2009 IEEE 34th Conference on Local Computer Networks (LCN 2009)
Zürich, Switzerland; 20-23 October 2009

# TCP Feno: Enhancement for Higher Accuracy of Loss Differentiation Over Small Buffer Heterogeneous Networks

Jae-Hyun Hwang, See-Hwan Yoo, and Chuck Yoo
Department of Computer Science and Engineering
Korea University, Seoul, Korea
Email: {jhhwang,shyoo,hxy}@os.korea.ac.kr

***Abstract*—It is well known that TCP shows performance degradation over wired/wireless networks since TCP regards packet loss as network congestion. TCP Veno has successfully addressed this fundamental problem by proposing an end-to-end loss differentiation algorithm, which distinguishes the cause of packet loss by the number of packets in the router buffer. Unfortunately, Veno's algorithm shows very low accuracy on small buffer routers that are emerging recently as a new challenge for Internet routers. In the small buffer networks, routers can overflow easily in a short time although Veno diagnoses wireless loss, and this leads to the failure in loss differentiation. Furthermore, when congestion loss is misdiagnosed as wireless loss, Veno shows poor Reno-friendliness since it can increase sending rate by setting *ssthresh* to a larger value than Reno. In this paper, we propose a more accurate loss differentiation algorithm for small buffer heterogeneous networks. Our algorithm accurately distinguishes wireless and wired packet loss by newly defining *Congestive rate*. Through extensive network simulations, we confirm that our new TCP *Feno* achieves not only higher accuracy, but also better Reno-friendliness while not losing performance efficiency.**

## I. Introduction

Traditionally, it has been an accepted opinion that a router should have a bandwidth delay product of buffer size to fully utilize bottleneck links [1]. However, as network bandwidth increases, the buffer size also increases so that it can hold several gigabits for the Internet traffic, and it tends to grow linearly with the link rate. Such a large buffer needs many external DRAMs so that it can be a handicap for router manufacturers in terms of costs and performance. Consequently, many researchers have proposed newly a smaller buffer size to solve the problem [2], [3], [4]. The small buffer router has a definite advantage that it can replace many off-chip DRAMs with a small on-chip SRAM, which scales in speed and much faster than DRAM. It can be carefully expected that the router buffer size will be gradually decreased to tens of packets in the future. In this paper, we focus on TCP adaptation for small buffer wired/wireless heterogeneous networks. Specifically, we found that TCP Veno's loss differentiation algorithm presents very low accuracy in over small buffer networks, and this paper proposes a more accurate algorithm.

It has been reported that TCP Veno shows high performance on wireless environments using its end-to-end loss differentiation algorithm [5]. Veno can distinguish the cause of packet loss by estimating the number of packets in the router buffer. The problem is that a small buffer is relatively easy to overflow in a short time. That is, even if Veno detects the buffer underflow that is misdiagnosed as wireless loss, network congestion can occur at any time. This causes a critical side-effect in terms of network safety since Veno increases the sending rate when it diagnoses packet loss as wireless loss although there is no wireless link. As a result, such misdiagnosis makes Veno more aggressive, and thus results in poor Reno-friendliness. To address this problem, we propose a more accurate loss differentiation algorithm by newly defining *Congestive rate*. We embed our algorithm into Veno implementation and name it Feno in order to distinguish from Veno. Extensive network simulations show that Feno achieves not only higher accuracy in loss differentiation, but also better Reno-friendliness over small buffer environments while still keeping performance efficiency over wireless networks like Veno.

## II. Revisiting loss differentiation in TCP Veno

In this section, we describe the loss differentiation algorithm of Veno that motivates our algorithm, and also explain the weakness over a small buffer router.

### A. Loss differentiation algorithm of TCP Veno

As explained in Section I, TCP Veno has a loss differentiation algorithm, which originally comes from the congestion avoidance algorithm of TCP Vegas [6]. In the concrete, Veno uses *Diff* value of Vegas to check whether or not the bottleneck link is currently congested. The *Diff* value can be obtained by subtracting *Actual* from *Expected*, which are defined as follows.

$$Expected = cwnd/BaseRTT \quad (1)$$

$$Actual = cwnd/RTT \quad (2)$$

$$Diff = Expected - Actual \quad (3)$$

where *cwnd* is the current congestion window size, and *BaseRTT* is the minimum of measured round-trip times (RTT). Since the *Diff* value ultimately means the over-shooting rate against the bottleneck link-rate, the number of packets queued in the router buffer can be expressed by

$$N = Actual \times (RTT - BaseRTT) = Diff \times BaseRTT \quad (4)$$

In TCP Vegas, $N$ is used to avoid network congestion by keeping it low as possible, but Veno uses it as a criterion of congestion. When $N$ is smaller than a threshold $\beta$ (generally 3), Veno diagnoses that the current network bandwidth is not fully utilized, so it regards packet loss as wireless loss. Otherwise, Veno confirms that the packet is lost by network congestion. In this manner, Veno performs the loss differentiation, and reduces *ssthresh* by a smaller amount $(1/5)$ rather than the half of *cwnd* for wireless loss in order to avoid unnecessary rate reduction.

### B. Accuracy and Reno-friendliness over a small buffer router

We observe that Veno's loss differentiation would show high misdiagnosis rate over small buffer networks. This is possible since the router buffer can be full in a relatively short time resulting in congestion loss even though Veno estimates that $N$ is sufficiently small to cause no congestion. We represent this situation to sum of $N$ values as follows when defining $B$ to be the router buffer size and $n$ to be the total number of flows. We also assume that the *BaseRTT* of all flows is the same.

$$B = \sum_{i=1}^{n} N_i = (diff_1 + diff_2 + ... + diff_n) \times BaseRTT \tag{5}$$

This expression shows that the buffer overflow can occur in accordance with the number of competing flows even if each $i$th *Diff* is small. For instance, if the average $N$ value is 2, which is less than the threshold $\beta$, the number of flows over $B/2$ can lead Veno to misdiagnosis. Accordingly, Veno needs a more accurate loss differentiation algorithm that can be applied over small buffer heterogeneous networks.

## III. New loss differentiation algorithm

This section explains our new loss differentiation algorithm in detail. We note that our algorithm is basically based on Veno's one, but by newly defining *Congestive rate*, we improve the accuracy in loss differentiation regardless of the router's buffer size.

### A. Introducing Congestive rate

Veno uses *Expected* and *Actual* of Vegas using *BaseRTT* and current RTT. In other words, it focuses on the start point of buffering to estimate the buffer underflow. Because underflow/overflow boundaries are so close when the router buffer size is small, we argue that such an underflow-oriented scheme is hard to differentiate congestive/non-congestive loss. To clarify the congestive loss condition, we introduce *CongRTT* which means the overflowable RTT. And then, we define *Congestive rate* by *CongRTT* and *Diff*2 to the difference between *Expected* and *Congestive rate* as follows.

$$Congestive\ rate = cwnd/CongRTT \tag{6}$$

$$Diff2 = Expected - Congestive\ rate \tag{7}$$

In the same way with (4), the number of packets to cause network congestion, denoted by $N'$, can be expressed by

$$\begin{aligned} N' &= Congestive\ rate \times (CongRTT - BaseRTT) \\ &= Diff2 \times BaseRTT \end{aligned} \tag{8}$$

From this, we define $N_{cong}$ to the number of packets to reach the buffer overflow from $N$, and this can be obtained by subtracting (4) from (8) as follows.

$$\begin{aligned} N_{cong} &= N' - N \\ &= (Diff2 \times BaseRTT) - (Diff \times BaseRTT) \\ &= (Actual - Congestive\ rate) \times BaseRTT \end{aligned} \tag{9}$$

This means that the closer to zero $N_{cong}$, the easier to overflow the buffer. We can therefore distinguish the cause of packet loss as network congestion when $N_{cong}$ is less than a congestion threshold, $\gamma$, even though $N < \beta$.

### B. Obtaining CongRTT and $\gamma$ value

While *BaseRTT* can be obtained readily by the minimum of measured RTTs, it is relatively difficult to derive *CongRTT* from the maximum of RTTs due to the delay spike and so forth. To obtain a stable *CongRTT*, we sample the measured SRTT (Smoothed RTT) whenever a congestive loss occurs (i.e. $N \geq \beta$). We found that the maximum SRTT is not suitable as *CongRTT* since the deviation of SRTTs is so high, and this makes *CongRTT*(and thus $\gamma$ value) unstable. Consequently, We use a EWMA(Exponentially Weighted Moving Average) filter to calculate *CongRTT*:

$$CongRTT = \alpha CongRTT + (1 - \alpha)SRTT \tag{10}$$

We set the gain value, $\alpha$, to 0.8 in this paper focusing on stability rather than agility. To cover all congestive RTTs from *CongRTT*, we find a proper $\gamma$ value heuristically throughout extensive experiments and determine to use $\gamma = 1$ in this paper. That is, the cause of packet loss can be regarded as network congestion when $N_{cong} \leq 1$, even if $N < \beta$.

## IV. Evaluation

We modify Veno implementation to use our loss differentiation algorithm, and name it Feno (Reno-Friendly Veno). The simulation experiments are performed by NS-2 and Linux TCP implementation for NS-2 provided by [7].

This section provides two experiment scenarios as follows:

- Scenario 1: Small buffer Wired network (no random loss)
- Scenario 2: Small buffer Wired/Wireless network (congestion + random loss)

We also performed some experiments with no congestion loss, but do not present the results since there is no significant difference in performance between Veno and Feno.

Fig. 1 depicts our simulation topology, which has a wired and a wireless bottleneck links. The congestive loss and non-congestive loss occurs separately at the wired and the wireless area respectively. The topology parameters are presented in Table I for each scenario, and the rest of the links that are connected to senders and receivers have 100Mbps of

TABLE I
SIMULATION PARAMETERS.

| Scenario no. | Scenario 1 | | Scenario 2 | |
|---|---|---|---|---|
| Bottleneck area | Wired | Wireless | Wired | Wireless |
| Bandwidth(Mbps) | $1 \sim 50$ | 100 | 10 | 20 |
| Propagation delay(ms) | 30 | 30 | 30 | 30 |
| Random loss(%) | 0.0 | 0.0 | 0.0 | $0.01 \sim 1.0$ |
| Buffer size(pkt) | 20 | BDP | 20 | BDP |

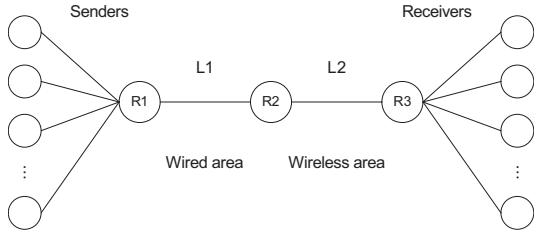



Fig. 1. Simulation topology.

bandwidth, 1ms of one-way propagation delay, and enough buffer size as much as bandwidth-delay product (BDP). Each simulation runs for 300 seconds, and the results are averaged by several times iterations. The packet size is fixed by 1440 bytes, and Selective ACK options are turned on.

### A. Scenario 1 - No random loss

First, we evaluate the accuracy in loss differentiation and Reno-friendliness for a small buffer wired environment. There are ten sending and ten receiving nodes, and five pairs of them use TCP Reno sessions. And then, we compare the case of using five Veno flows with the case of five Feno flows. Since there is no random loss for all links, the accuracy of the loss differentiation algorithm can be calculated by (number of congestion loss diagnoses / number of loss events * 100).

Fig. 2(a)-(b) show the accuracy comparison between Veno and Feno as the buffer size, the number of background flows are varied, respectively. As in the graph, Veno's algorithm shows very low accuracy especially under 20 packets (or high network traffics.) The main reason is that as described in Section II, the buffer size is so small enough to happen the buffer overflow easily although $N$ is less than $\beta$. On the other hand, our algorithm shows very high accuracy, more than 90% even at 10-packet buffer size, and 98% after 25 packets. Fig. 2(c) shows the accuracy on an increase in available bandwidth, and we confirm that our algorithm has better accuracy than Veno regardless of the bottleneck bandwidth.

The low accuracy of Veno's algorithm over a small buffer wired network can affect network safety, specifically Reno-friendliness. Since Veno reduces *ssthresh*, hence *cwnd*, by a factor of $1/5$ for the non-congestive loss, such misclassification can lead to more aggressive sending rate than Reno. Fig. 3(a)-(b) and (c)-(d) present the average throughputs that are normalized by original bandwidth portion of Reno flows from Fig. 2(a) and (c), respectively. That is, we can say that the flows are Reno-friendly when the normalized throughput is close to 1.0. Feno shows low reduction of Reno portion, less than 8% for most cases whereas Veno reduces the Reno portion by 22% to 42%.

### B. Scenario 2 - Congestion/random mixed losses

In scenario 2, we perform experiments over the small buffer heterogeneous environment, in which both congestive and non-congestive loss can occur. We provide three sub-scenarios as follows according to the random loss rate of L2.

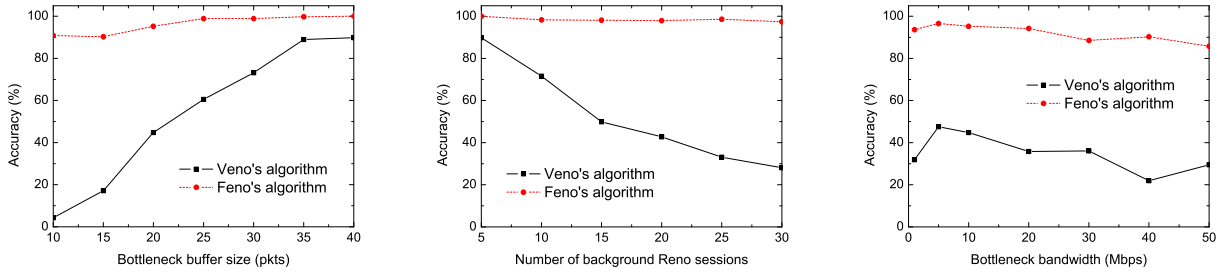
- Scenario 2.1: Wired dominant network (random loss rate – 0.01%)
- Scenario 2.2: Wired/wireless network (random loss rate – 0.1%)
- Scenario 2.3: Wireless dominant network (random loss rate – 1.0%)

Table II lists our experiment results. For each sub-scenario, we count the total number of loss event triggers, and classify congestive and non-congestive(wireless) losses to right and wrong decisions. Thus, the accuracy is calculated by (number of right diagnoses / number of loss events * 100).

In scenario 2.1, where the random loss rate is 0.01%, the accuracy is $(577 + 27)/1235 = 48.9\%$ for Veno and $(1152 + 7)/1226 = 94.5\%$ for Feno. In Veno, the accuracy in congestive loss is very high – $577/(577+15) = 97.5\%$, but the case of non-congestive loss shows very low – $27/(27+616) = 4.2\%$ – with lots of misdiagnoses. This is a similar result with Section IV-A, and thus Veno is poor at Reno-friendliness when the wireless channel is stable. Feno also has low accuracy in non-congestive loss – $7/(7 + 40) = 14.9\%$, but its overall accuracy is still high and the number of *Wrong Wireless* is absolutely low.

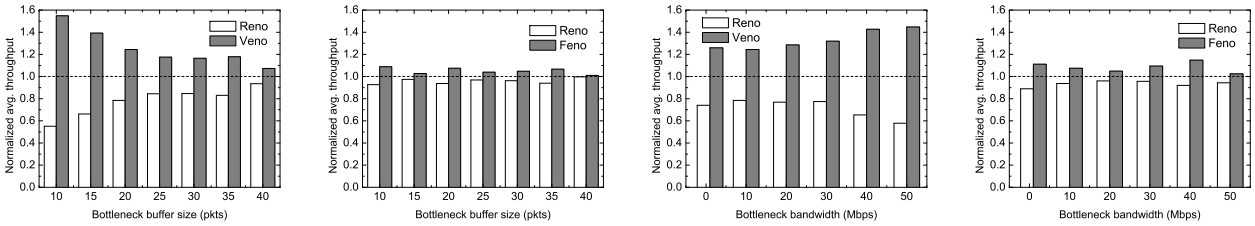
In scenario 2.2, where the random loss rate is 0.1%, the accuracy is $(480 + 237)/1300 = 55.2\%$ for Veno and $(993 + 51)/1280 = 81.6\%$ for Feno. This means that Feno's accuracy in congestive loss is lower than the case of scenario 2.1 – $993/(993 + 191) = 83.9\%$. This is because the loss differentiation algorithm of Veno and Feno assumes that over wireless networks, packet queuing rarely happens due to random loss. However, when congestive loss coexists with wireless loss, the assumption makes loss differentiation difficult since packet loss occurs randomly even if some packets are buffering currently. In this case, Veno regards that misdiagnosed wireless loss to congestion loss is rightful decision as long as $N \geq \beta$ [5]. Feno also follows the belief so that *Wrong Congestion* can be higher in whole wrong decisions. By contrast, Veno shows high *Wrong Wireless* by the effect of the small buffer, which can affect Reno-friendliness.

Lastly, the random loss rate in scenario 2.3 is 1.0%, and the non-congestive loss is more dominant. The accuracy is $(19 + 2142)/2374 = 91\%$ for Veno and $(12 + 2441)/2519 = 97.4\%$ for Feno. In this scenario, both two algorithms show high accuracy in loss differentiation since the assumption described above is satisfied by 1.0% of loss rate. To sum up, through these simulation experiments, we confirm that Feno has more accurate loss differentiation algorithm than Veno for most cases and attempts to reduce *Wrong Wireless* to keep the Reno-friendliness.




(a) Buffer size: 10∼40pkts(10Mbps) (b) Buffer size: 40pkts(10Mbps) (c) Buffer size: 20pkts(1∼50Mbps)

Fig. 2. Loss differentiation accuracy.




(a) Reno vs. Veno (b) Reno vs. Feno (c) Reno vs. Veno (d) Reno vs. Feno

Fig. 3. Reno-friendliness comparison under various buffer sizes and bandwidths.

TABLE II
EXPERIMENT RESULTS FOR SCENARIO 2.

| Scenario no. | TCP name | Total loss event | Right Congestion | Right Wireless | Wrong Congestion | Wrong Wireless |
|---|---|---|---|---|---|---|
| 2.1 | Veno | 1235 | 577 | 27 | 15 | 616 |
| | Feno | 1226 | 1152 | 7 | 27 | 40 |
| 2.2 | Veno | 1300 | 480 | 237 | 93 | 490 |
| | Feno | 1280 | 993 | 51 | 191 | 45 |
| 2.3 | Veno | 2374 | 19 | 2142 | 177 | 36 |
| | Feno | 2519 | 12 | 2441 | 22 | 44 |

## V. CONCLUSION

In this paper, we propose a more accurate loss differentiation algorithm than Veno's algorithm. In the loss differentiation, the possibility of misdiagnosed congestion loss to wireless loss should be reduced to keep the network safety, specifically Reno-friendliness. We have found that the differentiation algorithm of Veno shows poor accuracy over small buffer networks, which can infringe the bandwidth portion of Reno by its aggressive behavior. To address this problem, we newly define *Congestive rate*, which means a sending rate causes the buffer overflow. And then, we apply our algorithm to Veno implementation and name it Feno. Through extensive network simulations, we show that Feno can diagnose the cause of packet loss more exactly over small buffer heterogeneous networks, and also keep the Reno-friendliness, which is one of important features to evaluate a TCP variant.

## ACKNOWLEDGMENT

This work was supported by the Korea Research Foundation grant funded by the Korea government(MEST) (No. 2009-0076092).

## REFERENCES

[1] R. Bush and D. Meyer, "Some Internet Architectural Guidelines and Philosophy," *RFC 3439, IETF*, 2003.

[2] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '04)*, Aug. 2004.

[3] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Routers with Very Small Buffers," in *Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '06)*, Apr. 2006.

[4] Y. Gu, D. Towsley, C. V. Hollot, and H. Zhang, "Congestion Control for Small Buffer High Speed Networks," in *Proceedings of the 26th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '07)*, May 2007.

[5] 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.

[6] L. S. Brakmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE J. Sel. Ar. Comm.*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[7] "A Linux TCP implementation for NS-2," Available: http://netlab.caltech.edu/projects/ns2tcplinux/ns2linux/index.html, 2006.