been applied to improve the performance of streaming multimedia applications [1], [2], adding the dynamic nature of the wireless channel introduces challenges such as strict timeout deadlines and limitations to bandwidth availability as well as delay and jitter [3].
Typically, widely used services like Youtube use HTTP with TCP to deliver streaming media [4]. TCP significantly degrades the performance for interactive video applications like video conferencing since the three-way handshake and congestion control would cause undesirable delay [5]. Yet, most of the internet streaming traffic uses HTTP and HTTP traffic uses TCP due to its reliability. Despite TCP being ideally suited to wired networks and presenting several challenges to its use in wireless networks, random packet erasures being one of them, it remains one of the dominant types of traffic in the internet [6]. Packet erasures are one of the primary causes leading to excessive delay and buffering at the receiver [1].

One of the approaches to provide resilience to packet erasures includes layered coding where the media content is transmitted in layers in order to account for low bandwidth conditions [7], [8]. It has been demonstrated that network coding masks random losses from TCP and prevents the congestion control to be triggered for non-congestion related packet losses [9]. Network coding is an error-correction and throughput improving technique where packets from same flow or different flows can be combined and transmitted as payload without losing any information [10]. Based on the approach defined in linear network coding [11], the actual data is represented in the form of symbols from a finite field of size 2^8 or 2^{16}. Linear algebraic coefficients are randomly chosen from the finite field and are then combined along with the data bits to form the corresponding set of network codes. Consider a set of N messages, m_i, where $i \in Z$ and $0 \leq i < N$ and α_i is the set of linear algebraic coefficients randomly chosen from a finite field F associated to message, m_i. Then the set of output codes can be represented by y_i where:

$$y_i = \sum_{i=0}^{N} \alpha_i * m_i \tag{1}$$

It has been shown [11] that in order to be able to decode N messages with a high probability, N linearly independent network codes need to be received which form a full rank matrix at the receiver. However, in order to account for packet losses, we need to send redundant packets. Thus, in order to be able to decode all N messages successfully, $\Gamma * N$ number of packets are sent where Γ is the redundancy factor maintained anywhere between 1 and 1.5 [9]. The value of N is usually fixed and is called the bucket size. In the case of video streaming with network coding, the video

content that is present in the video source buffer is segmented by the TCP layer and the network coding layer then combines N such segments and sends N or more combinations to the destination.

Some of the first few papers on network coded TCP [9], [12] prove the effectiveness of network coding for TCP traffic by modifying the behavior of the acknowledgment process in TCP to improve the overall throughput of the system, at the cost of a decoding delay at the receiver. Recent research has focused on using inter-session network coding and demonstrated its potential advantages for high quality streaming media [13], [14].

This paper analyzes the interaction between TCP (New Reno) and an intra-session network coding module inserted in the network stack to improve the video quality. Latency and jitter are considered as quality evaluation metrics as they represent the fidelity of streaming data. A key problem is identified that is caused due to the congestion control nature of TCP and also provide a simple heuristic update algorithm to mitigate the specified problem. With the help of simulations, significant improvement in latency and jitter is observed at video playback for TCP traffic coupled with network coding.

Section II on related work, summarizes different solutions that have been proposed to overcome the challenges associated with delivering streaming video data over wireless networks. Section III describes our system model and assumptions. Section IV presents the analysis, implementation details and results observed for different scenarios followed by inferences. The future work arising from this work is discussed in section V followed by the concluding section VI.

2. RELATED WORK

One of the seminal works on quality video streaming over wireless networks [1] illustrates that packet level redundancy can be achieved by using erasure codes. Redundancy and robustness for streaming video have been achieved by layered and descriptive coding techniques, which provide an incremental improvement in the quality of streaming video which varies based on the type of packet losses incurred and the available bandwidth [7], [8].

Another form of erasure correction, network coding can also be used to improve the performance of video streaming services [15]. The use of network coding for multimedia, especially video streaming, has been shown to provide error control and also delay optimized delivery of video content [13]. An approach to combine both layered coding and network coding was evaluated in [16] and has been shown to negatively impact the video content delivery of the system, evaluated based on the number of layers of video decoded by the receiver. [15], provides an extensive overview of the interaction between network coding and media streaming applications and illustrates the advantages of network coding for streaming content delivery in Peer-to-Peer (P2P) streaming and priority based streaming with suitable examples. A novel perspective on video streaming was presented in [17], where an optimization framework was designed based on the cost constraint of minimizing initial start-up delay and interruptions while streaming. Although, network coding was used, it was mainly designed for P2P networks. They also showed that there was a trade-off in achieving interruption-free streaming, by buffering packets and the initial start-up delay. Another paper on buffering analysis for intra-session network coding [18] investigates the effect of number of packets combined, also

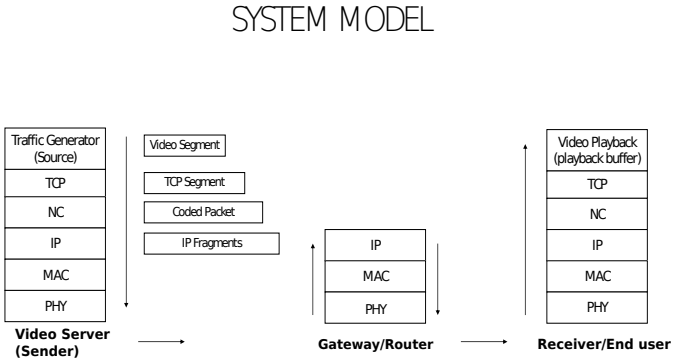

Fig. 1. System model depicting the video source and receiver's network stack; The arrows indicate the direction of packet flow along the stack

known as the generation (or bucket) size on the performance of the system for video streaming sessions but again it primarily focuses on multicast sessions while many of the internet based streaming applications involve unicast traffic. A recent paper on coded TCP [19] mimics the behaviour of TCP in the user space to achieve high throughput for lossy environments. However, their approach uses the delay bandwidth product to determine the coding block size which does not factor in the nature of the arriving traffic. To the best of our knowledge, none of the other work has explored the impact of TCP's congestion control coupled with intra-session network coding for unicast video streaming sessions in a wireless network focussing on providing low latency and jitter. The paper addresses this by the following:

(1) Observing and identifying the cause for latency at the receiver.
(2) Providing a solution to the problem, reducing the latency problem introduced due to network coding.
(3) Demonstrating latency and jitter improvement caused due to the modified network coding algorithm for both simulated and real-world traffic traces.

3. SYSTEM MODEL AND ASSUMPTIONS

Consider a multihop wireless network where the **video server** (**source**) node is responsible for generating and streaming video traffic to the end user. Video traffic can be mainly categorized into two types: video conference and full motion video. In case of a video conference type traffic, a steady bit-rate traffic is generated while full motion video creates a sequence of variable bit-rate traffic. A rapidly changing background creates a high bit-rate traffic while a steady background or a highly correlated set of consecutive video frames causes a low bit-rate traffic [20]. In our experiments, we also use actual traces captured from stored videos in MPEG4 format for realistic evaluation of our algorithm [21]. Assume a time slotted system and the server uses TCP as a transport protocol to provide reliable video transmission. The system model can be seen in figure 1. The server generates network coded packets by performing intra-session network coding using random linear packet combinations. The gateways simply forward these network coded packets which are eventually decoded by the **decoder** module present in the NC layer of the **receiver/end user** using Gaussian Elimination technique. The decoded packets are passed upto the playback buffer to maintain a steady rate of video

playback. The **playback buffer** is responsible for steady playback of received video content. We assume that the video playback occurs at a constant rate and there are no deliberate pauses caused by the user. We also assume that there is negligible delay between decoding the arrived coded packets and sending the decoded packets up to the playback buffer.

The following subsections describe the cross layer communication process between the TCP layer and the network coding layer for a streaming video application.

3.1 Traditional TCP

In the slow-start phase of TCP congestion control, the value of the congestion window rises exponentially with the reception of every ACK received. However, for an uncongested channel, the source (if it has data to send) should be able to send more data than the updated congestion window size. It has been shown [6] that TCP misinterprets the random packet losses due to wireless channel fading as congestion in wireless networks. As a result, congestion control mechanism is triggered which leads to an undesirable reduction in source sending rate. To avoid this undesirable control mechanism, the MAC layer in the intermediate nodes initiate the automatic repeat request (ARQ) retransmission process that prevents some of the link losses from being acknowledged as congestion [22]. These retransmissions do not affect the source sending rate, but they do add an unnecessary delay in data delivery.

3.2 TCP with network coding

A network coding layer is inserted in between TCP and IP layers that creates coded packets from the TCP segments passed down the stack. The operation of simple random linear network coding with TCP can be explained by considering 2 scenarios. Consider N TCP packets in the buffer where k is the bucket size and $cwnd$ represents the congestion window. For $N \geq k$, k packets are combined together and $\Gamma * k$ packets are transmitted. Since the operation is agnostic of the size of $cwnd$, the sending rate control is not achieved as optimally as TCP alone would normally do. In the second scenario, when $N < k$, in order to account for the fixed k packets, the network coding layer inserts zero payload packets. These zero payload packets satisfy the network coding process, but when they reach the receiver's buffer and are decoded, not all k packets have valid information to be decoded by the video playback. As a result, the video playback buffer needs to wait longer to receive the next valid video data, thus adding latency at the playback for such zero payload instances.

In this paper, a variable bucket size k is proposed to resolve the issue of added latency that lead to interruptions in service. To understand the problem, consider the following setup where the application generates 512 bytes of data at 500 kb/s with a 60% duty cycle and a fixed bucket size of two packets (and redundancy factor of one), which causes each TCP segment to become 536 bytes long. Initially, the source generates 2048 bytes of video content. The sender sends two coded packets of size 536 bytes each before receiving an ACK for receiving two data packets causing an update in the congestion window. In the meantime, the amount of video content added by the source has gone to 1584 bytes (as 1072 bytes have been received and 512 bytes are added to the source queue as seen in the figure 2). The size of the congestion window increases to four packets. So, now, the sender sends two coded packets, and forms another set of two coded packets (without waiting for an ACK), one of which is a random zero payload packet. The problem arises because the update in congestion window has no effect on the bucket size which leads the network coding layer to create zero payload packets (in the absence of actual video payload) to account for the bucket size.

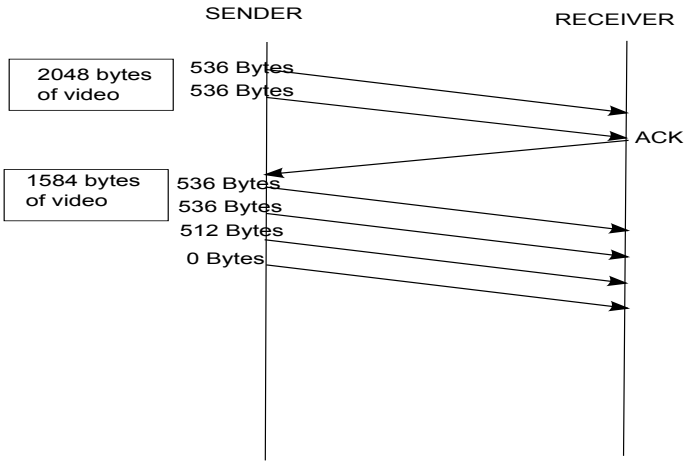

Fig. 2. Example of a zero payload packet creation for a fixed bucket size of 2 packets

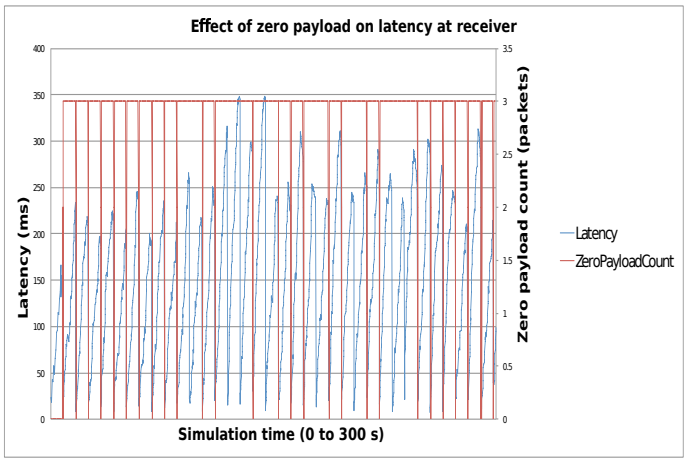

Fig. 3. Latency of video data reception at the playback buffer over simulation and effect of zero payload packets on latency for a fixed bucket size of four

In the figure 3, it can be seen that adding network coding has created instances of zero payload packets being received at the video playback which when superimposed over the latency observed for the entire simulation of 300 seconds, indicates that the latency in receiving valid video data is relatively higher when the zero payload count increases. This can be seen from the secondary vertical axis that indicates the increment in the number of zero payload packets sent.

3.3 Variable Bucket Size Intra-Session Network Coding

The objective is to ensure that no such zero payload instances occur that will lead to undesirable latency at the video playback. We propose that if the bucket size is dynamic, dependent on the amount of data available and adaptive to network conditions, there

will be no additional packets required to satisfy the constraint placed by the network coding algorithm. In essence, no zero payload packets are created which ensures that no latency is added by the network coding algorithm's constraints.

Initially, the bucket size is kept at 1, which means that the source node attaches only a single algebraic coefficient to form a network coded packet for transmission. On receiving an ACK, the congestion window is updated (increased) and the bucket size also increases based on the number of packets waiting to be transmitted at the sender. An increased congestion window implies an increase in the available transmission opportunity and more packets are drawn from the source video queue. The bucket size that determines the number of packets to be combined to form network coded packets is now calculated based on the available transmission opportunity and the amount of data present in the source video queue. Instead of a predetermined fixed bucket size, we now use the following pseudo code to determine the number of packet combinations to be sent across the network from the source. Let the available window be represented by w, the TCP buffer size by b and data in bytes still flight, that are still to be acknowledged be represented by $unack$.

if $w \geq (b - unack)$ **then**
  $bucketSize \leftarrow (b - unack)/maxPktSize$
else if $w \leq (b - unack)$ && $w \geq unack$ **then**
  $bucketSize \leftarrow (w - unack)/maxPktSize$
end if

With this modification to the block based encoding scheme, the bucket size becomes dynamic and dependent on the available packets at the source. As a result, the source node does not wait for packets to arrive before encoding and for a given transmission window, more packets are picked from the video source queue and encoded using random linear coding. We, then, check if the available transmission window is greater than the size of the video source queue. In our algorithm, we only consider the size of the video data queued at the TCP layer that are waiting to be transmitted which can be seen from the formula in the above pseudo code. As more packets arrive at the source queue from the application layer, the bucket size varies to accommodate for the increase in load and prevents excessive buffering at the source. When there are random losses in the network, the losses are masked from the sender and the source sending rate is unaffected because the coded packets sent ensure that as long as innovative packets are received, an ACK is sent to acknowledge the increase in the degree of freedom, keeping TCP unaware of any packet loss. When congestion in the network occurs, which is indicated by a missed ACK or 3 duplicate ACKs, the bucket size is adjusted based on the updated congestion window, which takes into account the missing degrees of freedom. Now, the available transmission opportunity reduces causing a decrease in the bucket size and slowing down the sending rate at the source.

4. PERFORMANCE ANALYSIS

4.1 Implementation

This section presents the ns-3 simulator's framework used to test the proposed system. The network coding module implemented in C by Keller [23] was used as a building block to insert between the TCP and IP layers of the stack. The following sub-section summarizes the network coding module used and the integration of the network coding module in ns-3 is summarized in subsequent sections.

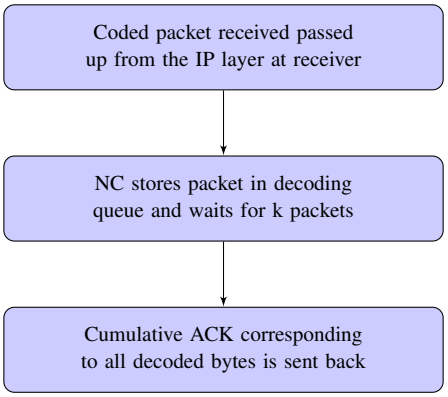

Fig. 4. Flowchart for packet flow at receiver

4.1.1 Random Linear Coding. The network coding module used for the evaluation was implemented using an existing model based on random linear network coding [23]. This module is responsible for creating a finite field, GF (2^{16}), forming encoding coefficients and global coefficient matrix based on the block size and the decoder. Using Gaussian elimination technique, the decoder decodes the received packets successfully if the received encoded packets are linearly independent.

4.1.2 Integration into ns-3. The placement of the network coding layer in the stack was decided based on the amount of complexity added in the implementation. If the network coding layer was implemented directly on the application layer packets, then it would not be able to mask the random packet losses due to the wireless channel from the congestion control action of TCP. Thus, the network coding layer was chosen to be inserted in between the transport and IP layer. The intermediate nodes are unaware of the presence of the network coding layer as they will simply forward the packet after checking the MAC. ns-3 facilitates quick integration of a new module due to salient features like helper modules and highly detailed PHY, MAC and the rest of the network stack [24].

Packet reception is summarized in figure 4. The TCP ACKs are modified to indicate the reception of cumulative number of innovative packets, known as the degree of freedom enabling faster processing which still serves the purpose of flow control. The network coding header allocates 8 bits to inform the receiver of the bucket size decided at the sender. The elements of the network coding header are as shown in the figure 5. The random seed in the header is used to ensure that the receiver uses the same finite field as the sender for successfully decoding the packet.

4.2 Simulation

Experiments are performed in case of a 3 node network as well as a larger network with increased traffic to demonstrate the effect of the congestion control mechanism and its dynamics on the performance of the system. The 3 node network is a string topology with a sender (video server), a receiver (end user) and an intermediate node (gateway). The large network consists of 20 nodes randomly placed on a 100*100 area. The simulation setup has been tabulated in 1. The packet error model for the wireless channel in ns-3 is based on [25]. The system has been first evaluated based on the proportion of zero payload packets received out of the total packets

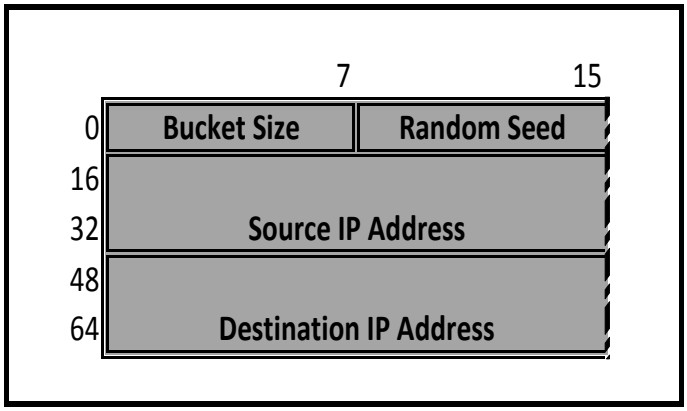

Fig. 5. Network coding header structure

Table 1. Simulation Setup

Parameter	Description/Value
Number of nodes	20
Mobility	Static
Routing	AODV
Propagation Loss	Friis
PER model	NIST Error model [25]
Video source	VBR traffic generator in ns-3
Max. application packet size	1472 bytes
Mean application data rate	1 Mbps

received, leading to latency at the end user's playback buffer. The effects on latency and jitter observed at the end user's playback buffer for the corresponding approaches are analyzed. It is further demonstrated that the latency observed for real-world video stream traces using our approach is significantly lower than that observed on TCP and TCP with fixed bucket size network coding.

4.3 Numerical Results

4.3.1 3-Node Topology. Consider a 3 node string topology where the intermediate node is only responsible for forwarding the packets. The sending node is the only traffic generating node in the network and, thus, it is never congested. After the initial increase in the transmission window, the available transmission opportunity does not vary significantly for the duration of the entire simulation of 300 seconds. In this scenario, for a fixed bucket size of two, it is observed that as the available transmission window changes, zero payload packets are created. With the proposed variable bucket size based network coding, all zero payload packet occurences are removed. The evolution of the congestion window is superimposed on the performance graph as seen in the figure 6 to show the effect of varying congestion window on the performance of the system. As the change in the congestion window is agnostic to the availability of data at the video source, there is a mismatch in the number of packets available for transmission with an increase in the transmission opportunity. This is not taken care of by the fixed bucket size approach while the variable bucket size scheme adapts itself to available traffic and the size of the transmission window.

4.3.2 Large Network Topology. Consider a network with 20 nodes randomly placed where several nodes are communicating with each other and there are multiple types of traffic flowing in the

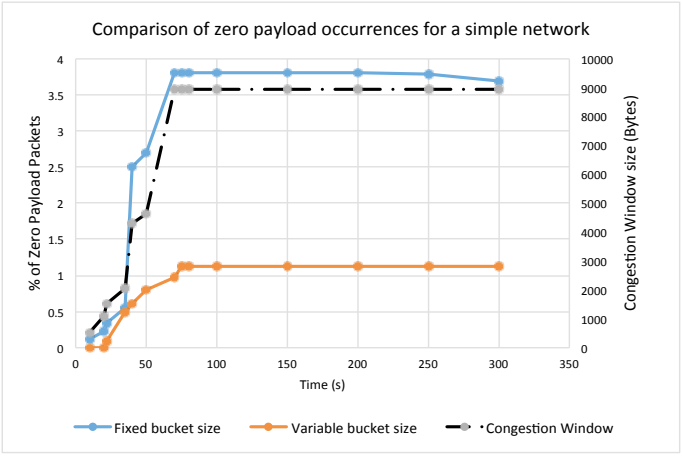

Fig. 6. Comparison of number of zero payload occurrences for fixed and variable bucket size network coding for a simple 3 node network

Table 2. Parameters of different traffic in the experiment

Parameter	CBR Traffic	Web browsing	Bulk file transfer
Transport Layer	UDP	TCP	TCP
Data Rate (Mbps)	1	2	2
Max. file size (Bytes)	512	1400	100000
Source node	0	1	3
No. of hops to sink	4	3	5

network. To introduce congestion in the network, we create three other applications, a constant bit rate traffic on UDP, a web browsing session using HTTP and bulk file transfer using TCP together with the VBR traffic in the network. The parameters used for the different traffic types are as shown in the table 2.

In highly dynamic network conditions, the available transmission window is not stable for long (as can be seen from figure 7), consequently leading to a large number of zero payload packets being received. On the other hand, the dynamic nature of the channel has a negligible effect on the performance of the system with our variable bucket size based network coding approach. With frequent changes in the size of the congestion window, the fixed bucket size approach suffers from excessive number of zero payload occurrences which is not observed on the variable bucket size approach.

4.4 Evaluating latency and jitter

One of the key performance metrics for a video streaming system is the latency. Latency is defined as the time it takes for the server's video application packet to travel down the stack from the application layer and be delivered to the application layer at the receiver [26], [13]. Since, we are evaluating our algorithm's performance, we consider the time taken only after the video encoding process at the application layer to calculate the latency as we don't impose any constraints on the video encoding overhead in the application layer. In figure 8, we observe that the latency in delivering the video data reduces to almost $1/10^{th}$ of that caused by using the traditional TCP transport. The added latency is due to packet erasures caused in the wireless channel. However, with the variable bucket size, the latency further reduces showing a reduction of almost $1/100^{th}$ compared to traditional TCP. This significant improvement provided by the variable bucket size algorithm can be

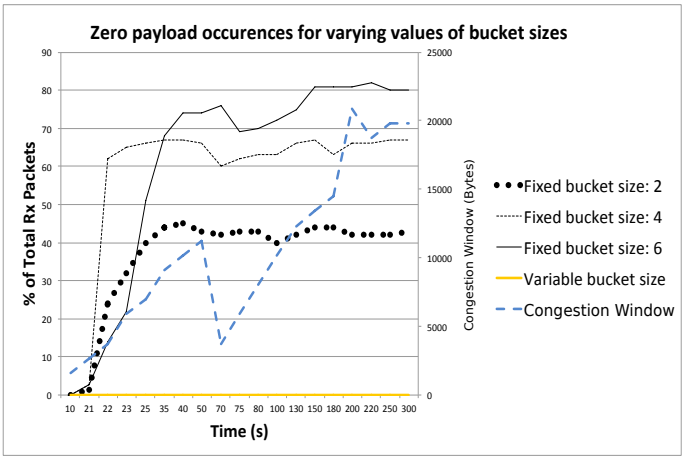

Fig. 7. Comparison of number of zero payload occurrences for fixed and variable bucket size network coding for a large network

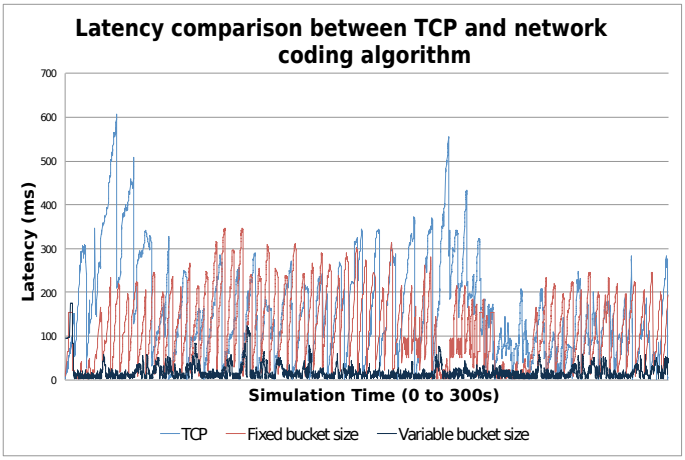

Fig. 8. Comparison of video data delivery latency for TCP, TCP with fixed bucket size and variable size network coding for the large network topology

Table 3. Parameters of "Silence of the Lambs" video trace

Parameter	Value
Peak Bit Rate (per sec)	$4.4 * 10^6$
Mean Bit Rate (per sec)	$5.8 * 10^5$
Mean Frame Size (Bytes)	$2.9 * 10^3$
Max. Frame Size (Bytes)	22239
Min. Frame Size (Bytes)	158

attributed to the aggressive behaviour of the algorithm to transmit the data based on available transmission opportunity.

However, if the bucket size were to increase as the amount of available transmission opportunity, the receiver at the network coding layer would end up waiting for large number of packets before passing them up to the video playback and the TCP's retransmit timer at the sender would time out causing undesirable retransmissions. Thus, based on heuristic data, an upper bound of 15 packets is set on the bucket size in order to avoid these

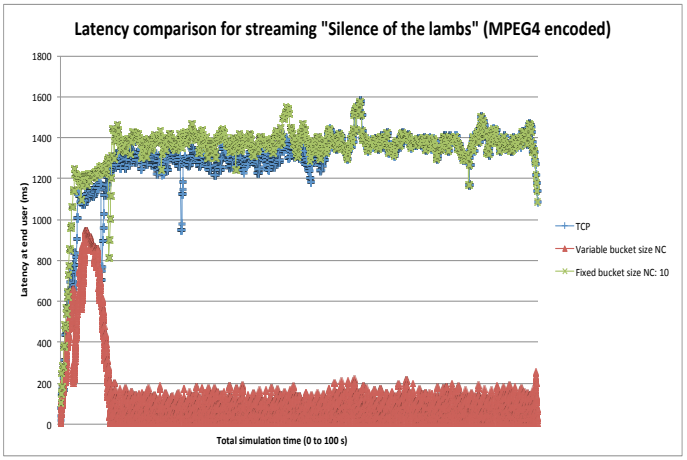

Fig. 9. Latency comparison for video trace of Silence of the Lambs for 100 seconds

retransmissions and keep the latency below 250 ms. Future work to determine an analytical expression for the bucket size could ensure a low latency experience for video playback.

In order to obtain a realistic perspective on these metrics, a real-world video trace is used that is an MPEG4 encoded file [21]. The parameters of the video data used to test our algorithm are in the table 3. For "Silence of the Lambs", as the packet sizes are larger compared to our VBR traffic generator in ns-3, significant latency is observed for TCP and TCP with fixed bucket size network coding as can be seen from the figure 9. However, due to the adaptive nature of the variable bucket size algorithm, larger packets are handled quickly and thus, the overall latency at the application layer of the end user is almost always below 250 ms, except initially where the latency is high due to the slow-start nature of the TCP congestion control.
Since latency and jitter are two important performance metrics that quantify the performance of a video streaming service [17], we determined the jitter for only TCP and TCP with both fixed and variable bucket size network coding. When plotted against the total simulation time (figure 10) it can be seen that the jitter for variable bucket size network coding is the least of all the other techniques and the peak jitter is around 180 ms which is seen only at the beginning of the simulation which is the initial start-up time involved in the algorithm.

4.5 Discussion

From our experiments, the following can be inferred:

—Despite a stable network and a constant transmission window opportunity, a network coding system with fixed bucket size generates zero payload packets affecting the latency of video streaming service. The quality of user experience, which is significantly affected by this latency observed at video playback [27], degrades further if the wireless network consists of multiple types of traffic that lead to congestion and contention of the wireless channel.

—Our proposed algorithm performs significantly better than the fixed bucket size counterpart for both uncongested and congested environments due to its ability to adapt to the network and video traffic conditions. As observed in the results, there are no zero

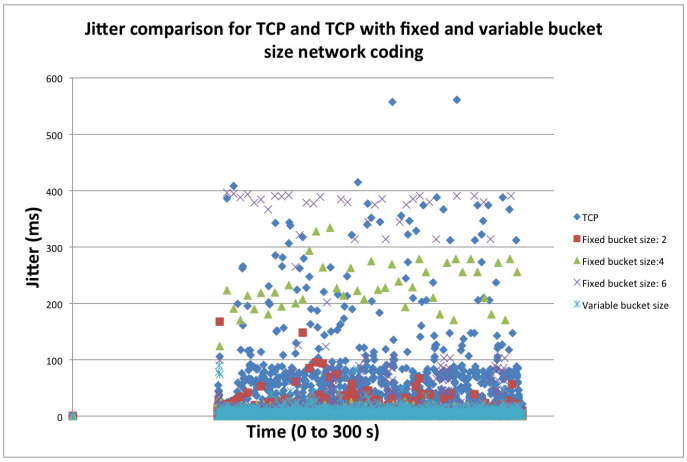

Fig. 10. Comparison of jitter for TCP, TCP with fixed bucket size and variable size network coding for the large network

payload packets observed, consequently leading to lower latency at the video playback.

—Timely delivery of video data is imperative for service providers and we observe from our experiments that with TCP the worst case latency can go upto 600 ms and when TCP with network coding is used, the latency goes upto 300 ms. However, with our variable bucket size algorithm, the delay is as low as 5 ms which is a tremendous improvement in the performance.

—Jitter is an equally important parameter as a streaming system with low jitter signifies that two consecutive packets are uniformly delayed. For uninterrupted viewing, the streaming should be jitter free [28]. Jitter in a streaming system [29] is avoided by having a delay jitter buffer at the receiver placed before the video decoding buffer that compensates for channel and traffic arrival variations. However, [30] showed that a single receiver buffer is sufficient to achieve a certain QoS guarantee. It can be seen from our experiments that the jitter after using variable bucket size network coding improves by 70% when compared with only TCP and by 60% when compared with TCP and fixed size network coding. The improvement in jitter can be attributed to the fact that the variable bucket size at the receiver also acts as a delay jitter buffer which compensates for variations caused due to the incoming traffic and network conditions.

5. FUTURE WORK

Due to the heuristic approach of the algorithm, there is a need for an analytical framework to determine bounds on the bucket size which eventually, affects the latency and jitter at the end user. As a part of the future work, we propose to use a dynamic programming framework with the aggregate objective of minimizing latency and jitter under the constraints of the arriving video traffic and congestion control methods. Another future area of study is the impact of the latest developments in the real-world implementation of the TCP stack on linux that includes the "Controlled Delay(CoDel)" algorithm [31] for preventing excessive queueing at intermediate nodes, on our algorithm's performance. The reason for evaluating our approach on TCP with CoDel is to analyze the impact of delay based packet drops in the intermediate nodes and how that leads to corresponding changes in the bucket size.

6. CONCLUSION

In this paper, the interplay between the behavior of TCP's congestion control mechanism and intra-session network coding was analyzed using simulations in ns-3. A significant performance lapse in terms of latency was observed at the video playback for streaming applications. A simple approach to dynamically adapt the bucket size before performing network coding on the video packets was proposed to get rid of the zero payload packets that cause the undesirable latency at the receiver. Simulation results demonstrated that the variable bucket size algorithm removes all instances of zero payload occurrences for both congested and uncongested environments reducing the latency and jitter at the video playback. Our work, thus, identifies one of the issues arising when TCP and intra-session network coding interact in a video streaming environment and successfully mitigates it using a simple heuristic algorithm while also providing timely delivery of streaming video data.

7. ACKNOWLEDGMENTS

This work was supported by the DARPA Computer Science Study Group Program, Grant Number HR0011-09-1-0039.

8. REFERENCES

[1] Hui Wang, Joyce Liang, and C Jay Kuo. Overview of Robust Video Streaming with Network. *Journal of Information Hiding and Multimedia Signal Processing*, pages 36–50, 2010.

[2] Niveditha Sundaram, Parameswaran Ramanathan, and Suman Banerjee. Multirate media stream using network coding. In *Proc. 43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005.

[3] Xiaoqing Zhu and Bernd Girod. Video Streaming Over Wireless Networks. *Proceedings of the European Signal Processing Conference, EUSIPCO-07, Poznan, Poland*, 2007.

[4] Majed Haddad, Eitan Altman, Rachid El-azouzi, Tania Jiménez, Salah Eddine Elayoubi, Sana Ben Jamaa, Arnaud Legout, Ashwin Rao, Université Avignon, France Telecom, and Issy Moulineaux. A Survey on YouTube Streaming Service. *Proceedings of the 5th international ICST conference on performance evaluation methodologies and tools*, pages 300–305.

[5] Monia Ghobadi and Matt Mathis. Trickle : Rate Limiting YouTube Video Streaming. *Proceedings of the USENIX Annual Technical Conference (ATC)*, page 6, 2012.

[6] Kai Xu Ye Tian and Nirwan Ansari. TCP in Wireless Environments: Problems and Solutions. *IEEE Radio Communications*, pages 27–32, March 2005.

[7] Jiang Guo, Ying Zhu, and Baochun Li. CodedStream : Live Media Streaming with Overlay Coded Multicast. *In Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking (MMCN 2004)*.

[8] J. Chakareski, S. Han, and B. Girod. Layered coding vs. multiple descriptions for video streaming over multiple paths. *Multimedia Systems*, 10(4):275–285, April 2005.

[9] Jay Kumar Sundararajan, Devavrat Shah, Muriel Medard, Szymon Jakubczak, Michael Mitzenmacher, and Joao Barros. Network Coding Meets TCP: Theory and Implementation. *Proceedings of the IEEE*, 99(3):490–512, March 2011.

[10] R. Ahlswede, S.-Y.R. Li, and R.W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[11] S.-Y.R. Li and R. W. Yeung. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, February 2003.

[12] P. Samuel David and Anurag Kumar. Network coding for TCP throughput enhancement over a multi-hop wireless network. *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, pages 224–233, January 2008.

[13] H. Seferoglu and A. Markopolou. Delay-Optimized Network Coding for Video Streaming over Wireless Networks. *2010 IEEE International Conference on Communications*, pages 1–5, May 2010.

[14] Kien Nguyen, Thinh Nguyen, and Sen-Ching Cheung. Video Streaming with Network Coding. *Journal of Signal Processing Systems*, 59(3):319–333, February 2009.

[15] Nikolaos Thomos and Pascal Frossard. Network Coding and Media Streaming (Invited Paper). *Journal of Communications*, 4(9):628–639, October 2009.

[16] Rohan Gandhi, Meilin Yang, Dimitrios Koutsonikolas, Y Charlie Hu, Mary Comer, Amr Mohamed, and Chih chun Wang. The Impact of Inter-layer Network Coding on the Relative Performance of MRC / MDC WiFi Media Delivery. *Optimization*, pages 1–6, 2011.

[17] Ali ParandehGheibi, Muriel Medard, Asuman Ozdaglar, and Srinivas Shakkottai. Avoiding Interruptions A QoE Reliability Function for Streaming Media Applications. *IEEE Journal on Selected Areas in Communications*, 29(5):1064–1074, May 2011.

[18] Brooke Shrader and Anthony Ephremides. Queueing Delay Analysis for Multicast With Random Linear Coding. *IEEE Transactions on Information Theory*, 58(1):421–429, January 2012.

[19] Minji Kim, Jason Cloud, Ali Parandehgheibi, Leonardo Urbina, Kerim Fouli, Douglas Leith, and Muriel Médard. Network Coded TCP (CTCP). *arXiv preprint arXiv: 1212.2291*, 2012.

[20] Savera Tanwir and Harry Perros. A Survey of VBR Video Traffic Models. *IEEE Communications Surveys & Tutorials*, PP(99):1–25, 2013.

[21] Martin Reisslein and Frank H.P. Fitzek. MPEG4 and H.263 Video Traces for Network Performance Evaluation. *IEEE Network*, (December):40–54, Nov/Dec 2001.

[22] Yegui Cai, Shengming Jiang, Quansheng Guan, and F Yu. Decoupling congestion control from TCP (semi-TCP) for multihop wireless networks. *EURASIP Journal on Wireless Communications and Networking*, 2013(1):149, June 2013.

[23] L. Keller. Ncutilsc. http://code.google.com/p/ncutils/.

[24] Network simulator 3. `http://www.nsnam.org/ns-3-dev`. Accessed: 2010-09-30.

[25] Guangyu Pei and Thomas R Henderson. Validation of OFDM error rate model in ns-3. *Boeing Research Technology*, pages 1–15, 2010.

[26] Mark Claypool and Jonathan Tanner. The Effect of Jitter on the Perceptual Quality of Video. *Proceedings of ACM Multimedia*, 1999.

[27] Gupta P. Vishwanath A. Perspectives on Quality-of-Experience for Video Streaming over WiMax. *Mobile Computing and Communications Review*, 13(4):15–25.

[28] Guanfeng Liang and Ben Liang. Jitter-free probability bounds for video streaming over random VBR channel. *Proceedings of the 3rd international conference on Quality of service in heterogeneous wired/wireless networks - QShine '06*, page 6, 2006.

[29] V. Varsa and I. Curcio. Transparent end-to-end packet switched streaming service (PSS);rtp usage model (release 5). *3GPP TR 26.937 V1.4.0*, (June), 2003.

[30] Jenka Hrvoje Stockhammer T. Streaming Video Over Variable Bit-Rate Wireless Channels. *IEEE Trans. Multimedia*, 6(2):268–277, 2002.

[31] Kathleen Nichols and Van Jacobson. Controlling Queue Delay. *Proceedings of ACM Queue*, 2012.