Towards a Distributed TCP Improvement through Individual Contention Control in Wireless Networks

Hengheng Xie[1], Azzedine Boukerche[1], Robson De Grande[1], Fei Richard Yu[2]
[1]PARADISE Research Lab,
School of Electrical Engineering and Computer Science, University of Ottawa, Canada
[2]School of Information Technology, Carleton University

***Abstract*—TCP suffers degradation in wireless networks, which is caused by the improper, static definitions on the lower layers. In order to improve the TCP performance in wireless networks, the strategy of the lower layer should be reconsidered. In this paper, the TCP transmission is flattened in order to combine the TCP segment transmission and TCP Acknowledgement (ACK) transmission into one transmission. The performance of TCP is also analyzed, in order to find out the corresponding parameters affecting the TCP throughput. Based on the analysis, contention window size shows as one of the parameters that greatly affects the TCP performance. A discrete-time Markov decision process is adopted in order to solve the TCP throughput maximization. Based on the analysis, a TCP-distributed algorithm is proposed. Several simulations are conducted to verify the improvements of TCP-distributed by comparing both TCP Reno and TCP Vegas. Simulation results show that TCP-distributed can perform better than the two TCP variants, and it also limits the delay in an acceptable range.**

I. Introduction

Due to its increasing popularity, wireless networks have been extensively used as the principal mean for most of the current applications. However, wireless networks present some differences when compared with wired networks, causing communication issues to applications and protocols. TransmissionControl Protocol (TCP) [1] is one of the protocols which is popular in wired networks, but it suffers considerable performance degradation in wireless networks. Congestion avoidance is an important feature of TCP. Each packet loss leads to the decrement of the congestion window size in TCP congestion mechanism, no matter the cause: time out or duplicate acknowledgements.This assumption is correct only in the stable network environment where packet loss on the lower layer is rare, such as wired networks. However, the packet loss over the wireless networks is mainly due to contention, fading and interference and not on congestion, as identified in recent works [2].

Because of the degradation of TCP in wireless networks, most researchers recently focus on the optimization of the congestion control mechanism [3], [2], [4], [5], [6], [7]. The main focus of TCP improvement may lay on the queuing management, delay acknowledgement, congestion window adjustment, or packet loss cause identification. These approaches successfully improve the TCP performance in some cases. Many observations also indicate that bursty traffic caused by congestion control mechanisms degrades TCP performance [8]. Some other researches on TCP pacing are mainly centred on the transport layer [9], [10], [11]. However, the improvement in TCP is limited, because the congestion avoidance mechanism of TCP is easily interrupted by the unstable condition of wireless networks. TCP is still unable to confirm the actual cause of segment loss because the lower layer is hidden from TCP. Besides modifying the TCP, other approaches should be adopted to improve the performance of TCP. Besides the improvement on TCP, other researchers attempt to optimize the TCP performance through a cross-layer technique [12], [13], [14], [15], [16], [17]. Since observations support the notion that contention is the main reason for TCP segment loss on wireless network [2], contention avoidance eventually increases TCP performance.

In this paper, a TCP-distributed solution is proposed, which sets the corresponding contention window size for each node along the TCP transmission path. The corresponding contention window size is calculated based on the analysis on the TCP throughput. By combining the TCP segment transmission and TCP ACK transmission in to one single analysis path, the analysis of the TCP throughput can be evaluated in a simple way. A discrete-time Markov decision process is adopted to solve the TCP throughput maximization. Based on the analysis, each node calculates it own contention window size based on the information from one-hop neighbours. Verification simulations are conducted to verify the performance improvement of TCP-distributed.

The remainder of the paper is organized as follows. Section II demonstrates TCP performance estimation, RTT estimation, and the packet loss probability estimation. Section III discusses discrete-time Markov Decision Process to solve the maximization problem. Section IV describes selected simulations in order to emphasize the performance of our proposed system. Finally, Section V summarizes the contribution of this paper and present the future work of our research.

II. TCP Performance Estimation

TCP suffers degradation in wireless networks, mainly caused by TCP ignoring the mechanisms from the lower layer. In order to improve the performance of TCP, such mechanisms

† This work is partially supported by NSERC DIVA Strategic Reserach Network Grant, Canada Research Chairs Program and NSERC Discovery Grant.

from the lower layer should select the most appropriate parameter values. Consequently, the TCP performance should be analyzed carefully to determine the parameter set affecting it. There are several implementations of TCP, and the implementation selected in this research is the TCP Reno, which is one of the most popular TCP variants. A throughput model of TCP Reno has been proposed [18] as shown in Equation 1 and Equation 2. In these equations, $E[RTT]$ denotes the mean number of the Round Trip Time (RTT). b denotes the number of segments each ACK has acknowledged, which is assumed to be 1 in this work. T_0 denotes the initial timeout, p denotes the TCP segment loss probability, and W_{max} denotes the maximum congestion window size. In the mathematical model of TCP Reno, it can be determined that the parameters that represent the effect on the TCP throughput are the RTT and p. Both RTT and p are parameters still observed in TCP. To further investigate the effect of the parameters in the lower layer, both RTT and p should be analyzed.

$$E[B] \approx \frac{1}{E[RTT]\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{3bp/8}\right) p(1+32p^2)} \tag{1}$$

$$B(RTT, p) = \min\left(\frac{W_{max}}{E[RTT]}, E[B]\right) \tag{2}$$

TCP Vegas is another recent popular TCP variant. A throughput model for TCP Vegas has been defined [19], and its throughput estimate is presented in Equation 3, in which n denotes the expected number of consecutive Loss Free Periods (LFP) that occur between one slow start (SS) and another slow start period. Equation 4 presents the calculation of N_{SS2TO}, in which W_0 denotes the average congestion window size during a stable backlog state, and T_{TO_0} denotes the average duration of the rst TO in a TO series. D_{TO} is defined in Equation 5. Based on the model, the factors of the throughput of TCP Vegas are segment loss probability and RTT, which are the same as that of TCP Reno.

$$E[B]_{Vegas} \approx \frac{(n+1)\left(\frac{1-p}{p} + W_0\right) + \frac{2p(2-p)}{(1-p)^2}}{N_{SS2TO}RTT + D_{TO}} \tag{3}$$

$$N_{SS2TO} = 2logW_0 + (n+1)\frac{1-p}{pW_0} + (1 + \frac{n}{4}\frac{W_0}{8}) + \frac{9n}{8} - \frac{11}{4} + \frac{4 - 2^{logW_0}}{W_0} \tag{4}$$

$$D_{TO} = \left((1-p)^2 \sum_{k=1}^{6}((2^k - 64k + 320)(2p - p^2))^{k-1} + \frac{64}{(1-p)^2} - 321\right) T_{TO_0} \tag{5}$$

A normal TCP transmission is presented in the Figure 1. A TCP segment will be forwarded along the selected path hop by hop. We assume that the routing protocol for our research is the Ad hoc On-Demand Distance Vector (AODV). Therefore, the selected path for the Acknowledgement (ACK) is the reverse path of the path for the TCP segment. We also assume that fragmentation of the TCP segment is disabled. Therefore, there is only one frame for each TCP segment. Until it reach the destination, an ACK is sent from the destination node to the source node following the reversed path hop by hop. Any failure on both TCP segment transmissions or ACK transmissions causes the retransmission of the TCP segment from the source node. Therefore, both the path of the TCP segment transmissions and the path of the ACK transmissions should be measured.

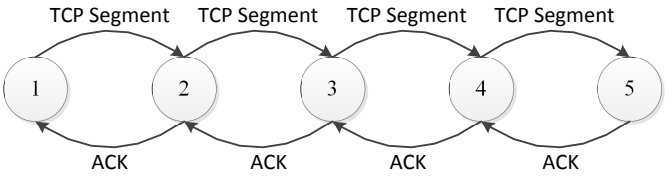

Fig. 1. Normal TCP transmission along a selected path

The TCP transmission is flatted to one transmission path as it is shown in Figure 2. The transmission path is extended by the reverse path of ACK transmission. The TCP segment transmissions and ACK transmissions are considered in one combined path. In this case, TCP segment transmission and ACK transmission can be analysed as one transmission. Any transmission failure will cause the retransmission from the source node, node 1. Along this combined path, different requirements are needed at different parts of the path, in order to improve TCP performance. At the first part of the combined path, fast retransmission is required when transmission failure occurs, and the packet loss rate is not a concern. The reason is that the cost of the retransmission is still low when the transmission failure is close to the source node. A fast retransmission can overwhelm the effect of the high packet loss rate in this case. At the last part of the combined path, a low packet loss rate is required because the cost of the retransmission is very high.

From the analysis, it can be seen that the transmissions at the first part of the combined path are mainly TCP segment transmissions, and the transmission at the last part of the combined path are mainly ACK transmissions. Therefore, the value of the parameters for TCP segment transmissions and the ACK transmissions should be different. This means each node along the path should have two set of parameters: one is for the TCP segment transmission, and the other one is for the ACK transmission in the reverse path. The estimation of RTT and p are discussed in the following sections, and the parameters on lower layer, which affects the RTT and p, are analysed [20].

A. RTT Estimation

In this section, the mean number of RTT is discussed. As shown in Figure 2, the average RTT of a packet transmission along the path is the accumulated transmission time along the

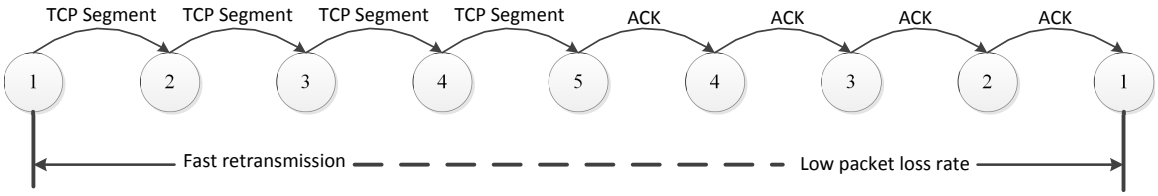

Fig. 2. Flattened TCP transmission along a selected path

combined path. The estimated time of the transmission of each link, $E[T_{data}]$, can be analysed by the following equations. $E[t_q]$ denotes the estimated time to wait in the queue of a packet. $E[t_{data}]$ stands for the transmission time of a packet over the physical layer, which can be simply calculated from Equation 7. L_{data} represents the size of the packet, and r denotes the transmission rate. It is worthy noting that the size of the TCP segment is larger than the size of an ACK. $E[t_{LLD}]$ stands for the link layer delay of a TCP segment in the current node.

$$E[T_{data}] = E[t_q] + E[t_{data}] + E[t_{LLD}] \tag{6}$$

$$E[t_{data}] = \frac{L_{data}}{r} \tag{7}$$

The delay of the link layer is mainly caused by the delay of the RTS/CTS mechanism, the delay of packet retransmissions and the transmission time of the packet. $\frac{L_{rts}}{r}$ and $\frac{L_{cts}}{r}$ denote the transmission time of RTS and CTS respectively. $\frac{L_{ACK_{LL}}}{r}$ represents the transmission time of the link layer ACK. $Delay_{LL}$ stands for the mean number of delay of each transmission failure in the link layer. $E[N_{rts}]$ denotes the average number of transmissions to successfully deliver a RTS. $E[N_{data}]$ designates the average number of transmissions to successfully deliver a packet. $L_{ACK_{LL}}$ represents the size of the link layer ACK.

$$\begin{aligned} E[t_{LLD}] = \quad & \left(\frac{L_{rts}}{r} + \frac{L_{cts}}{r} + E[delay_{LL}](E[N_{rts}] - 1)) \right. \\ & \left. + E[delay_{LL}](E[N_{data}] - 1)) + \frac{L_{ACK_{LL}}}{r} \right) \end{aligned} \tag{8}$$

In each transmission failure in the link layer, $E[TO_{LL}]$ indicates the time to timeout when an RTS is sent and it is waiting for a CTS. $E[BF]$ represents the average backoff of a failure. $slotTime$ denotes the length of each time slot.

$$E[delay_{LL}] = E[BF]slotTime + E[TO_{LL}] \tag{9}$$

The timeout of each RTT transmission is calculated by accumulating the time to transmit the RTS, the time to transmit the CTS, and the maximum estimation of network propagation delay $PDelay_{max}$.

$$E[TO_{LL}] = \frac{L_{rts}}{r} + \frac{L_{cts}}{r} + 2 * PDelay_{Max} \tag{10}$$

The average number of transmission to successfully transmit the RTS and a packet is presented in Equation 11 and Equation 12. $N_{rts_{max}}$ denotes the maximum attempts of the RTS transmission, and $N_{data_{max}}$ indicates the maximum attempts of the packet transmission.

$$E[N_{rts}] = \frac{1 - F_{rts}^{(N_{rts_{max}}+1)}}{1 - F_{rts}} \tag{11}$$

$$E[N_{data}] = \frac{1 - F_e^{(N_{data_{max}}+1)}}{1 - F_e} \tag{12}$$

The following equations estimate the time for transmission of each link in the corresponding node. To estimate the total RTT for each TCP segment, Equation 13 presents the estimated RTT at node i along the path is equal to the accumulation of the $E[T_{data}]$ of link i multiplying to the N_{tran}, the average number of transmissions, in all the link along the combined path. Equation 14 indicates the how to calculate the $E[RTT]^i$ based on the one-hop information.

$$E[RTT]^i = \sum_{j=i}^{D} E[T_{data}]^j N_{tran}(j) \tag{13}$$

$$\begin{cases} E[RTT]^D = 0 \\ E[RTT]^i = E[T_{data}]^i N_{tran}(i) + E[RTT]^{i+1} \end{cases} \tag{14}$$

To calculate the average number of transmission of each link, the average number of packet N_{pack} is needed, which is shown in Equation 16. D denotes the destination of the combined path. MAX denotes the maximum packet transmission attempts. p_i denotes the transmission rate of link i. The average number of packet that is needed to transmit in a link is also determine by the transmission failure of the following link along the combined path. The reason is that each failure of transmission in TCP at the following link will cause the current link to retransmit the packet. Equation 17 presents the method to calculate the $N_{pack}(i)$ based on one-hop in formation.

$$N_{tran}(i) = \frac{1}{1 - p_i} N_{pack}(i) \tag{15}$$

$$N_{pack}(i) = 1 + \sum_{j=i}^{D} p_j^{MAX} \tag{16}$$

$$\begin{cases} N_{pack}(D) = 0 \\ N_{pack}(i) = 1 + p_i^{MAX} + (N_{pack}(i+1) - 1) \end{cases} \quad (17)$$

Based on the analysis, it is shown that $E[BF]$ is the parameter that affects the transmission time of each link. Eventually, the RTT of a TCP segment is affected by the $E[BF]$, which is actually decided by the contention window size. Even though the data frame size is another parameter affecting the RTT, the data frame is bounded to Maximum Transmission Unit (MTU). Therefore, the data frame size is not considered in this research.

B. TCP Segment and Link Layer Packet Loss Probability

The packet loss probability, p, is also affected by the $E[BF]$. The equations are indicated as follows. F_{rts} denotes the loss probability of a RTS transmission. F_{BER} represents the failure probability caused by the Bit Error Rate (BER). F_c indicates the packet loss probability caused by the collision.

$$F_{rts} = F_{BER} + F_c \quad (18)$$

$$\begin{cases} F_{BER} = 1 - (1 - BER)^{L_{rts}} \\ F_c = \frac{1}{E[BF]} \end{cases} \quad (19)$$

F_e stands for the failure probability of the frame transmission in the MAC layer. If the number of RTS failures is over the maximum number of transmission attempts, the frame transmission is counted as fail as well. If the number of frame transmission failures is over the maximum attempts, the TCP transmission fails as well.

$$F_e = 1 - (1 - BER)^{L_{data}} + F_{rts}^{N_{rts_{max}}+1} \quad (20)$$

$$p = 1 - (1 - F_e^{N_{data_{max}}}) \quad (21)$$

Based on the analysis, the packet loss probability is also related to the $E[BF]$ and the data frame size. As mentioned in the previous section, the data frame size is not considered in this research. Only $E[BF]$ is investigated in this work. Based on the analysis from this section, the contention window size is the parameter that is important to the TCP throughput. To maximize the TCP throughput, the suitable contention window size needs to be properly determined.

III. Discrete Markov Decision Process

In this section, a discrete Markov Decision Process (MDP) [21] is integrated in this research to determine the suitable contention window size. The state space of this MDP is $\{s(t), b(t)\}$ as shown in Figure 3, in which $b(t)$ is the stochastic process representing the backoff time counter for a given stage $s(t)$.

The transitions are presented in Equation 22. The first term of the equation indicates that the backoff timer is reduced in each slot time. The second term indicates that if a packet is successfully delivered, the backoff stage is reset to 0 and a backoff timer is chosen from $(0, CW_0 - 1)$. The third term of the equation defines that a unsuccessful delivery causes a backoff stage increase and the backoff timer is set from $(0, W_i - 1)$. If the backoff stage reaches value m, it stops increasing.

$$\begin{cases} P\{i,k|i,k+1\} = 1 & k \in (0, W_i - 2) \quad i \in (0, m) \\ P\{0,k|i,0\} = (1-p)/W_0 & k \in (0, W_0 - 1) \quad i \in (0, m) \\ P\{i,k|i-1,0\} = p/W_i & k \in (0, W_i - 1) \quad i \in (1, m) \\ P\{m,k|m,0\} = p/W_m & k \in (0, W_m - 1) \end{cases} \quad (22)$$

The contention window size, $a_k = \{a_{cw}(k)\}$, is considered as the action of the model defined in this work. The direct reward of this model is the TCP throughput, which is represented in Equation 23.

$$\begin{aligned} &R_k(a_k) = \max \textstyle\sum_{i \in X} \\ &\textstyle\sum_{j \in X} P(i,j) \sum a_k * [B(RTT, p) + R_{k+1}(a_{k+1})] \end{aligned} \quad (23)$$

IV. Simulations

In this section, verification simulations are conducted in the Network Simulator-2 (NS2). In order to prove the performance of the TCP-distributed, the algorithm is compared to both the normal TCP Reno and TCP Vegas. The simulations are conducted in the chain topology in different network sizes. A FTP source is attached in the source end, which generates infinite traffic. The information of the other node is transmitted by the control messages from one-hop neighbour when the path is discovered or maintained. The time interval for the control message is set to 1 second. Therefore, each node is able to calculate the corresponding contention window size for maximizing the TCP throughput. IEEE802.11a is chosen as the MAC layer protocol.

The first simulation is conducted with TCP Reno. The evaluation function of TCP-distributed is represented in Equation 2. Figure 4 presents the simulation results of the throughput by comparing TCP-distributed and the normal TCP Reno. It can be clearly observed that TCP-distributed can perform better than the normal TCP-Reno by providing extra throughput. This extra throughput is achieve with the TCP-distributed providing different contention control along the path, which meets the requirements of each node based on their position in the path. Based on the analysis in the previous section, the first part of the combined path requires a fast retransmission, and the last part of the path requires a lower segment loss probability when transmitting a TCP segment. The analysis of the delay is presented in Figure 5. The delay of the TCP-distributed is slightly larger than the delay of normal TCP Reno because of the control messages. However, the delay of the TCP-distributed is still in a acceptable range for most of the applications.

Our solution is compared with TCP Vegas under the same simulation scenario in order to verify its performance with different TCP variants. The evaluation function used in TCP

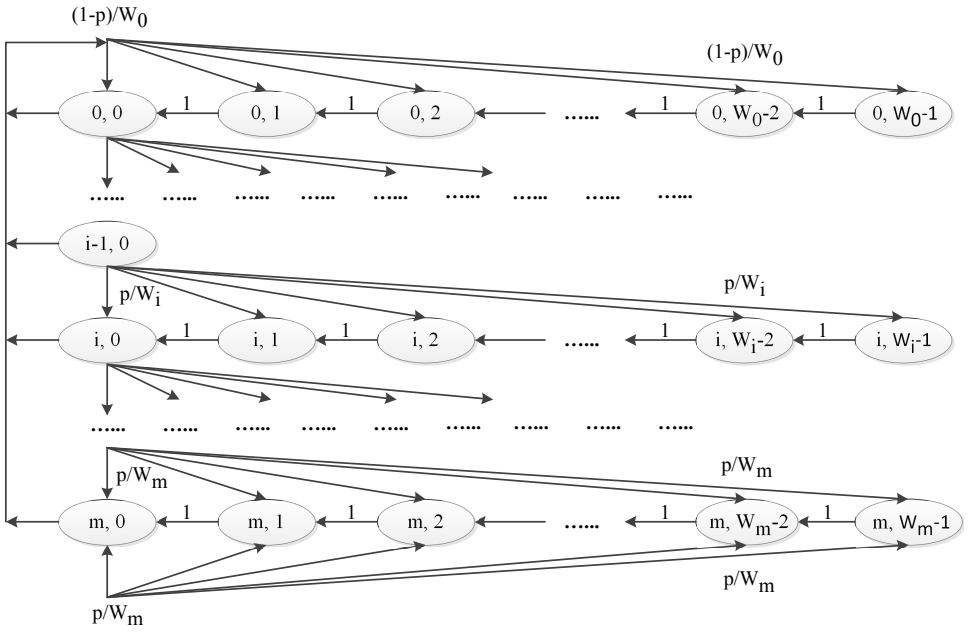

Fig. 3. Discrete Markov Decision Process of MAC layer

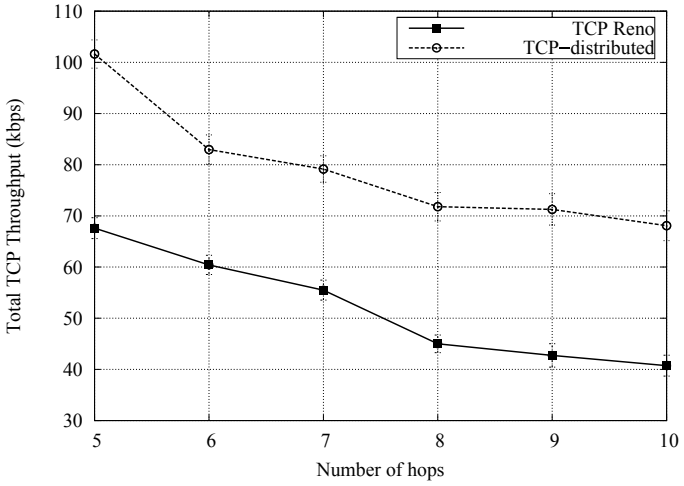

Fig. 4. Throughput of the TCP-Distributed vs normal TCP Reno

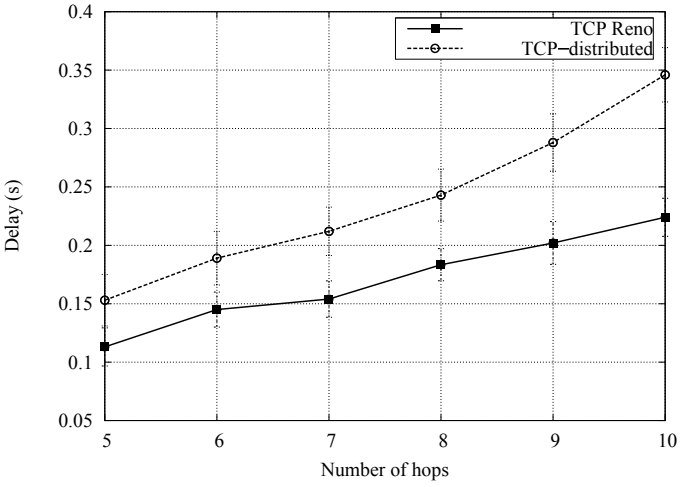

Fig. 5. Delay of the TCP-Distributed vs normal TCP Reno

Vegas is based on the model of Equation 3. Figures 6 and 7 show the results of the simulations. The throughput of the TCP-distributed is better than that of the TCP Vegas, which is caused by correctly setup of the MAC layer parameters of TCP-distributed. However, the improvement is not as high as that of TCP-distributed vs TCP Reno. This smaller gain on throughput is due to the fact that the performance of TCP Vegas is already improved when compared to to TCP Reno. Therefore, the space for improvement is also limited. The delay of TCP Vegas is much lower than that of TCP Reno because TCP Vegas emphasizes on the packet delay as a sign of the congestion, which dramatically reduces the delay of TCP segment delivery. The increment of the delay of TCP-distributed is also limited in a reasonable range.

V. Conclusion

Besides modifying the TCP mechanism, cross-layer approaches comprehend another important method to improve TCP performance. Because the wireless network infrastructure is different from the wired network infrastructure, some assumptions adopted in TCP are improper for wireless communications. TCP lacks the information of the lower layer, which causes the TCP to not choose the best strategy for transmissions. In order to improve the TCP performance, a different transmission strategy in the lower layers is needed. By flattening the TCP transmissions, a combined path is generated, which includes TCP segment transmissions and TCP ACK transmissions. Therefore, rather than analysing the TCP segment and TCP ACK as two transmissions, the analysis

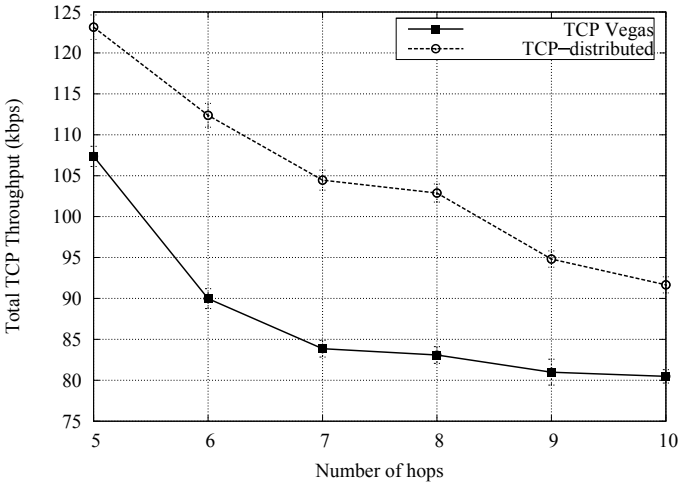

Fig. 6. Throughput of the TCP-Distributed vs normal TCP Vegas

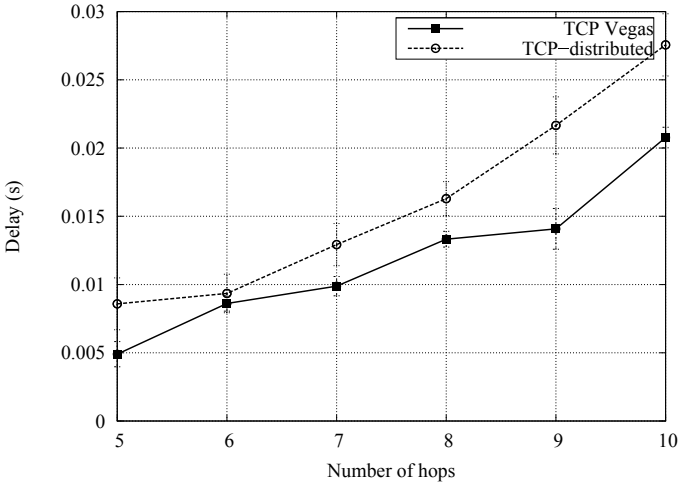

Fig. 7. Delay of the TCP-Distributed vs normal TCP Vegas

on the combined path can treat the two transmissions as one. Based on the analysis, an set of equations are derived to estimate the RTT and packet loss probability, which is used for the evaluation of the TCP throughput. A discrete Markov decision process is adopted to find the corresponding contention window size to the maximize the TCP throughput. TCP-distributed is proposed based on this analysis. Verification simulations are conducted on both TCP Reno and TCP Vegas in order to verify that the TCP-distributed is independent of the TCP variant. The simulation results represent that the TCP-distributed can perform better than both TCP Reno and TCP Vegas in terms of throughput. The delay of TCP-distributed is also limited in a acceptable range. As future work, the TCP-distributed will be applied to the vehicular networks, in which mobility is a major factor to be considered. A more accurate Markov model for IEEE802.11p will be also designed and adopted since IEEE802.11p is designed specially for vehicular networks.

REFERENCES

[1] *Transmission Control Protocol*, IETF RFC Std. 793, 1981.

[2] Z. Fu, H. Luo, P. Zerfos, S. Lu, and L. Zhang, "Mario gerla: The impact of multihop wireless channel on TCP performance," *IEEE Transactions on Mobile Computing*, vol. 4, pp. 209–221, 2005.

[3] G. Marfia and M. Roccetti, "TCP at last: reconsidering TCP's role for wireless entertainment centers at home," *IEEE Transactions on Consumer Electronics*, vol. 56, pp. 2233–2240, Nov. 2010.

[4] H. Seferoglu and A. Markopoulou, "Network coding-aware queue management for TCP flows over coded wireless networks," *Networking, IEEE/ACM Transactions on*, vol. 22, no. 4, pp. 1297–1310, Aug 2014.

[5] R. Silva Goncalves, V. Delisandra Feltrim, and L. Fondazzi Martimiano, "TCP-UEM: Detecting link failure by keeping end-to-end semantics," *Latin America Transactions, IEEE (Revista IEEE America Latina)*, vol. 11, no. 3, pp. 975–982, May 2013.

[6] X. Wang, Z. Li, and J. Wu, "Joint TCP congestion control and CSMA scheduling without message passing," *Wireless Communications, IEEE Transactions on*, vol. 12, no. 12, pp. 6194–6204, December 2013.

[7] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: Incast congestion control for TCP in data-center networks," *Networking, IEEE/ACM Transactions on*, vol. 21, no. 2, pp. 345–358, April 2013.

[8] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of tcp pacing," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, mar 2000, pp. 1157 –1165 vol.3.

[9] S. ElRakabawy and C. Lindemann, "A practical adaptive pacing scheme for tcp in multihop wireless networks," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 4, pp. 975 –988, aug. 2011.

[10] G. Bhutani, "A near-optimal scheme for tcp ack pacing to maintain throughput in wireless networks," in *Communication Systems and Networks (COMSNETS), 2010 Second International Conference on*, jan. 2010, pp. 1 –7.

[11] C.-Y. Luo, N. Komuro, K. Takahashi, and T. Tsuboi, "Paced tcp: A dynamic bandwidth probe tcp with pacing in ad hoc networks," in *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*, sept. 2007, pp. 1 –5.

[12] K. Shi, Y. Shu, O. Yang, J. Wang, and J. Luo, "Improving TCP performance for EAST experimental data in the wireless LANs," *IEEE Transactions on Nuclear Science*, vol. 58, pp. 1825–1832, Aug. 2011.

[13] X. Zhang, W. Zhu, N. Li, and D. Sung, "TCP congestion window adaptation through contention detection in ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 59, pp. 4578–4588, Nov. 2010.

[14] E. Park, D. Kim, H. Kim, and C. Choi, "A cross-layer approach for per-station fairness in TCP over WLANs," *IEEE Transactions on Mobile Computing*, vol. 59, pp. 898–911, Jul. 2008.

[15] C. Luo, F. Yu, H. Ji, and V. Leung, "Cross-layer design for TCP performance improvement in cognitive radio networks," *IEEE Transactions on Vehicular Technology*, vol. 59, pp. 2485–2495, Jan. 2010.

[16] D. Chen, H. Ji, and V. Leung, "Distributed best-relay selection for improving TCP performance over cognitive radio networks: A cross-layer design approach," *IEEE Journal on Selected Areas in Communications*, vol. 30, pp. 315–322, Feb. 2012.

[17] V. Mai, T. Thang, and A. Pham, "Performance of TCP over free-space optical atmospheric turbulence channels," *Optical Communications and Networking, IEEE/OSA Journal of*, vol. 5, no. 11, pp. 1168–1177, Nov 2013.

[18] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP reno performance: A simple model and its empirical validation," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 133–145, 2000.

[19] C. B. Samios and M. K. Vernon, "Modeling the throughput of tcp vegas," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 71–81, Jun. 2003. [Online]. Available: http://doi.acm.org/10.1145/885651.781037

[20] H. Xie, W. Pazzi, and A. Boukerche, "A novel cross layer tcp optimization protocol over wireless network by markov decision process," in *GLOBECOM12. Global Communication Conference*, 2012, pp. 5945–5950.

[21] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 3, pp. 535–547, March 2000.