REboost: Improving Throughput in Wireless Networks Using Redundancy Elimination

Kilho Lee, Daehyeok Kim, and Insik Shin, *Member, IEEE*

***Abstract*— Traffic redundancy elimination (RE) is an attractive approach to improve the throughput in bandwidth-limited networks. While previous studies show that the RE is useful for improving the throughput in such networks, we observed that the RE would not be an effective solution in wireless networks. We found the TCP congestion control cannot take advantage of the RE, without knowing how the underlying RE system manipulates each TCP packet. In this letter, we present a novel technique called REboost to enable the TCP layer to be aware of the underlying RE system and improve the throughput. Our evaluation with a prototype shows that REboost significantly improves the throughput compared with the previous RE systems.**

***Index Terms*— Traffic redundancy elimination (RE), deduplication, congestion control, packet loss, link delay.**

I. Introduction

With the growth and proliferation of mobile devices (e.g., smartphones and tables) and mobile networks, users can access various types of mobile applications almost anywhere. Some of these applications, such as video streaming and video calling, require high network throughput[1] to provide a high quality of service (QoS). Unfortunately, as mobile networks typically have limited bandwidth on their wireless links, meeting the QoS requirements of such applications is not a trivial task.

The traffic redundancy elimination techniques (RE) [1]–[5] have emerged as an effective way to improve network throughput by reducing repeatedly transferred bytes. It situates a pair composed of a RE sender and a RE receiver in the middle of the network, and suppresses redundant byte transmission between the RE pair. Both the RE sender and receiver maintain identical cache entries by caching all traffic flowing between them. Using the cache entries, the RE sender can replace repeated bytes with their cache references, and the RE receiver can construct the original traffic with the cache references.

A number of wireless RE systems [2]–[4] have been proposed to alleviate traffic congestion in bandwidth-limited wireless networks. They employ two common design principles: 1) placing the RE receiver inside mobile devices and the RE sender on intermediate nodes in the network (e.g., SGSN in cellular networks [3] and an intermediate RE-helper node [4]) to suppress redundant transmissions on the wireless links and 2) implementing the RE functionality on top of the IP layer to support various transport layer protocols including UDP and TCP.

Previous studies claim that wireless characteristics such as packet losses and node mobility hinder the identification and reduction of redundant traffic. To solve this problem, they propose advanced RE techniques such as a loss resilient cache sync protocol [2] and a feedback based cache update algorithm for mobility support [3]. A recent work [5] reports that up to 42% of repeated mobile traffic is reducible with the RE.

Although previous RE systems effectively eliminate redundant bytes for improving the network throughput, from our preliminary experiment, we found that the IP layer RE would not be effective in wireless networks. We implemented a prototype of IP layer RE system, and measured how the system affects the network throughput by using a smartphone (i.e., Nexus 5) connected to the commercial LTE network. We observed that when the network has a long delay or a high packet loss rate, the throughput of TCP traffic does not increase even if the RE reduces a significant amount of traffic.

The reason for this anomaly is that the TCP layer cannot take advantage of the benefits from the IP layer RE in wireless networks. In theory, since the RE reduces the size of each packet, the TCP layer can send a greater number of packets by increasing the sending rate until the rate reaches the link capacity (i.e., bandwidth). However, if the network experiences long delays or packet losses, the sending rate will not be able to increase. This is because when the network conditions are unstable, the sending rate is decided by network events such as duplicated ACKs regardless of the link capacity. There consequently exists a discrepancy between the data rate at the link layer and the TCP sending rate when the IP layer RE is applied to the unstable wireless network.

It is challenging to solve this inconsistency problem. Conceivably an explicit increase in the TCP sending rate could solve this problem. However, this is not straightforward because arbitrary changes in the TCP sending rate could break the TCP congestion control behaviors. Thus, we should design a mechanism that increases the TCP sending rate while preserving the original TCP congestion control behaviors.

We propose a new RE technique called REboost that effectively addresses the challenges to improve the TCP throughput with the IP layer RE in wireless networks. In REboost, we design the TCP layer to control its sending rate depending on the size of compressed packets, rather than the size of original packets. While REboost can explicitly increase the sending rate at the TCP layer, it maintains the data rate at the link layer to follow the original TCP behaviors. Therefore, it can preserve the original TCP congestion control principles, and can improve the TCP throughput at the same time. Our evaluation results show that REboost can successfully improve the TCP throughput even when the TCP sending rate is restricted by a long delay and a high packet

Manuscript received September 10, 2016; accepted October 8, 2016. Date of publication October 19, 2016; date of current version January 6, 2017. This work was supported in part by BSRP (NRF-2015R1D1A1A01058713), IITP (B0101-15-0557), NCRC (2010-0028680), and NRF(2015M3A9A7067220) funded by the Korea Government (MEST/MSIP/MOTIE). The associate editor coordinating the review of this letter and approving it for publication was B. Bellalta.

K. Lee and I. Shin are with the School of Computing, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea (e-mail: khlee.cs@kaist.ac.kr; insik.shin@cs.kaist.ac.kr).

D. Kim is with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: daehyeok@cs.cmu.edu).

Digital Object Identifier 10.1109/LCOMM.2016.2618798

[1]We define the *throughput* as the number of bytes that have arrived at the application layer per unit time.

TABLE I
TRANSMISSION TIME OVER THE LTE NETWORK

	mean(s)	stdev(s)
Without RE	3.78	0.51
IP layer RE	3.61	0.53

loss rate. We also prove the compatibility of REboost by combining it with various TCP versions.

The rest of this letter is organized as follows. Section II explains the TCP throughput problem of the RE over wireless networks, Section III describes our approach, Section IV presents our evaluation results, and Section V concludes.

II. RE ON WIRELESS NETWORKS

A. Assumptions on RE System

We assume that the RE system has the following characteristics. The RE system operates on the IP layer to support various transport protocols and to take advantage of ease of deployment. In addition, the RE receivers are deployed inside mobile devices to bring benefits to the last-mile of wireless networks which typically suffer from bandwidth limitations. The RE senders are deployed at content servers or network middleboxes (e.g., Web Proxy or PEP [6]), depending on the policy of the service providers. The RE system adopts basic techniques to support RE functionalities including MODP [1] for chunk boundary detecting, chunk-match [7] for cache indexing, and SHA-1 for chunk hashing.

B. TCP Throughput With RE System in Wireless Networks

1) Preliminary Experiment: We built a RE prototype running on the IP layer in the Linux kernel to conduct our motivating experiment. We set a desktop PC as a RE sender and an Android mobile phone (i.e., Nexus 5) as a RE receiver. The RE sender and the RE receiver are connected to the Internet via a campus network and a LTE network serviced by Korea Telecom, respectively. We generated network traffic by sending a 10 Mbyte TCP stream (i.e., A MPEG-4 v2 video file) from the RE sender to the RE receiver.

2) Observation: Table I represents the averages of 50 trials of file transmission with and without the RE system and their standard deviations. The RE can reduce the traffic size to 4.5 Mbyte. However, we observed that the RE is ineffective in the wireless network environment despite the significant reduction in the amount of traffic.

3) Root Cause of the Anomaly: We found that the TCP congestion control restricts an increase in the TCP sending rate, and hinders the RE system from improving the throughput. In the wireless network environment, the TCP sending rate with the RE system is comparable to that without RE. This is because when the network experiences long delays or packet losses, network events (e.g., duplicated ACKs) decide the sending rate regardless of the current link capacity. Therefore, the packet compression with the RE system does not affect the sending rate even though there is enough remaining link capacity.

C. Challenges of Improving TCP Throughput

In order to improve the network throughput with a RE system in wireless networks, the RE system should explicitly increase the TCP sending rate by adjusting the size of the TCP congestion window. However, arbitrary changes in the window can damage the TCP congestion control since the window size should be determined by the current network state. Therefore, we should design a scheme that can extend the window size while maintaining the original congestion control principle.

III. SOLUTION: REboost

A. Design

We propose REboost, a novel technique to address the challenge. REboost runs together with an IP layer RE system on a single node. It operates on the TCP layer and accesses to cache information of the underlying IP layer RE system. REboost recognizes redundancy in each packet from the cache information and adjusts the TCP sending rate of the node based on the redundancy in each packet.

When the RE system removes duplicated bytes from each packet, the link layer data rate at time t (denoted as dr_t), is less than the corresponding TCP sending rate (denoted as tr_t). We enable REboost to access cache entries of the RE system running on the IP layer. Due to this, REboost knows how much the size of each packet can be reduced by the RE and it can estimate dr_t when it determines tr_t. At time t, REboost refers to the sending rate determined by the original TCP congestion control (denoted as ref_t), and increases tr_t until the estimated dr_t reaches ref_t. If the underlying IP layer RE successfully reduces the size of each packet, the following conditions hold:

$$tr_t \geq ref_t \tag{1}$$

$$dr_t = ref_t \tag{2}$$

Condition 1 implies that REboost can make the TCP sending rate greater than the original TCP, and Condition 2 implies that REboost preserves congestion control behaviors of the original TCP, since the data rate at the link layer follows the intended sending rate of the original TCP.

B. Implementation

We built a prototype implementation on top of Linux 3.4 to validate the effectiveness of REboost. REboost was implemented as a part of Linux TCP. It runs together with the IP layer RE system on a single node and adjusts the TCP sending rate depending on the awareness of the RE system.

1) IP Layer RE Awareness: In order to estimate dr_t corresponding to tr_t at REboost on the TCP layer, the underlying RE sets a kernel flag to inform the TCP of its presence, and passes a pointer for the RE cache to REboost. With the RE cache pointer, REboost can calculate the amount of bytes that will be reduced by the IP layer RE and thereby estimates dr_t.

2) Linux Congestion Control: The Linux TCP controls its sending rate based on $cwnd_t$, which represents the size of the congestion window in the number of packets at time t. Since Linux TCP typically determines the size of each packet based on the Maximum Segment Size (MSS), the TCP sending rate at time t is determined as $\frac{cwnd_t * MSS}{RTT}$. The REboost implementation uses this rate as ref_t.

3) Increase in TCP Sending Rate: The REboost implementation increases the TCP sending rate by adjusting the size of each packet. We modified the TCP segmentation routine to enlarge the compressed size of each packet to MSS. Although each packet could be larger than MSS at the TCP layer, this does not violate any packet size constraints because the packet will be compressed to MSS at the IP layer. REboost keeps

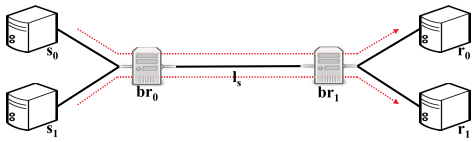

Fig. 1. Testbed overview.

$cwnd_t$ unchanged, to preserve the original TCP congestion control behaviors.

With the modified TCP segmentation, the following conditions hold:

$$\frac{\sum_i^{cwnd_t} len(p_i)}{RTT} \geq \frac{cwnd_t * MSS}{RTT} \quad (3)$$

$$\frac{\sum_i^{cwnd_t} len(p_i')}{RTT} = \frac{cwnd_t * MSS}{RTT} \quad (4)$$

where we introduce the notations as follows.

p_i	The i-th packet in the congestion window.
p_i'	The compressed i-th packet after applying RE.
$len(p_i)$	The byte length of packet p_i.

The left-hand sides of Conditions 3 and 4 represent tr_t and dr_t of REboost, respectively, and the right-hand sides represent ref_t. Condition 3 implies that REboost implementation increases tr_t until it becomes greater than ref_t which is determined by the original TCP, since the size of each packet is greater than or equal to MSS while $cwnd_t$ is identical to the original TCP. Condition 4 implies that the implementation preserves the original congestion control behaviors, since its data rate at the link layer is equal to ref_t.

IV. Evaluation

A. Experiment Setup

We built a testbed as shown in Figure 1. It consists of six desktop machines connected via 100 Mbit/s Ethernet in a *dumbbell* topology. Since it is difficult to conduct controllable and reproducible experiments over wireless links, the testbed emulates wireless link behaviors over the Ethernet links. In addition, the dumbbell topology enables network flows to compete with each other over a single bottleneck link (i.e., Link l_s). Thereby it enables flow fairness experiments on the testbed. In the testbed, a sender node s_i and a receiver node r_i are connected through software bridges [8] br_0 and br_1. The software bridges emulate link delays and packet losses of wireless links over the Ethernet link l_s by using Netem [9] and Netfilter [10]. The testbed utilizes widely used packet loss models; the Bernoulli model for independent packet loss patterns and the Simple-Gilbert model [11] for burst packet loss patterns. In the Bernoulli model, each packet is dropped according to Bernoulli probability p. On the other hand, the Simple-Gilbert model is a Markov model with two states, G (0% loss) and B (100% loss). The state is updated prior to each packet transmission with given state transition probabilities. Throughout all the experiments, a 10 Mbyte video file was used as source traffic to ensure a sufficient number of round-trips. The file size was reduced from 10 Mbyte to 4.5 Mbyte after applying the RE. In addition, the default bandwidth of each link was 100 Mbit/s (i.e., Ethernet bandwidth), and Cubic [12] was applied as a default TCP variant. Furthermore, since Netem incurs randomness of network events (e.g., packet loss timings), we conducted 30 trials for each experiment and presented their averages as final results. We compared the performance of REboost with two baselines, a system without the RE denoted as *No-RE* and a system with the IP layer RE denoted as *IP-RE*. All baselines use the unmodified TCP.

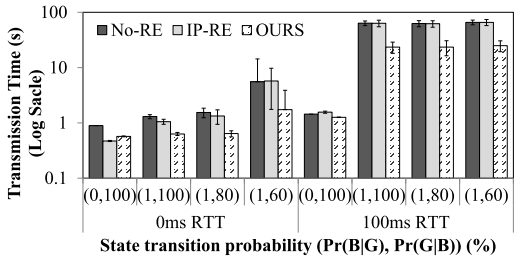

Fig. 2. Transmission times under the unstable networks.

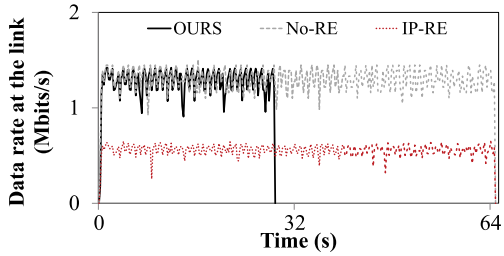

Fig. 3. Link layer data rate under the unstable network.

B. TCP Throughput Improvement in Unstable Networks

The main purpose of REboost is to improve network throughput with a RE system under unstable wireless networks. In order to evaluate how well REboost achieves this objective, we measured the file transmission time while changing link delays and packet loss rates. The node s_0 transmitted the 10 Mbyte video file to the node r_0 with or without RE. To emulate wireless link behaviors, the link l_s was set to have 0 ms or 100 ms of RTT and set to use the Simple-Gilbert loss model. In order to control the degree of packet loss and burstiness, the state transition probability from G to B (i.e., Pr $(B|G)$) was set to 0% or 1% and the probability from B to G (i.e., Pr $(G|B)$) varied from 100% to 60%. Figure 2 shows that REboost outperforms other baselines, even though the network suffers from long delays and packet losses. This improvement comes from the enlarged TCP sending rate based on the packet redundancy information from the underlying RE system. The original TCP determines its sending rate based on network events regardless of link capacity under unstable networks. Therefore, despite of a significant amount of reduction in the traffic size, IP-RE cannot increase the TCP sending rate and shows similar performance to that of No-RE. In contrast, REboost explicitly adjusts the TCP sending rate based on information about the amount of traffic reduction from the IP layer RE, and outperforms other baselines.

C. Preserving TCP Principles

Another important challenge of REboost is to preserve original TCP behaviors when it explicitly increases the TCP sending rate. Since arbitrary changes in the TCP sending rate can hurt the performance of each flow and the fairness of the entire network, we evaluated how REboost affects the data rate of each flow and the fairness of the network.

1) Link Layer Data Rate: We measured the link layer data rate of a single flow while applying REboost under the unstable network, which has 100 ms of RTT and 1%

TABLE II
JAIN'S TCP FAIRNESS INDEX OF COMPETING FLOWS

Flow 1	Flow 2	Mean	Stdev
No-RE	No-RE	0.9956	0.0167
OURS	No-RE	0.9919	0.0104
IP-RE	No-RE	0.9493	0.0115

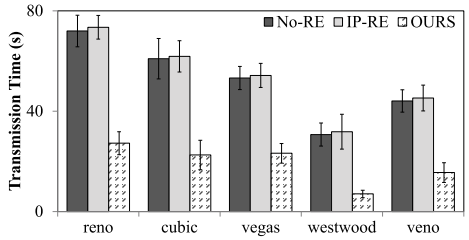

Fig. 4. Transmission times with various TCP versions.

of packet loss rate. To enforce identical packet drop patterns between each experiment, the network was set to drop every 100-th packet. Figure 3 shows that the data rate of REboost performs similar to the original TCP (i.e., No-RE).

2) TCP Fairness: In addition, we measured the data rate of two flows when they compete with each other, in order to evaluate how REboost affects the flow fairness of the entire network. We generated flow 1 (i.e., from s_0 to r_0) and flow 2 (i.e., from s_1 to r_1), at the same time. Flow 1 utilizes REboost or baselines, while flow 2 does not use the RE system. For the fast convergence to a fair share of bandwidth and elimination of randomness in both flows, the network parameters were set to have 20 ms of RTT and no packet loss. Reno TCP was used since it can provide a fair share of link bandwidth. We calculate the Jain's fairness index [13] by using the average throughput of the competing flows. The closer to 1.0 the index is, the better fairness. Table II shows that REboost archives fairness comparable to the original TCP (i.e., No-RE).

3) Implications: Figure 3 and Table II show that REboost performs similar to the original TCP in terms of the link layer data rate and the fairness. This is because that REboost preserves the congestion control algorithm (i.e., changes in $cwnd$) of the original TCP and only enlarges each packet size to MSS after applying the RE. IP-RE shows a lower data rate and a worse fairness than No-RE due to the inconsistency between the link layer data rate and the TCP sending rate.

D. Interoperability With TCP Variants

Since many TCP variants have been proposed so far, interoperability with various TCP variants is one of the important factors of REboost. We evaluated the performance of REboost when it runs with five different TCP variants of their own congestion control behaviors. We used Reno and Cubic [12] which operate reactively to packet losses, Westwood [14] and Vegas [15] which operate proactively based on the network bandwidth estimation, and Veno [16] which differentiates the causes of packet loss and reacts according to the cause. To emulate unstable wireless links, the network conditions were set to have 100 ms of RTT and Bernoulli random loss with 1% of probability.

Figure 4 shows that REboost results in consistent performance improvements even though it runs with different TCP variants. The performance improvements of REboost come from the modified segmentation algorithm, not the congestion control algorithm. All TCP variants share the identical segmentation algorithm and have their own congestion control algorithm. And the segmentation algorithm is independent of the congestion control algorithm. Therefore, REboost shows consistent performance improvements whatever TCP variants run with REboost.

E. TCP Throughput Improvement in Commercial LTE Network

We evaluate the effectiveness of REboost in a commercial LTE network by using an identical setup presented in Section II-B.1. REboost took 2.68 seconds in average of 50 trials, with a standard deviation of 0.44 seconds. REboost shows 26% faster transmission time than IP-RE (i.e., 3.61 seconds in average) even in the commercial LTE network.

V. CONCLUSION

In this letter, we propose REboost to resolve the TCP throughput problem that IP layer RE systems typically experience in wireless networks. REboost enables the TCP to explicitly increase its sending rate with awareness of the underlying RE systems while preserving its original behaviors. Our implementations and evaluations show the effectiveness of REboost for improving the TCP throughput with RE systems in wireless networks. Since REboost is transparent to network functionalities, we expect that it can achieve further throughput improvements by combining with advanced caching techniques [17].

REFERENCES

[1] N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," in *Proc. ACM SIGCOMM*, 2000, pp. 87–95.
[2] C. Lumezanu, K. Guo, N. Spring, and B. Bhattacharjee, "The effect of packet loss on redundancy elimination in cellular wireless networks," in *Proc. ACM IMC*, 2010, pp. 294–300.
[3] S. Sanadhya, R. Sivakumar, K.-H. Kim, P. Congdon, S. Lakshmanan, and J. P. Singh, "Asymmetric caching: Improved network deduplication for mobile devices," in *Proc. ACM MobiCom*, 2012, pp. 161–172.
[4] A. Beirami, M. Sardari, and F. Fekri, "Wireless network compression via memory-enabled overhearing helpers," *IEEE Trans. Wireless Commun.*, vol. 15, no. 1, pp. 176–190, Jan. 2016.
[5] S. Woo, E. Jeong, S. Park, J. Lee, S. Ihm, and K. Park, "Comparison of caching strategies in modern cellular backhaul networks," in *Proc. ACM MobiSyS*, 2013, pp. 319–332.
[6] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance enhancing proxies intended to mitigate link-related degradations," RFC. 3135, Jun. 2001.
[7] B. Aggarwal *et al.*, "EndRE: An end-system redundancy elimination service for enterprises," in *Proc. USENIX NSDI*, 2010, pp. 1–14.
[8] *Linux Bridge*. (Oct. 25, 2016). [Online]. Available: https://wiki.linuxfoundation.org/networking/bridge
[9] *Linux Netem*. (Oct. 25, 2016). [Online]. Available: https://wiki.linuxfoundation.org/networking/netem
[10] *The Netfilter.org Project*. (Oct. 25, 2016). [Online]. Available: http://www.netfilter.org/
[11] G. Hasslinger and O. Hohlfeld, "The Gilbert–Elliott model for packet loss in real time services on the Internet," in *Proc. 14th GI/ITG Meas., Modelling Eval. Comput. Commun. Syst. (MMB)*, Mar. 2008, pp. 1–15.
[12] S. Ha, I. Rhee, and L. Xu, "Cubic: A new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.
[13] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," Digital Equipment Corp. (DEC), Tech. Rep. TR-301, 1984.
[14] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proc. ACM MobiCom*, 2001, pp. 287–297.
[15] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 24–35, Oct. 1994.
[16] C. P. Fu and S. C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 2, pp. 216–228, Feb. 2003.
[17] D. Bulti and K. Raimond, "Optimizing caching in a patch streaming multimedia-on-demand system," *J. Comput. Sci. Eng.*, vol. 9, no. 3, pp. 134–141, Sep. 2015.