ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

TRANSFORMING NETWORK CODING WITH TCP FOR BROADCASTING STREAMING DATA IN MULTI-HOP WIRELESS LAN

Ashish Parejiya*, Dr. Vinod Desai
*PhD Scholer(CSS), Mewar University, Chittorgarh, INDIA
Computer Science Departments, Govt. Science College, Navsari, INDIA

ABSTRACT

This paper focuses on using Network Coding (NC) with TCP for multi-hop wireless networks to provide fault-tolerant and timely delivery of streaming data. The paper shows that there is an inherent latency in video playback when TCP with random linear NC is employed. With the objective of reducing latency and jitter at the receiver, the paper proposes a Variable Bucket size based Network Coding (VBNC) technique that modifies the TCP congestion control to adapt to the arriving traffic and dynamic network conditions. Simulation results demonstrate that on an average the proposed algorithm reduces observed latency at playback by 80% and jitter by more than 50% over standard TCP. More importantly, a significant reduction in the initial start-up delay is observed which enhances the performance of streaming services.

Keywords: Bucket size, congestion control, finite field, intra-session network coding, latency, random linear network coding, ns-3, round trip time, streaming media, video aware network coding

INTRODUCTION

Network coding has found many applications [1] in wireless networks, since the idea was first developed by Ahlswede, et. al [2]. Recent research has shown that due to the throughput-enhancing and error-resilient capabilities of NC in wireless networks, delay-optimized and quality media streaming delivery is achievable [3], [4], [5], and [6]. However, reliable streaming media delivery still faces challenges of low bandwidth availability, channel variations and congestion in the network [7], [8]. In order to achieve reliability, most of the streaming traffic use TCP as a transport protocol [9] which is designed to handle network congestion and varying bandwidth for wired networks [10]. On the other hand, TCP adds inherent delays in wireless networks mainly due to undesirable retransmissions [11], [12].

Seminal work on combining TCP with NC, demonstrated enhancement in the throughput by avoiding undesirable packet retransmissions both using bidirectional flows of traffic [13] and modified TCP ACKs [14]. Recent work on coded TCP [6] presented a new congestion control technique for NC that entirely replaces the current TCP stack but provides significant benefits in throughput and data completion times. However, this technique requires a complete overhaul of the current TCP stack which makes it difficult to implement on already existing networks.

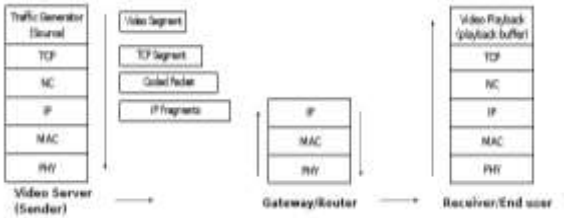


Fig. 1. System model depicting the video source and receiver's network stack; the arrows indicate the direction of packet flow along the stack

The main contribution of this paper is the modification to the existing TCP's packet-sending mechanism by using a variable coding block size (bucket size) NC technique. This technique is bounded by the amount of traffic available to send, transmission opportunity available from the TCP ACK feedback and the total round-trip-time obtained after receiving the TCP ACK. This paper will demonstrate using simulations that the proposed approach reduces average latency and jitter significantly, while also providing low latency at the start of a video in wireless networks. The paper is organized as follows: Section II describes our system model and assumptions. Section III presents our analysis, implementation details and results observed

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

for different scenarios followed by inferences. The future work arising from this work is discussed in section IV followed by the concluding section V.

Parameters	*Silence of the lambs*	*Formula 1*
Peak Bit Rate (per sec)	4.4×10^6	2.9×10^6
Mean Bit Rate (per sec)	5.8×10^5	8.4×10^5
Mean Frame Size (Bytes)	2.9×10^3	4.2×10^3
Max. Frame Size (Bytes)	22239	14431
Min. Frame Size (Bytes)	158	130

TABLE I. PARAMETERS OF THE DIFFERENT VIDEO TRACES

I. SYSTEM MODEL AND ASSUMPTIONS

Our system model is presented in figure 1. We consider a multi-hop wireless network where the video server (source) node is responsible for generating and streaming video traffic to the end user. In our experiments, we use actual traces captured from stored videos in MPEG4 format for realistic evaluation of our algorithm [15]. The parameters of the different video streams used are tabulated in I. The server uses TCP as a transport protocol to provide reliable video transmission and it generates network coded packets by performing intra-session NC using random linear packet combinations. The channel error model is based on the NIST error model for 802.11b [16]. The gateways forward this 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 up to 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.

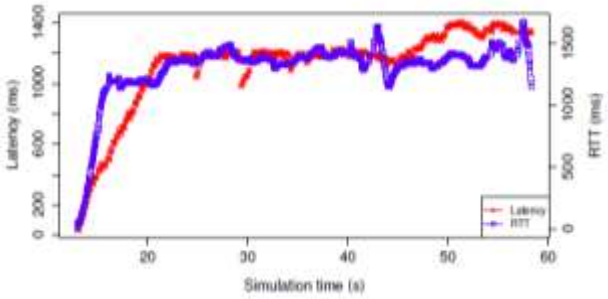
Next, we describe the cross layer communication process between the TCP layer and the NC layer for a streaming video application.

Standard 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 [11] 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 [17]. These retransmissions do not affect the source sending rate, but they do add an unnecessary delay in data delivery which is observed in the round-trip-time (RTT) computed at the TCP layer. The sender's TCP layer estimates the RTT for every ACK received based on the mean RTT observed over previously obtained samples of RTT along with a factor of the mean deviation in RTT [10], [18].

A. TCP with NC

A NC layer is inserted in between TCP and IP a layer that creates coded packets from the TCP segments passed down the stack. The operation of simple random linear NC with TCP can be explained by considering two scenarios. Consider N TCP packets in the buffer where k is the bucket size and cwnd represents the congestion window. For N >= k, k packets are combined together and G 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 NC layer inserts zero payload packets, which effectively means that the sender waits for more packets to arrive to fill the coding buffer. An estimated RTT is computed for every cumulative ACK sent by the receiver that indicates the reception of the packets coded together of bucket size k. So, with k, the value of estimated RTT increases.



ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Fig. 2. Relationship between end-to-end latency and RTT for fixed bucket size: 4 RLNC for the video trace "Silence of the lambs"

We demonstrate the implication of increasing RTT on the end-to-end latency observed at the receiver. It is clear from the figures 2, 3 that there is an almost proportional increase in latency with increase in RTT. A latency of as high as 1.2 s is observed which would completely kill a streaming session [19]. We perform these experiments using the trace for "Silence of the lambs" over a period of 100 s.

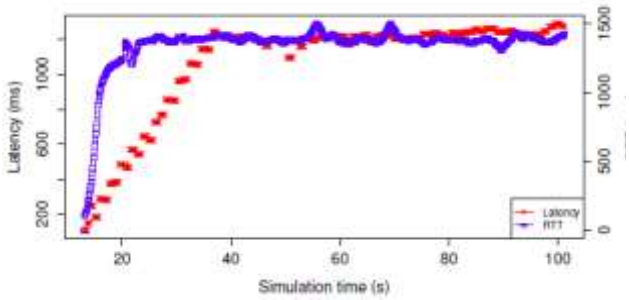


Fig. 3. Relationship between end-to-end latency and RTT for fixed bucket size: 10 RLNC for the video trace "Silence of the lambs"

B. Variable Bucket Size Intra-Session Network Coding (VBNC)

In order to mitigate the problem of added latency due to RLNC, we modify the coding bucket size based on three parameters:

- Congestion window: TCP's congestion window is an application layer agnostic parameter. Since, the NC layer lies below the transport layer, we utilize the information regarding the available transmission window and number of bytes that are waiting to be acknowledged
- Available traffic: Depending on the kind of traffic to be streaming, different media sources are characterized by packet sizes and arrival rates as shown in the table I
- RTT: TCP's RTT indicates the available bandwidth which is used to determine the new congestion window. However, the fixed bucket size NC affects the total RTT as shown in figures 2 and 3

At the initialization stage, coding bucket size is 1. 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, the amount of data present in the source video queue and the bytes that are yet to be acknowledged, called bytes in flight. 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 bytes in flight be represented by unack. The available window is essentially determined by mincwnd, rwnd where cwnd is the current size of the congestion window and rwnd corresponds to the number of bytes the receiver can hold indicated in the ACK sent.

If w = (b - unack) **then**

bucketSize ? (b - unack)/maxPktSize

Else if w = (b - unack) && w = unack **then**

bucketSize ? (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. This results in an aggressive source sending rate which benefits media streaming traffic that are characterized similar to those specified in table I. When there are random losses in the network, the redundant G coded packets prevent undesirable retransmissions by TCP [20]. However, in order to prevent a timeout at the TCP source, for every innovative packet received, an ACK, indicating a new degree of freedom is sent [14]. 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.

However, from figures 2 and 3, we conclude that there is a significant impact of NC on RTT causing an increase in overall latency. So, we place an upper bound on the bucket size dependent on the RTT. The RTT from Google across US goes up to a maximum of 100 ms [21] and we use this bound on RTT to determine our bucket size. When the RTT goes above 100 ms, we reduce the bucket size by half. This is a heuristic approach which is conservative. More analysis is required to determine the analytical expression on the bounds for the coding bucket size.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

PERFORMANCE ANALYSIS

A. Implementation

In this section, we present the ns-3 simulator's framework used to test the proposed system. The NC module implemented in C by Keller [22] (based on Random Linear NC (RLNC)) was used as a building block to insert between the TCP and IP layers of the stack. The following sub-section summarizes the NC module used and the integration of the NC module in ns-3 is summarized in subsequent sections.

1) *Random Linear Coding:* This module is responsible for creating a finite field, GF (216), 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.

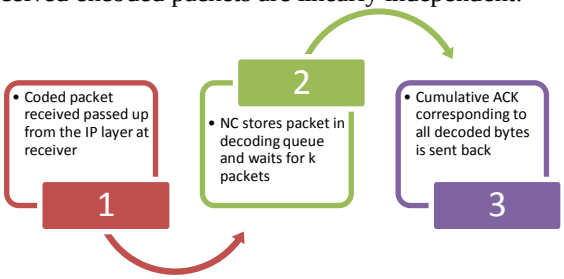


Fig. 4. Flowchart for packet flow at receiver

2) *Integration into ns-3:* The placement of the NC layer in the stack was decided based on the amount of complexity added in the implementation. If the NC 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 NC layer was chosen to be inserted in between the transport and IP layer. The intermediate nodes are unaware of the presence of the NC 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 [23].

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 NC header allocates 8 bits to inform the receiver of the bucket size decided at the sender. The elements of the NC header are as shown in the figure 5. The random seed in the header is used to ensure that the receiver generates the same set of coefficients for the finite field as the sender for successfully decoding the packet.

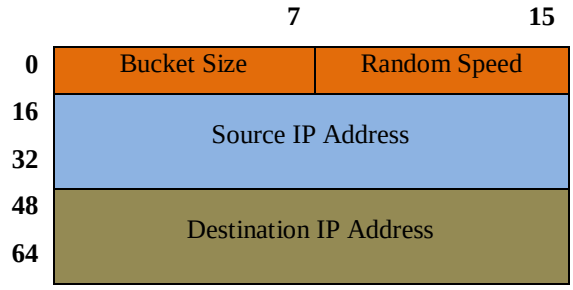


Fig. 5. Network coding header structure

A. Simulation

Our experimental set up is described in the table II. We simulate a topology where nodes are placed at random within a 1000 * 1000 square units of area. We used the real-world traces of "Silence of the lambs" and "Formula 1" [15] to plot the latency observed for TCP and the variable bucket size NC with TCP.

TABLE II SIMMULATION SETUP

Parameter	Description/Value
Number of nodes	20
Mobility	Static
Routing	AODV
Propagation Loss	Friis
PER Model	NIST Error Model [16]
Video Source	Video traces obtained from [15]

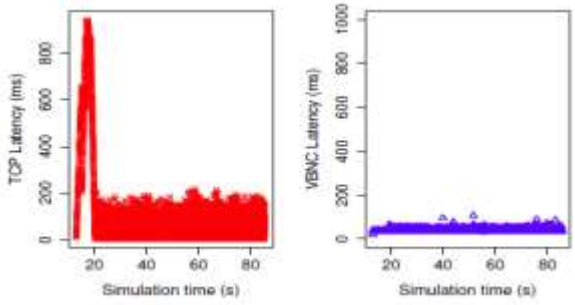


Fig. 6. Latency comparison between TCP and variable bucket size NC for "Silence of the lambs" trace

1) *Latency:* Among the different sources of latency [24], we only focus on the impact of TCP and the layers below. Thus, we define latency as the time it takes for the server's video application packet (after the process of video encoding) to travel down the

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

stack from the application layer and be delivered to the application layer at the receiver (before the process of video decoding) [24], [25]. In figure 6, we see that where standard TCP experiences a peak delay of more than 1 s, our algorithm causes a peak latency of up to 90 ms for the trace of "Silence of the lambs". On the other hand, the peak latency increases to approximately 200 ms for the "Formula 1" trace. We see that as compared to standard TCP there is an 80% reduction in latency at the receiver. The significant improvement provided by the variable bucket size algorithm can be attributed to the aggressive behavior of the algorithm to transmit the data based on available transmission opportunity.

2) Jitter: Jitter is another important metric that affects the performance evaluation of a streaming service [5]. We define jitter as the average difference in the video frame arrival observed at the receiver's playback buffer [5], [25]. This difference in the arrival leads to interruptions while video playback [26]. We plot the jitter variation for both the video traces considered for the latency experiment over a simulation time of 100 s. From the figure 8, we see that for "Silence of the lambs", at the start of the video, jitter for standard TCP is around 200ms as compared to 150ms for VBNC. However, the mean jitter for the duration of the video lies at around 200ms which is a 50% reduction than that in the case of TCP.

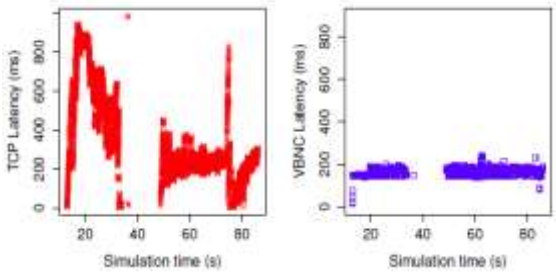


Fig. 7. Latency comparison between TCP and variable bucket size NC for "Formula 1" trace

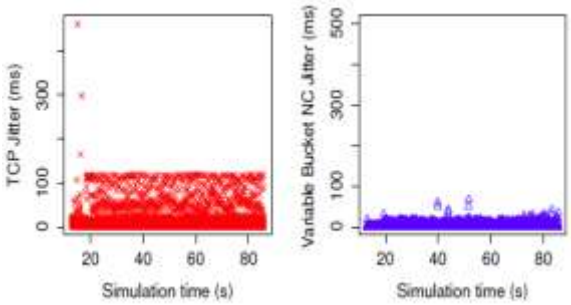


Fig. 8. Jitter comparison for TCP and variable bucket size NC for " Silence of the lambs" trace

Similarly, the mean jitter for the "Formula 1" trace in case of VBNC is approximately 60% less than that for TCP.

The increased jitter in the second case can be attributed to the fact that the mean frame sizes and data rate are higher than those for the "Silence of the lambs". However, the trend in latency and jitter is observed to be similar using our approach as opposed to standard TCP that incurs a much larger latency and jitter for the second case.

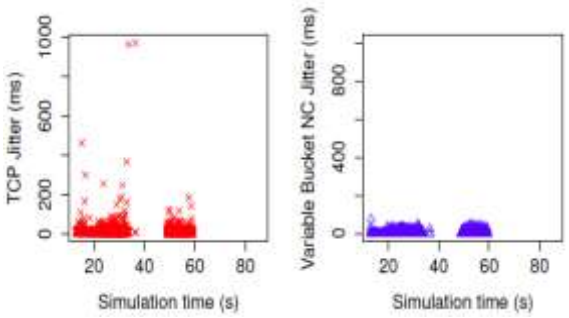


Fig. 9. Jitter comparison for TCP and variable bucket size NC for "Formula 1" trace

B. Discussion and Future Work

Although, RLNC has been shown to provide throughput enhancements for TCP [13], [14], it also adversely affects the overall RTT measured at TCP. Our proposed approach significantly outperforms TCP by reducing the end-to-end latency by 80% and jitter by more than 50% as demonstrated on real-world video traces. The stable response of our approach as seen from the experiments can be attributed to the fact that the upper bound on the bucket size is placed based on a desired RTT. This bound acts as a control feedback which ensures that the latency and jitter stay within desirable limits. There is a need to derive an analytical expression for the coding bucket size with the objective of minimizing latency and jitter. 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 [27] for preventing excessive queueing at intermediate nodes, on our algorithm's performance. Evaluating our approach on TCP with CoDel would help in analyzing the impact of delay based packet drops in the intermediate nodes and how that leads to corresponding changes in the bucket size.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

CONCLUSION

In this paper, we have proposed a modified NC algorithm that is implemented below the transport layer with minor modifications to the congestion control mechanism of TCP. This approach succeeds in providing low latency and jitter for streaming media as compared to the standard TCP which is corroborated by our simulation results that show an 80% reduction in end-to-end latency and more than 50% reduction in jitter for video data delivery. From a practical stand-point, minimal changes are required to be implemented at the TCP layer which enables its immediate deployment in real-world scenarios for streaming media in ad hoc wireless networks.

REFERENCES

1. Fragouli and E. Soljanin, "Network coding applications," *Foundations and Trends in Networking*, vol. 2, no. 2, pp. 135–269, 2007.
2. R. Ahlswede, S.-Y. Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=850663
3. N. Thomos and P. Frossard, "Network coding and media streaming (Invited Paper)," *Journal of Communications*, vol. 4, no. 9, pp. 628–639, Oct. 2009. [Online]. Available: http://ojs.academypublisher.com/index.php/jcm/article/view/1874
4. H. Seferoglu and A. Markopolou, "Delay-optimized network coding for video streaming over wireless networks," *2010 IEEE International Conference on Communications*, pp. 1–5, May 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5502651
5. A. ParandehGheibi, M. Medard, A. Ozdaglar, and S. Shakkottai, "Avoiding interruptions a QoE reliability function for streaming media applications," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 5, pp. 1064–1074, May 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5753570
6. M. Kim, J. Cloud, A. Parandehgheibi, L. Urbina, K. Fouli, D. Leith, and M. M´edard, "Network coded TCP (CTCP)," *arXiv preprint arXiv*: 1212.2291, 2012.
7. X. Zhu and B. Girod, "Video streaming over wireless networks," *Proceedings of the European Signal Processing Conference*, EUSIPCO- 07, Poznan, Poland, 2007.
8. H. Wang, J. Liang, and C. J. Kuo, "Overview of robust video streaming with network," *Journal of Information Hiding and Multimedia Signal Processing*, pp. 36–50, 2010.
9. D. Lee, B. E. Carpenter, and N. Brownlee, "Media streaming observations : trends in UDP to TCP ratio," *International Journal on Advancements in Systems and Measurements*, vol. 3, no. 3, pp. 147–162, 2010.
10. V. Jacobson, "Congestion avoidance and control," Proceedings of SIGCOMM' *88, Stanford, CA, no. 60*, pp. 1–25, August 1988.
11. K. X. Ye Tian and N. Ansari, "TCP in wireless environments: problems and solutions," *IEEE Radio Communications*, pp. 27–32, March 2005.
12. M. Ghobadi and M. Mathis, "Trickle : rate limiting YouTube video streaming," *Proceedings of the USENIX Annual Technical Conference* (ATC), p. 6, 2012.
13. P. S. David and A. 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)*, pp. 224–233, Jan. 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4554414
14. J. K. Sundararajan, D. Shah, M. Medard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets TCP: theory and implementation," *Proceedings of the IEEE, vol. 99, no. 3, pp. 490–512, Mar. 2011.* [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5688180
15. M. Reisslein and F. H. Fitzek, "MPEG4 and H.263 video traces for network performance evaluation," *IEEE Network, no.* December, pp. 40–54, Nov/Dec 2001.
16. G. Pei and T. R. Henderson, "Validation of OFDM error rate model in ns-3," *Boeing Research Technology*, pp. 1–15, 2010.
17. Y. Cai, S. Jiang, Q. Guan, and F. Yu, "Decoupling congestion control from TCP (semi-TCP) for multi-hop wireless networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, p. 149, Jun. 2013. [Online]. Available: http://jwcn.eurasipjournals.com/content/2013/1/149
18. V. Paxson and M. Allman, "Computing TCP's retransmission timer," *RFC 2988*, 2000.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

19. I. Grigorik, "Latency: the new web performance bottleneck," *last accessed: September 2013.* [Online]. Available: http://www.igvita.com/2012/07/19/latency-the-newweb- performance-bottleneck/
20. P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *no. May, pp. 1–16, 2005.*
21. M. Belshe and R. Peon, "SPDY protocol," *last accessed: January 2013. [Online].* Available: http://tools.ietf. orglhtml/draft-mbelshe-httpbisspdy- OO
22. L. Keller, "Ncutilsc," http://code.google.com/p/ncutils/.
23. "Network simulator 3," http://www.nsnam.org/ns-3-*dev, accessed: 2010- 09-30.*
24. I. Gloice Dean Works, "Analysis of video latency in uav," *Ph.D. dissertation, Auburn University,* Auburn, Alabama, 2008.
25. M. Claypool and J. Tanner, "The effect of jitter on the perceptual quality of video," *Proceedings of ACM Multimedia, 1999.*
26. G. Liang and B. 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, p. 6, 2006. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1185373.1185382
27. K. Nichols and V. Jacobson, "Controlling queue delay," *Proceedings of ACM Queue, 2012.*