International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 24 (2017) pp. 14370-14376
© Research India Publications. http://www.ripublication.com

# Design and Performance Evaluation of TCP Performance Enhancement Algorithm with Machine Learning in Wireless Environments

**[1]An Kyu Hwang, Jae Yong Lee[2]* and Byung Chul Kim[3]**

*Dept. of Infocomm Eng., Chungnam National University, Daejeon, 305-764, Korea.*

*[1]Orcid: 0000-0002-5048-1442*

**Abstract**

In the integrated environment of wired and wireless networks, packet losses can be caused by various issues such as congestion and wireless packet loss. The existing TCP senders treat the packet losses as the loss of packets in the transmission queue which are caused by network congestion, resulting to the TCP performance degradation. In this paper, the network congestion and wireless packet loss are discriminated based on machine learning algorithm with one hidden layer. If the estimated result is decided as packet loss from congestion, then congestion window and ssthresh is reduced to half, otherwise, those values are kept if it is decided as a wireless error. The proposed algorithm uses basic TCP NewReno, but in case of packet loss, it modifies congestion control schemes based on prelearned machine learning algorithm. The simulation results show the improved TCP performance as compared with various existing TCP algorithms.

**Keywords:** Wireless TCP, Congestion Control, Machine Learning

## INTRODUCTION

The Internet currently provides diverse data services including multimedia video, and the usage of the wireless network has been greatly increased due to the changes in various devices such as smart-phone and tablet PC. Transmission Control Protocol (TCP) is the most widely used transmission protocol. It uses a sliding window in order to control flow and congestion in data transmission between devices. It is also a connection-oriented protocol, which guarantees the data transmission by re-transmitting a damaged segment. The initially proposed TCP congestion control scheme [1] recognizes all the packet loss in the network as a congestion and reduces the size of the window based on congestion control. It is only adequate for a wired network with low Bit Error Rate (BER).

Various works on TCP have been carried out. Early Tahoe used Slow-Start and fast retransmission techniques and TCP NewReno speeded up the TCP recovery by applying the fast recovery technique. In addition, many works have been conducted to improve TCP performance in the wireless networks. I-TCP [2], M-TCP [3], Snoop [3], and Freeze-TCP [5] were efforts to improve its performance by separating wired and wireless connections (Split Connection Approach). TCP Vegas [12], TCP Veno [6], and TCP Westwood [7] have been suggested to improve the performance by measuring the end-to-end available bandwidth.

In this paper, we predict the cause of packet loss through pre-learned machine learning algorithm when fast retransmissions occur due to packet loss at the sender side. This study proposes to carry out the conventional Additive Increase Multiplicative Decrease (AIMD) [9] algorithm when the prediction made by the machine-learning algorithm indicates that the packet loss is caused by congestion, but it maintains the existing CWnd when it is the loss due to wireless error. The proposed scheme improves TCP performance by preventing unnecessary decrease of congestion window if the loss is decided as a random loss by machine learning prediction.

The remainder of this paper is organized as follows. Section 2 reviews related works to enhance the performance of TCP in the wireless network and the machine-learning techniques. Section 3 explains the congestion control algorithm using machine learning scheme proposed in this study. Section 4 shows the performance analysis results through simulation. Lastly, Section 5 presents conclusions and future study topics.

## RELATED WORKS

Various works on TCP have been conducted to enhance data transmission performance to cope with rapidly increasing wireless network usage. In this section, we survey prior efforts on TCP congestion control, machine learning techniques and its application to TCP throughput enhancements.

### TCP Vegas [12]/ TCP Veno [6] Algorithm

TCP Vegas [12] is an algorithm that predicts the amount of data that can be transmitted based on RTT and adjusts CWnd. The sender measures the RTT whenever it receives ACK and calculates the residual amount of the buffer based on the measured RTT to regulate the CWnd. Moreover, it predicts the amount of data that is buffered in the network based on the lowest and latest RTT values. The difference between the two values is the Diff value, which is used to determine the congestion status of the network.

TCP Veno [6] uses the 'Diff' value, similar to the TCP Vegas algorithm, and it increases CWnd differently in the slow start interval for each RTT as shown in (1) depending on Diff and $\beta$ values. In the congestion avoidance interval, the CWnd is linearly, increased differently as show in (1). Moreover, it determines the cause of packet loss by comparing Diff value and β in the fast retransmission and resets CWnd as shown in (2). If it is determined as congestion because of the large Diff value, it makes the ssthreshold to the half of CWnd. When the Diff value is small, it makes the ssthreshold to the 4/5 of CWnd.

$$CWnd = \begin{cases} CWnd + 1 & , \ if\ Diff < \beta \\ CWnd + 0.5 & , \ otherwise \end{cases} \qquad (1)$$

$$ssthresh = \begin{cases} CWnd \times (4/5) & , \ if\ Diff < \beta \\ CWnd/2 & , \ otherwise \end{cases} \qquad (2)$$

TCP Vegas and TCP Veno have the advantage of regulating transmission rate by determining the status of the network. But, they are not suitable when network situation changes abruptly and multiple flows compete. Moreover, the throughput of TCP Vegas is inferior to TCP NewReno when they share bottleneck link. TCP Veno, on the other hand, has difficulty in choosing an appropriate $\beta$ value under various network environments and its performance is highly affected by $\beta$ value.

**TCP Westwood [7] / Westwood+ [8] algorithm**

TCP Westwood (TCPW) is a congestion control algorithm based on TCP NewReno protocol stack by using the end-to-end bandwidth estimation. TCPW recalculates the bandwidth estimation (BWE) when a sender receives ACK. When there is three duplicated ACKs or timeout, it uses a congestion control algorithm such as (3) and (4).

$$\begin{aligned} &Three\ duplicated\ ACKs: \\ &ssThresh = (BWE \times RTT_{min})/Seg_{size} \\ &CWnd = ssThresh \end{aligned} \qquad (3)$$

$$\begin{aligned} &Time\ out: \\ &ssThresh = (BWE \times RTT_{min})/Seg_{size} \\ &CWnd = 1 \end{aligned} \qquad (4)$$

TCP Westwood+ (TCPW+) is an advanced algorithm of TCPW. It was proposed to overcome the shortfall that the bandwidth estimation algorithm of TCPW does not operate normally under ACK compression environment and reverse traffic condition. TCPW+ offers higher performance than TCP NewReno and is fair for allocating bandwidth [8]. But, TCPW shows inaccurate estimation results when network conditions change abruptly a lot and burst error occurs.

**Machine learning technique**

Machine learning is to build a learning model based on the vast amount of data instead of outputting results by a program. It means to make an algorithm so that a computer can analyze data on its own and find out a specific pattern. Machine learning has been studied extensively in many games such as quiz, chess, and go game. Currently, studies are conducted in various fields including security, self-driving, financial transaction, image recognition, and service business for providing contents suited to customers' tastes. However, machine learning studies are at the early stage in the network field. In this paper, we use machine learning method to distinguish the losses caused by network congestion and wireless random error, and this section outlines the perceptron concept used in our machine learning technique.

**Single-layer Perceptron**

Perceptron can be divided into a single-layer perceptron that can solve the linearly separable problem by using an artificial neural network and a multilayer perceptron that can solve the problem that is not linearly separable in the 2D and above. The single-layer perceptron is a model that converts multiple inputs into one binary output as shown in Fig. 1. The weight (ω) is calculated according to the importance that affects the output among the input vector values (X). The net value indicates a sum of multiplications of input values and weight. $f$ is an activation function that outputs 1 or 0 after comparing the net value with the threshold value. The single layer perceptron can be divided into an input layer and an output layer and the net value of the neuron is calculated by (5). The activation function is based on a linear function, a step function, or a sigmoid function. Equation (6) shows a step function that outputs 1 if the net value is larger than the threshold value and outputs 0, otherwise.

$$net = \sum_{i}^{n} W_i X_i + W_0 X_0 \qquad (5)$$

$$f(net) = \begin{cases} 1 , & if\ net \geq threshold \\ 0 , & otherwise \end{cases} \qquad (6)$$

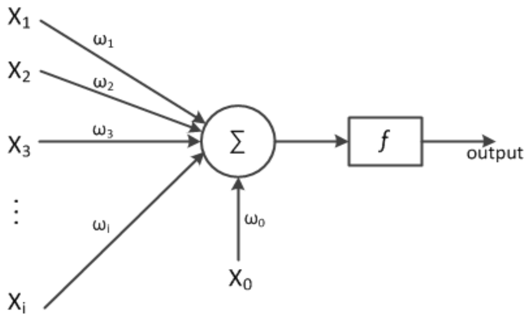

**Figure 1:** Structure of perceptron

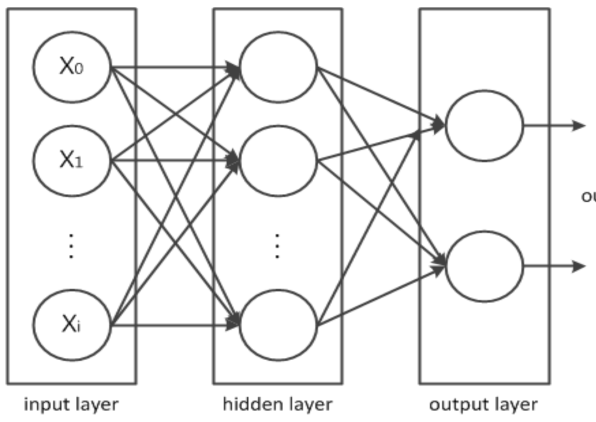

**Figure 2:** Architectureal graph of a multilayer perceptron with one hidden layer

## Multi-layer Perceptron (MLP) [15][16]

The single-layer perceptron can only handle linearly separable problems and it cannot solve difficult questions that cannot be presented by a straight line on a 2D plane. A MLP structure is shown in Fig. 2 and it is a neural network with one or more hidden layer(s). It can overcome the shortfall of the single-layer perceptron by making the input and output characteristics of the hidden layer nonlinear.

In the single-layer perceptron, the weight is learned by using the difference between the target value and the actual output value. However, it is not possible to define the target value of the hidden layer in the multilayer perceptron. Therefore, the multilayer perceptron controls the weight of a hidden layer in order to reduce the error between output values of an output layer and the target value. This series of processes is called backpropagation because it is done in the opposite direction of the processing direction of the neural network.

MLP is composed of a structure that delivers the output value transmitted from the input layer to a hidden layer and the output values of a hidden layer are delivered to another hidden layer. The net value calculation of each neuron is shown in (7). The activation uses a sigmoid function (8). The sigmoid function has values only between 0 and 1. The decision field is made of the gentle curve and the value abruptly increases at the midpoint. Therefore, it is possible to do neural network learning by using the gradient descent.

$$net = \sum_{i}^{n} W_i X_i \tag{7}$$

$$f(net) = \frac{1}{1 + e^{-net}} \tag{8}$$

## Previous works applying machine learning scheme in TCP Protocol Enhancement

Remy [10][11] proposed a new congestion control algorithm by using machine learning. Remy used ACK interval, the ratio of recent RTT and minimum RTT, and data transmission time for learning and regulated congestion by adjusting CWnd and packet transmission interval through output value. Remy did not take into account the performance degradation in the untrained environment and the parameters for wireless random errors in the wireless environment.

Moreover, MPTCP-ML [17] scheme is proposed to select a best path in MPTCP [20][21] using machine learning. Additionally, research on increasing TCP performance through RTT prediction [18] and prediction of transmission rate for a broad range of file sizes have been conducted [19].

## Congestion Control Algorithm by Predicting the Cause of Wired and Wireless Packet Loss Using Machine Learning (ML-TCP)

This section proposes ML-TCP that predicts the cause of the packet loss in wired and wireless networks using machine learning and applies it to the congestion control algorithm. ML-TCP operates based on TCP NewReno and applies the prediction results of machine learning when fast retransmission occurs due to packet loss. When the cause of packet loss is predicted as a network congestion from the prediction based on machine learning, it uses TCP NewReno congestion control algorithm. If it is predicted as a wireless random error, it maintains CWnd and ssThreshold values and simply retransmit the lost packet.

The ns-3 simulator [13] was used to train machine learning to be used in ML-TCP. The network configuration is shown in Fig. 3 and the queue sizes of nodes N1 and N2 were set to 100. RTTmin was set to 60ms. Nodes N4 and N6 randomly generate UDP traffic to cause a loss due to congestion at node N1. Additionally, the simulation was performed to generate a wireless error in the wireless link between nodes N2 and N3. The cause of packet loss was analyzed by analyzing the Wireshark [14] output data provided by ns-3. The recently received ACK interval, the last RTT, the minimum RTT, and the number of packets (P(c)) that are sent but not received ACK were chosen as parameters for the input layer and used for MLP training. Specifically, ML-TCP used RTTmin/P(c) and RTTcurrent/P(c) as input layer parameters for training machine learning and utilized the loss due to network congestion and the information loss due to wireless error, which were analyzed through Wireshark analysis, as output layer results for training.

The ML-TCP was trained with 100 fast retransmission events extracted from TCP NewReno with machine learning. The learning accuracy rate for 100 data was 97%. The trained ML-TCP was verified by using the other 125 retransmission events and the verification results showed that it accurately predicted the cause of packet loss with 96% of accuracy. ML-TCP can be applied simply by adding a packet loss decision prediction algorithm to the FastRetransmit module that processes upon receiving three duplicated ACKs based on TCP NewReno.

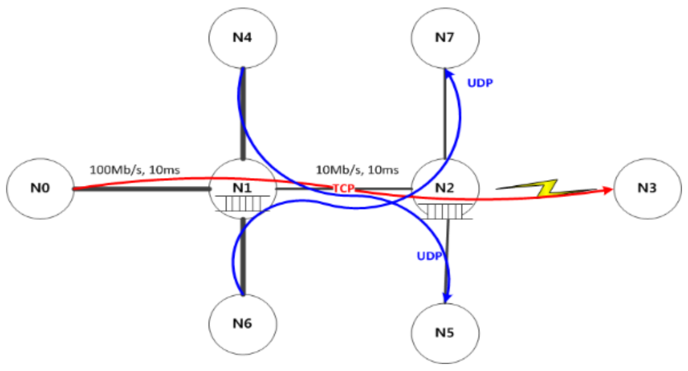

**Figure 3:** Simulation topology for machine learning

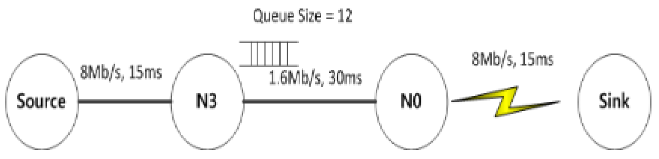

**Figure 4:** Simulation topology

### Simulation and Performance Evaluation

This section compares and analyzes the performance of conventional TCP NewReno, TCP Veno, and the proposed ML-TCP in the wireless network environment by using ns-3 simulation. The simulated network model is shown in Fig. 4, and simulation was conducted with only increasing the wireless packet loss rate between node N0 and Sink node in the background traffic-free environment.

**Table 1:** Confusion matrix of TCP Veno and ML-TCP

| Actual | | Pridicted $10^{-5}$ RL | $10^{-5}$ CL | $10^{-4}$ RL | $10^{-4}$ CL | $10^{-3}$ RL | $10^{-3}$ CL | $10^{-2}$ RL | $10^{-2}$ CL | $10^{-1}$ RL | $10^{-1}$ CL |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Veno ($\beta$=5) | RL | 0 | 0 | 0 | 7 | 2 | 102 | 50 | 732 | 5207 | 15 |
| | CL | 0 | 54 | 0 | 49 | 0 | 12 | 0 | 0 | 0 | 0 |
| ML-TCP | RL | 0 | 0 | 9 | 0 | 94 | 8 | 603 | 1 | 4556 | 0 |
| | CL | 1 | 53 | 0 | 53 | 0 | 42 | 0 | 0 | 0 | 0 |

Table 1 shows the comparison between the predicted and actual causes of packet loss for TCP Veno ($\beta$=5) and ML-TCP. TCP Veno accurately predicts in the environment with little wireless random loss and the environment with frequent wireless loss. On the other hand, it shows lower accuracy rate when wireless packet loss rate increases from $10^{-3}$ to $10^{-2}$ and mispredicts the loss due to wireless random loss as the loss due to network congestion. This result shows that the throughput of TCP Veno depends on $\beta$ and random loss rate. Also, the simulation results show that ML-TCP has higher accuracy for predicting packet loss cause in all environments.

**Table 2:** Confusion matrix of TCP Veno by $\beta$

| Actual | | Pridicted $10^{-5}$ RL | $10^{-5}$ CL | $10^{-4}$ RL | $10^{-4}$ CL | $10^{-3}$ RL | $10^{-3}$ CL | $10^{-2}$ RL | $10^{-2}$ CL | $10^{-1}$ RL | $10^{-1}$ CL |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Veno ($\beta$=5) | RL | 0 | 0 | 0 | 7 | 2 | 102 | 50 | 732 | 5207 | 15 |
| | CL | 0 | 54 | 0 | 49 | 0 | 12 | 0 | 0 | 0 | 0 |
| Veno ($\beta$=10) | RL | 0 | 0 | 0 | 6 | 3 | 89 | 468 | 322 | 5226 | 0 |
| | CL | 0 | 54 | 0 | 49 | 0 | 16 | 0 | 0 | 0 | 0 |
| Veno ($\beta$=15) | RL | 0 | 0 | 0 | 6 | 19 | 81 | 746 | 46 | 5226 | 0 |
| | CL | 0 | 54 | 0 | 49 | 0 | 4 | 0 | 0 | 0 | 0 |
| Veno ($\beta$=20) | RL | 0 | 0 | 0 | 8 | 14 | 83 | 787 | 3 | 5226 | 0 |
| | CL | 1 | 53 | 1 | 48 | 0 | 14 | 0 | 0 | 0 | 0 |
| Veno ($\beta$=25) | RL | 0 | 0 | 4 | 4 | 428 | 0 | 3211 | 0 | 5226 | 0 |
| | CL | 54 | 0 | 48 | 0 | 7 | 0 | 0 | 0 | 0 | 0 |

Table 2 shows the analysis results of the packet loss prediction for varying $\beta$ values for TCP Veno. When there is a large number of Random Loss(RL) in the environment with a wireless packet loss rate between $10^{-3}$ and $10^{-2}$, it misjudges it as Congestion Loss (CL) when $\beta$ is low and it accurately determines it as RL when $\beta$ is high. However, if $\beta$ is high in the environment with low wireless packet loss rate, it tends to misjudge CL as RL, which is a big shortcoming of TCP Veno.

Therefore, TCP Veno needs to choose a $\beta$ value that provides the optimal performance depending on the network environment. However, the machine learning-based algorithm to predict packet loss cause has an advantage that it is not much affected by the network situation.

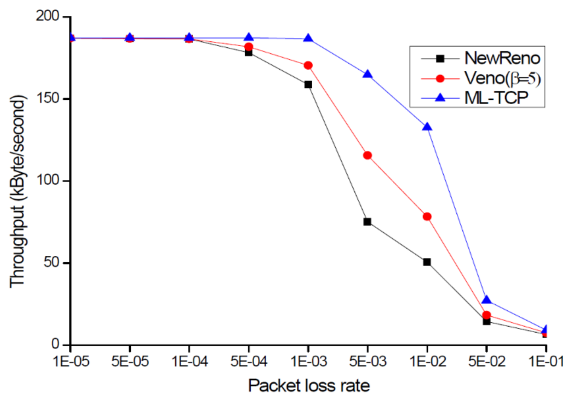

**Figure 5:** Throughput for varying wireless packet loss rate

Fig. 5 shows the throughput of each TCP algorithm for varying wireless loss rate. TCP NewReno, TCP Veno, and ML-TCP show similar performance in the environment without wireless loss. However, the simulation results show that TCP NewReno has the lowest performance in higher wireless random loss case because TCP NewReno decides all the loss came from network congestion and reduces CWnd. As can be shown in Table 1, TCP Veno does not significantly improve the throughput because RL is incorrectly judged to CL for the packet loss rate of $10^{-3}$ to $10^{-2}$. When wireless packet error rate increases from $10^{-4}$ to $10^{-2}$, the amount of transmission for TCP NewReno and TCP Veno decreases to 27% and 42% level of no random error result respectively, while it decreases to 70% for ML-TCP. The result indicates that ML-TCP improves the performance.

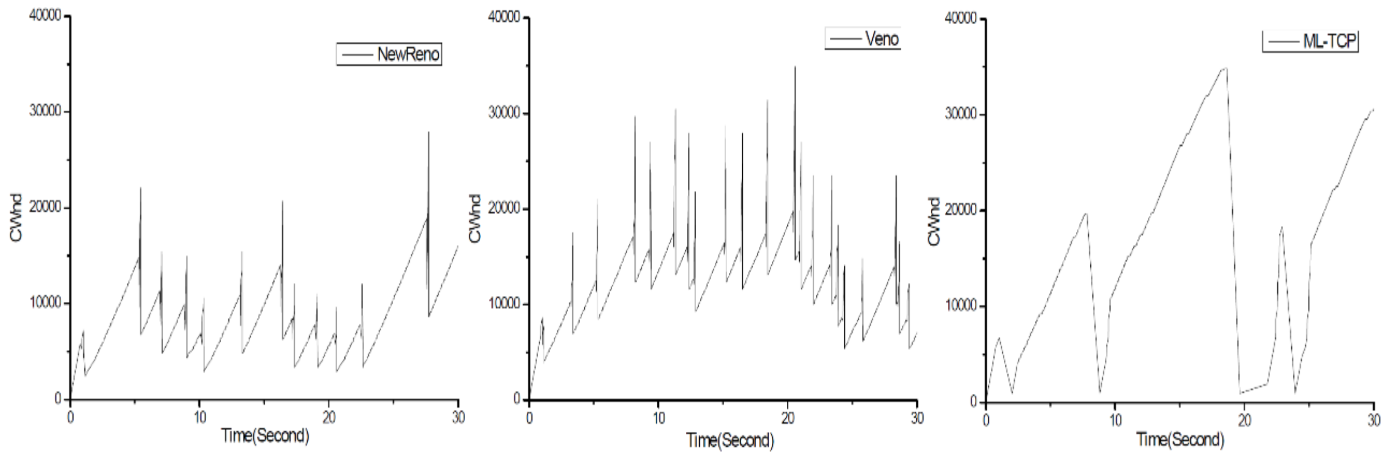

**Figure 6:** Congestion window size vs. time

Fig. 6 shows the changes in CWnd for each protocol when wireless packet loss rate is $10^{-2}$. TCP Veno generally maintains a higher CWnd than TCP NewReno, thereby exhibiting a higher amount of transmission. The result of simulation shows that ML-TCP discovered more RL per hour due to high CWnd, and thus it generated less CWnd reduction because the prediction accuracy rate for RL is high.

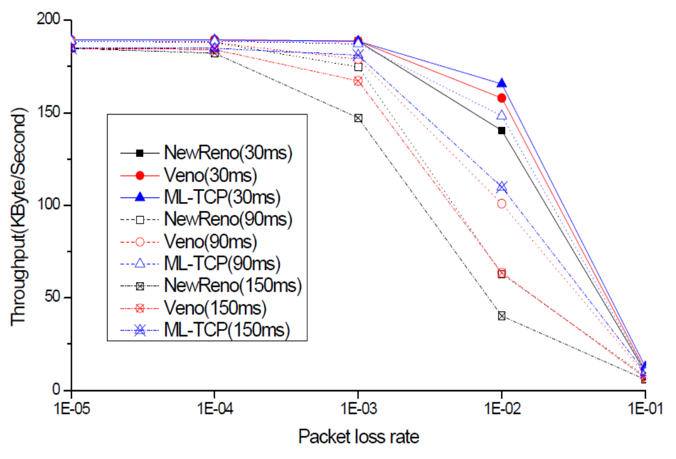

**Figure 7:** Throughput for various RTT

Fig. 7 shows a comparison of the TCP throughput with increasing RTT to 30, 90, and 150ms in the same environment as in Fig. 4. ML-TCP shows a significant increase in transmission rate in environments with high RTT and RL compared to existing TCP.

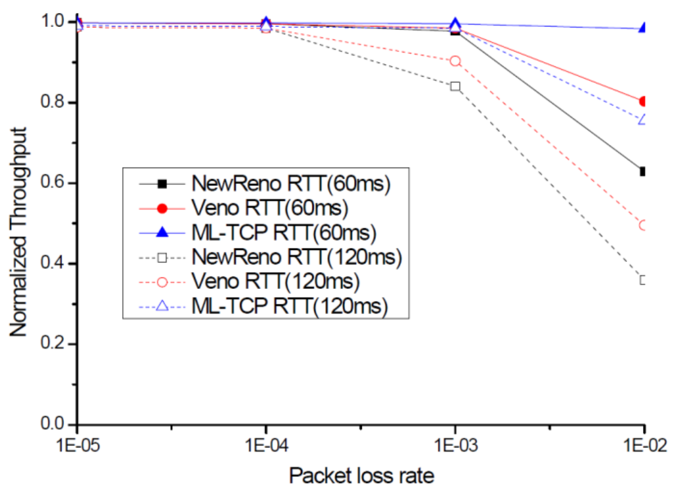

**Figure 8:** Normalized Throughput for various RTT

Fig. 8 shows a graph of the normalized throughput on the basis of 30ms when RTT increases to 60 and 120ms. It is confirmed that the three protocols do not show a large difference in the transmission rate when the wireless packet loss rate is low because most of loss occurs from network congestion. On the other hand, when large packet losses were generated due to wireless error, the amount of transmission differed a lot by RTT. In an environment with a wireless packet loss rate of $10^{-2}$, TCP NewReno shows that as the RTT increases from 30ms to 120ms, the throughput dropped to 35% and the TCP Veno dropped to 50%. On the other hand, ML-TCP shows that the transmission amount is maintained at 75% in the same environment through simulation.

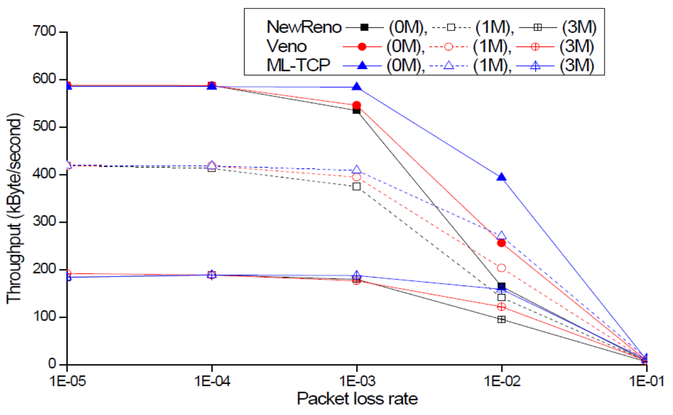

**Figure 9:** Throughput for Background Traffic

Fig. 9 is the TCP throughput comparison with increasing UDP background traffic in a network model similar to Fig. 3. The RTT parameter is set to 40ms and the minimum link bandwidth as 5Mb/s. All the protocols show the same throughput at 4.7Mbps, 3.3Mbps, and 1.3Mbps when background traffic is generated at 0Mbps, 1Mbps, and 3Mbps, respectively, in the absence of wireless errors. But, ML-TCP shows 18 ~ 50% better performance than TCP Veno when wireless packet loss rate is $10^{-2}$.

## CONCLUSIONS

In the integrated environment of wired and wireless networks, the TCP performance is degraded because it considers random losses in the wireless link as network congestion and unnecessarily reduces the transmission rate. In this paper, we propose an algorithm that applies the prediction of machine learning when packet loss occurs after learning packet loss caused by network congestion loss or wireless error using a machine learning algorithm. Proposed algorithm (ML-TCP) is based on TCP but apply different congestion control according to prediction of the cause of loss. If packet loss is predicted as congestion loss, the congestion window is reduced to half, but predicted as random loss, it is maintained. The proposed algorithm can be easily applied to TCP NewReno, and it shows by simulations that the performance is improved.

As a future work, we will apply the proposed algorithm to real Linux machine and evaluate the performance of TCP and fairness between TCP. We will also apply an improved deep learning algorithm that can learn the network situation at the beginning part of a TCP connection in real time without using the previously learned information.

## ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government. (NRF-2017R1D1A1B03028766), (NRF-2015R1D1A3A01018044)

## REFERENCES

[1] V. Jacobson, "Congestion Avoidance and Control", Proceedings of ACM SIGCOMM '88, Stanford, Aug, 1988.

[2] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts", Proceedings of the 15th International Conference on Distributed Computing Systems, Vancouver, Canada, Jun 1995.

[3] Kevin Brown, Suresh Singh, "M-TCP: TCP for Mobile Cellular Netwroks", ACM Computer Communications Review (CCR), vol. 21, no.5, 1997.

[4] H. Balakrishnan, V.N. Padmanabhan, S. Seshan,and Randy Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", ACM SIGCOMM'96, pp. 256-269, Palo Alto, CA, Aug, 1996.

[5] Tom Goff, James Moronski, Vipul Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments", in Proceedings of IEEE INFOCOM 2000, Israel, Mar, 2000.

[6] C. P. Fu and S. C. Liew, "TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks," IEEE Journal on Selected Areas in Communications, pp. 216-228, Feb. 2003.

[7] Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M., Wang, R. TCP Westwood: End-to-End Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks, in Proceedings of ACM Mobicom 2001, Rome, Italy, Jul 2001.

[8] Grieco, L. A., and Mascolo, S., “Westwood TCP and easy RED to improve Fairness in High Speed Networks”, in Proceedings of IFIP/IEEE Seventh International Workshop on Protocols for High Speed Networks, PfHSN02, Berlin, Germany, Apr, 2002.

[9] D. Chiu and R. Jain, “Analysis of the increase/decrease algorithms for congestion avoidance in computer networks,” J. Comput. Networks, vol. 17, no. 1, pp. 1–14, Jun, 1989.

[10] K. Winstein and H. Balakrishnan. “TCP ex Machina: Computer-Generated Congestion Control”. In SIGCOMM, Hong Kong, August 2013.

[11] A. Sivaraman, K. Winstein, P. Thaker, and H. Balakrishnan. An experimental study of the learnability of congestion control. In SIGCOMM, 2014.

[12] Brakmo, L. S., O’Malley, S.W., and Peterson, L. “TCP Vegas: End-toend congestion avoidance on a global Internet”. IEEE Journal on Selected Areas in Communications (JSAC), 13(8), (1995), 1465-1480.

[13] ns-3 network simulator, http://nsnam.org/.

[14] Wireshark, https://www.wireshark.org/.

[15] P. J. Werbos. Advanced forecasting methods for global crisis warning and models of intelligence. General Systems Yearbook, 22:25{38, 1977.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol.1: Foundations. Bradford Books/MIT Press, Cambridge, MA, 1986.

[17] Jonghwan Chung, Dahyeon Han, Jiyoung Kim, Chong-kwon Kim."Machine learning based path management for mobile devices over MPTCP". 2017 IEEE International Conference on Big Data and Smart Computing, Feb, 2017.

[18] B. A. A. Nunes, K. Veenstra, W. Ballenthin, S. Lukin, and K. Obraczka. “A Machine Learning Framework for TCP Round-trip Time Estimation”, EURASIP Journal on Wireless Communications and Networking, 2014(1):1–22, 2014.

[19] M Mirza, M Sommers, P Barford, X Zhu, “A machine learning approach to TCP throughput prediction”, SIGMETRICS Perform. Eval. Rev.35, 97–108, Jun, 2007.

[20] A.Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, “TCP Extension for Multipath Operation with Multiple Address”, IETF RFC 6824, 2013.

[21] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, et al.,“ How hard can it be designing and implementing a deployable multipath TCP”, Proceedings of the 9th USENIX conference on Networked SystemsDesign and Implementation, 2012.MinSeok Kim, SungRyul Kim, Jeonghyun Kim, and Younghwan Yoo, “Design and Implementation of MAC Protocol for SmartGrid HAN Environment", proceedings of the IEEE CIT 2011, Sep. 2011.