TCP-Mobile Edge: Accelerating Delivery in Mobile Networks

Tao Han and Nirwan Ansari
New Jersey Institute of Technology
Newark, New Jersey
Email: (th36, nirwan.ansari)@njit.edu

Mingquan Wu and Heather Yu
Huawei Technologies, USA
Bridgewater, New Jersey
Email:(mingquan.wu, heatheryu)@huawei.com

***Abstract*—Owing to the imminent fixed mobile convergence, Internet applications are frequently accessed through mobile nodes. However, service delivery latency is too high to satisfy user expectations. In this paper, we design a new TCP algorithm, TCP-ME (Mobile Edge), to accelerate the service delivery in mobile networks. Considering the QoS (Quality of Service) mechanisms of mobile networks, TCP-ME is designed to differentiate the packet loss caused by wireless errors, traffic conditioning of mobile core networks, and Internet congestion, as well as to react to the packet loss accordingly. To detect wireless errors, we mark the ACK (Acknowledge) packets in the uplink direction at the base station, and the marking threshold is a function of the instantaneous downlink queue length and the number of consecutive HARQ retransmissions. We modify the ECN mechanism with deterministic marking to detect Internet congestion. The packet loss caused by traffic conditioners of mobile networks is detected by whether the incoming DUPACK is marked or not. TCP-ME adapts the inter-packet interval when the packet loss is caused by wireless errors or the admission control mechanism. If the packet loss is due to Internet congestion, TCP-ME applies the TCP-New Reno's congestion window adaptation algorithm. Simulation results show that TCP-ME can speed up web service response time in mobile networks by about 80%.**

I. Introduction

With the rapid development of wireless access techniques and user devices, Internet applications are being migrated to wireless networks. Although advanced physical layer technologies have enhanced the capacity and reliability of wireless networks, the service delivery latency in wireless networks is still too high to satisfy user expectations [1]. Such network latency hinders further deployment of time sensitive applications in wireless networks. To enhance service delivery in wireless networks, extensive research has been done to improve TCP performance. Solutions [2] either try to shield the wireless network from the wired network through a TCP proxy located at the wireless-wire border [3], or attempt to adjust the congestion control algorithms according to the characteristics of wireless networks [4]. However, to our best knowledge, none of these TCP flavors take the QoS mechanism of mobile core networks into account. In mobile networks, the mobile users, before they access Internet, negotiate for the QoS parameters including MBR (Maximum Bit Rate) and the bounded burst size. Based on this information, the gateway of a mobile network constructs a traffic conditioner [5] to limit the maximum rate of traffic admitted to the mobile network. The traffic conditioner either drops non-conforming packets or marks them for preferential dropping if congestion occurs. The token bucket algorithm is recommended as a traffic conditioner at the mobile gateway [5]. Therefore, if the packet arrival rate is larger than the token rate which equals to MBR, the packet will likely be dropped.

The interaction between the TCP congestion control mechanism and the traffic conditioning algorithm has been widely studied. These efforts either adapt the traffic conditioning algorithm to the TCP mechanism [6] or modify the TCP congestion control algorithm according to the characteristics of traffic conditioners [7]. The traffic conditioning algorithm, e.g., the token bucket algorithm, may introduce bursty packet drops when the token bucket has loaned all its tokens [6]. The bursty packet losses beat the TCP congestion window size down to zero. To reduce the bursty packet loss, several solutions have been proposed to adapt the traffic conditioners such as increasing the bucket size [6], adapting the per-flow buffer size according to the traffic burstiness [8], and applying an early dropping policy that drops a packet when the occupancy of the buffer exceeds a predefined threshold but before it is close to overflowing [9]. However, these proposals are not applicable to mobile networks because the parameters of the traffic conditioning algorithm are static and pre-determined according to user service contracts. On the other hand, several TCP congestion control protocols have been made to adjust the packet sending rate [10] or maximum segment size [7] according to the characteristics of the traffic conditioner have been proposed. However, these TCP mechanisms assume the parameters of the traffic conditioner, e.g., the token rates and bucket sizes, are known by the server. In fact, the server located on Internet can hardly know the QoS parameters of mobile networks. In addition, the above TCP algorithms assume the networks can guarantee the data rates according to the parameters of the traffic conditioner. However, the data rate can hardly be guaranteed because the data, before entering mobile networks, have to traverse Internet, which provides only best effort services.

In this paper, we solve the above problem by proposing a new TCP flavor, TCP-ME, which can differentiate the packet loss caused by wireless loss, traffic conditioner in mobile networks, and Internet congestion. Based on the source of the packet loss, TCP-ME effectively estimates the token rate of the traffic conditioner, and adjusts the inter-packet interval accordingly. The rest of the paper is organized as follows. Section II presents in details the TCP-ME mechanism and its operation. Section III presents the simulation results. We conclude the paper in Section IV.

II. TCP-ME

Packets delivery in mobile networks may encounter three different causes of loss: wireless errors, traffic conditioner, and Internet congestion. To differentiate the cause of packet loss and to react accordingly are the leading goals of TCP design. TCP-Jersey [4] modifies the original ECN mechanism to differentiate the packet loss caused by wireless errors from network congestion. TCP-ME applies the similar idea but further detects the packet loss caused by the traffic conditioner at the gateway of mobile networks. Having Identified the cause of packet loss, TCP-ME reacts to the packet loss with different mechanisms, thus improving TCP performance in term of the service response time in mobile networks.

1) Detecting Wireless Errors: To detect the packet loss caused by wireless errors, we implement the ACK marking algorithm at BSs (Base Stations). The proposed marking algorithm utilizes the CE bit in the IP header of the uplink ACKs, which is not used in current TCP/IP protocols [11], to detect the wireless loss. Therefore, in the proposed scheme, we define the CE bit in the IP header of the uplink ACKs as wireless loss alarm (WLA) bit. The WLA bit is set when the equivalent length of the downlink queue of the BS exceeds a predefined threshold. One similar idea reported in [12] proposes to mark the outgoing ACKs when the router the queue length exceeds a threshold to reduce the time required to notify the sender about the congestion. However, our proposed ACKs marking algorithm, instead of detecting wireless congestion, aims to feedback the wireless errors to the server by marking the WLA bit. The equivalent downlink queue length is calculated as

$$L_E = \begin{cases} \dfrac{L_I}{1-(n/\lambda)^3}, & \text{for } n < \beta \ ; \\ L_{th}, & \text{for } n \geq \lambda \ , \end{cases} \tag{1}$$

where L_E is the equivalent queue length, L_I is the instantaneous downlink queue length when the uplink ACK arrives at the BS, L_{th} is the ACK marking threshold, n is the number of consecutive HARQ retransmissions, and λ is the maximum allowed number of HARQ retransmissions. The intuition behind the equivalent queue length calculation is that the more the consecutive HARQ retransmissions, the higher probability of wireless errors, and thus the larger the equivalent queue length. In addition, BS usually keeps individual queues for the attached users, and therefore, the equivalent queue length indicates the wireless condition of individual subscribers. Since the pure ACK packets should not be marked as ECN capable [11], the WLA bit is only marked at BS. Therefore, the marked ACK packets can effectively reflect the wireless errors. When the TCP sender receives ACKs with WLA marked, the TCP sender deduces that the TCP receiver is experiencing bad wireless channels, and the packet loss indicated by the DUPACKs is more likely to be caused by wireless errors.

2) Detecting Internet Congestion: To detect the Internet congestion, we inherit the congestion warning (CW) mechanism of TCP Jersey [13], in which the router marks all the packets when the average queue length exceeds a threshold, and the TCP sender that receives the marked packets will determine whether to adjust its transmission strategy. The non probabilistic packet marking helps the TCP sender detect the packet loss caused by network congestion. When the TCP sender receives DUPACK with ECE bit marked in the TCP header, the TCP sender knows for sure that the network is in the congested state and the packet loss indicated by the DUPACKs is more likely to be caused by congestion.

3) Token Rate Estimation: In mobile networks, the maximum bit rate of traffic admitted into the network is limited by the traffic conditioner at the gateway of the mobile networks. The token bucket algorithm is recommended as the traffic conditioner by 3GPP [5]. Let r be the token rate, and b be the bucket size. Data are conformed if the amount of data transmitted during any time period T does not exceed $b + rT$. The bucket size is recommended to equal the maximum SDU (Service Data Unit) size. Let m be the SDU size; if the packet inter-arrival time is less than $\frac{m}{r}$, corresponding packets will be dropped[1]. To avoid the packet loss caused by the traffic conditioner, the TCP sender shall estimate its packet inter-arrival time at the gateway, and adjust the packet sending interval accordingly. ACKs-triggered TCP algorithms do not consider the effect of traffic conditioners, and thus they do not adapt the inter-packet interval according to the estimation of packet inter-arrival. However, estimating the packet inter-arrival time is not trivial for two reasons. First, the TCP sender does not know the token rate of the traffic conditioner at the gateway. Second, the crossing traffic at the intermediate routers may unpredictably change the packet intervals. As shown in Fig. 1(a), a small inter-packet interval, T_1, at the TCP sender is prolonged by the crossing traffic, thus resulting in a large inter-packet arrival interval, $T_1^{'}$, at the gateway. Fig. 1(b) shows an opposite scenario. Even though the TCP sender keeps a large inter-packet interval, the intermediate routers may shorten the interval,thus resulting in packet loss at the gateway. Therefore,

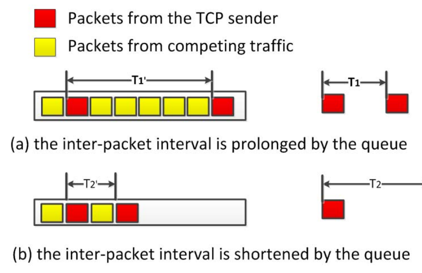
🟥 Packets from the TCP sender
🟨 Packets from competing traffic
T1'
T1
(a) the inter-packet interval is prolonged by the queue
T2'
T2
(b) the inter-packet interval is shortened by the queue

Fig. 1. The dynamics of the inter-packet intervals.

a large inter-packet interval cannot guarantee that packets are

[1]Here, we assume the nonconformant packets will be dropped rather than being marked for preferential dropping.

conformed by the traffic conditioner at the gateway. TCP-ME adjusts its packet sending interval according to the indication of packet loss caused by the traffic conditioner. As discussed in the previous subsection, TCP-ME recognizes the wireless errors by checking whether the WLA bit of the DUPACK is marked, and detects the Internet congestion according to the CW mechanism. If neither WLA bit nor CW bit is marked, TCP-ME renders the cause of the packet loss owing to the inter packet arrival interval being smaller than the tokens' arrival interval. In this case, TCP-ME sets the maximum sending rate, R_{max}, to the current sending rate, R_n, which equals to $1/T_n$, where T_n is the current packet sending interval, and reduces the sending rate by a factor α. To make sure the sending rate is only reduced once in one RTT (Round Trip Time), TCP-ME sets the recovery sequence number, S_{rec}, to the maximum unacked sequence number, S_{unack}^{max}, when the sending rate is reduced. When the TCP sender receives DUPACKs owing to a packet loss caused by the traffic conditioner, the TCP sender checks the sequence number of the lost packet first. Only when the sequence number is larger than S_{rec}, the TCP sender reduces the sending rate. The sending rate adaptation algorithm is shown in Table I. Here, ε is a parameter which controls the

TABLE I
SENDING RATE ADAPTATION ALGORITHM

Sending Rate Increasing

```
SRI()
{
    if ($R_n >= R_{max} - \varepsilon$)
        $R_{n+1} = R_n + C$;
    else
        $R_{n+1} = R_n + \beta \times (R_{max} - R_n)$;
    end if
}
```

Sending Rate Decreasing

```
SRD()
{
    $S_{rec} = S_{unack}^{max}$
    if ($!IP.WLA$)
        $R_{max} = R_n$;
        $R_n = \alpha \times R_n$;
    else
        $R_n = \alpha \times R_n$;
    end if
}
```

transition between the additive rate increasing phase and the proper rate searching phase. C is the additive rate increase step size. β, which is less than 1, is the factor for proper rate searching. A larger β indicates a more aggressive rate increase. Since $SRI()$ reacts to packet loss caused by both wireless errors and traffic conditioner, TCP-ME differentiates the sources of the packet loss. If the packet loss is caused by the traffic conditioner, $SRD()$ sets R_n as R_{max} because R_n is more likely reflecting the upper bound of the allowed sending rate. Otherwise, if the packet loss is caused by wireless errors, the TCP sender does not update R_{max} because the packet loss caused by wireless errors does not reflect the token rate at the traffic conditioner. However, the TCP sender reduces the sending rate by a factor of α for two reasons. First, if WLA is marked, it indicates that the TCP receiver is experiencing bad wireless condition. Therefore, reducing packet sending rate can avoid excessive packet loss due to wireless errors. Second, TCP-ME reduces the sending rate rather than the congestion window since reducing the sending rate can also enhance the probability of packets being admitted by the traffic conditioner at the gateway of mobile networks.

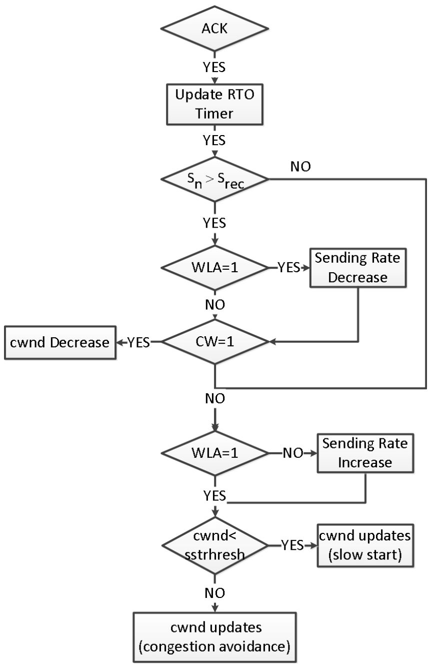

Fig. 2. TCP sender's response to ACK.

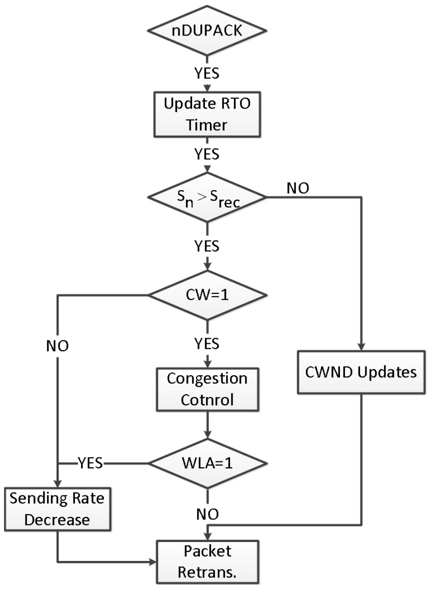

Fig. 3. TCP sender's response to DUPACK.

4) Operation of TCP-ME: The key advantage of TCP-ME is to differentiate the causes of packet loss, and react to them accordingly. If the packet loss is caused by network congestion, the TCP New Reno's congestion control mechanism is applied to adjust the congestion window and to retransmit the lost packet. If the packet loss is caused by wireless errors or traffic conditioner, TCP-ME adjusts its transmitting rate as described in the previous subsection and retransmits the lost packet immediately without modifying the congestion window. The flowcharts of the TCP-ME sender's responses to normal ACK and DUPACK are illustrated in Fig. 2 and Fig. 3, respectively.

III. Simulation Results

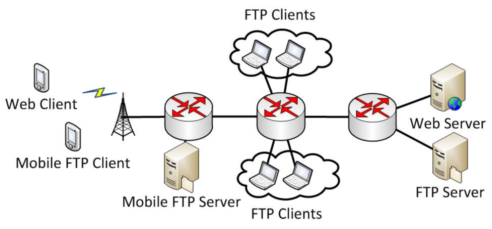

Fig. 4. The network topology for simulations.

Simulations are designed to demonstrate the merits of TCP-ME over the other TCP flavors in terms of service response time in a mobile network. The network topology for the simulation is shown in Fig. 4. We use FTP as the traffic across the wired network and mobile network. The mobile FTP client is served by the mobile FTP server while the other FTP clients are served by the FTP server. In the simulation, the web page contains 1 Mbyte HTML file, 10 media images with their sizes uniformly distributed between 500 and 2000 bytes, and 50 small images with sizes uniformly distributed between 10 and 400 bytes. In the mobile network, we model the wireless channel by using free space as the pathloss model and ITU pedestrian A as the multipath channel model. The parameters of the rate adaptive algorithm are $\varepsilon = 50 bits/sec$, $C = 2000 bits/sec$, $\beta = 0.15$, and $\alpha = 0.9$.

Fig. 5 shows the average web page response time of different TCP flavors. The yellow bars are the average page response times, and the red bars show the standard deviations of the page response times. In the simulation scenario, the average page response time of TCP-ME is about 30 seconds while those of the other simulated TCP flavors including TCP-Westwood, TCP-New Reno, TCP-SACK, and TCP Reno are more than 150 seconds. As compared to the other simulated TCP flavors, TCP-ME reduces the page response time by about 80%. In addition, the standard deviation of the response time of TCP-ME is small–indicative of a relative steady performance in term of downloading the object.

Fig. 6 shows the response time of the inline objects from the HTML file. TCP-ME has the shortest object response time as compared to the other TCP flavors. The average object response time of TCP-New Reno is the largest among the simulated TCP flavors. However, the average page response time of TCP-New Reno is less than that of TCP-Westwood,

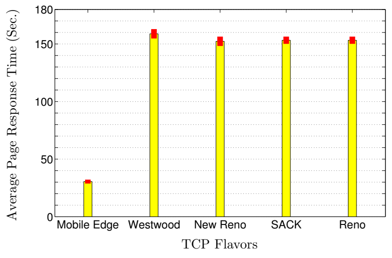

Fig. 5. The web page response time.

and is almost the same as those of TCP-SACK and TCP-Reno. This is because TCP-New Reno is more aggressive during the fast recovery period, and results in more packet loss due to the effect of the traffic conditioner. Therefore, some of the packets of the TCP-New Reno flows surfer long delay and lead to a large average object response time. Since the average object response time is much smaller than the average page response time, several response times of individual large objects do not have a significant impact on the average page response time.

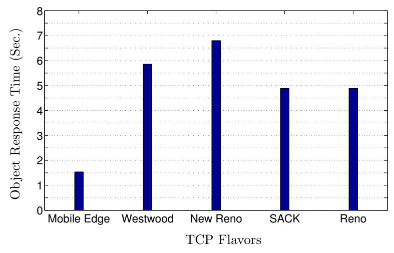

Fig. 6. The response time for the inlined object.

Fig. 7 shows the amount of traffic dropped at the mobile gateway which implements the traffic conditioner. Owing to the rate adaptive algorithm, TCP-ME can effectively estimate the token rate of the traffic conditioner, and thus the average traffic drop of TCP-ME flows is smaller than those of the other TCP flavors. TCP-ME improves the performance of the service response time by avoiding excessive packet loss. This also indicates that TCP-ME does not try to steal the bandwidth from users of other TCP flavors. The average packet drops of TCP-New Reno, TCP-Reno and TCP-SACK are almost the same, and they are overlapped. Fig. 8 shows the average number of TCP retransmissions on the web server. TCP-ME achieves the least retransmissions. That is to say, given the same size of the web page, the sever using TCP-ME sends out the least amount of traffic. Therefore, the improvement gained by using TCP-ME does not sacrifice the performance of other users who utilize different TCP flavors.

Fig. 9 shows the effectiveness of our ACK marking mechanism. We record the time when the packet is lost and the WLA is marked. One green diamond indicates that one ACK is marked while one blue circle means the packet is lost due to wireless errors. The x-axis shows the time when the packet is lost and WLA is marked. The packet loss statistics are collected at the client side while the marked ACKs are

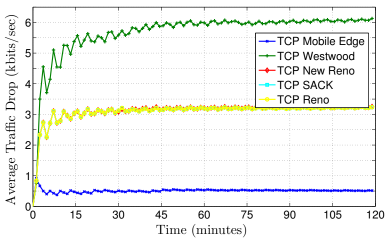

Fig. 7. The average traffic drop at the gateway.

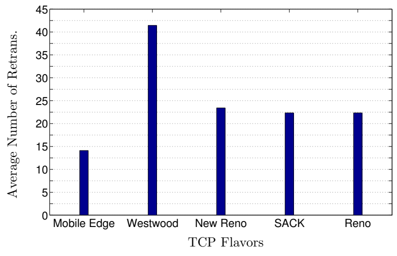

Fig. 8. The number of packet transmissions on web server.

recorded at the server side. The wireless errors are not fed back to the TCP sender if there are no uplink ACKs when the packet loss happens. Therefore, there are less marked ACKs than packets lost due to wireless errors. The ACK marking mechanism cannot perfectly feedback the wireless errors. However, in general, the packet loss caused by wireless errors in mobile networks is not significant since the mobile networks apply HARQ and ARQ retransmission to enhance the reliability of the wireless transmissions. In addition, we keep the marking mechanism aggressive by choosing a small marking threshold. The aggressive marking strategy supports the token rate estimation and adaptation since it enables the TCP sender to detect the packet loss caused by the traffic conditioner.

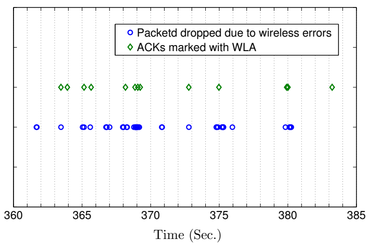

Fig. 9. The test of ACKs marking mechanism at BS.

Fig. 10 shows the performance of the TCP-ME's sending rate adaptive algorithm. The token rate of the traffic conditioner at the gateway is 384 kbps which is the recommended value for best effort services in WiMAX networks. The initial sending rate of TCP-ME is 200 kbps. The median value of the sending rate is about 392 kbps, implying that the sending rate of TCP-ME is larger than the token rate of the traffic conditioner most of the time. This is because the crossing traffic may enlarge the packet intervals as discussed in the

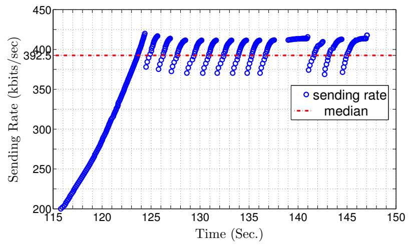

Fig. 10. The sending rate adaptation of TCP-ME.

previous section. Therefore, although the web server has a larger sending rate, the packet arrival rates seen by the gateway are about the token rates of the traffic conditioners.

IV. Conclusion

In this paper, we have proposed TCP-Mobile Edge which can differentiate the packet loss caused by wireless errors, traffic conditioners and the network congestion, and react to the packet loss accordingly. Simulation results show that TCP-ME can reduce the average web page response time by 80% in the simulation scenarios. Other simulated TCP flavors do not perform well in the simulated scenarios because they do not consider the impact of the traffic conditioner at the mobile gateway.

References

[1] M. Claypool and *et. al*, "Characterization by measurement of a CDMA 1x EVDO network," in *WICON '06*, Boston, MA, USA, Aug. 2006.

[2] Y. Tian, K. Xu, and N. Ansari, "TCP in wireless environments: problems and solutions," *Communications Magazine, IEEE*, vol. 43, no. 3, pp. S27–S32, Mar. 2005.

[3] K.-Y. Wang and S. Tripathi, "Mobile-end transport protocol: an alternative to TCP/IP over wireless links," in *IEEE INFOCOM '98*, San Francisco, CA, USA, Mar. 1998.

[4] K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for wireless IP communications," *Selected Areas in Communications, IEEE Journal on*, vol. 22, no. 4, pp. 747–756, May 2004.

[5] "3GPP TS 23.107 (version 10.0.0 release 10): Quality of service (qos) concept and architecture," Mar. 2011, technical Specification.

[6] R. van Haalen and R. Malhotra, "Improving TCP performance with bufferless token bucket policing: A TCP friendly policer," in *IEEE LANMAN '07*, Princeton, NJ, Jun. 2007.

[7] D. Ikegami and T. Tsuchiya, "Evaluating TCP throughput and packet size under policing environment," in *IEEE IC-BNMT '10*, Beijing, China, Oct. 2010.

[8] Y.-C. Chen and X. Xu, "An adaptive buffer allocation mechanism for token bucket flow control," in *IEEE VTC2004-Fall*, Los Angeles, CA, USA, Sep. 2004.

[9] P. Mishra, "Effect of leaky bucket policing on TCP over ATM performance," in *IEEE ICC '96*, Dallas, TX, USA, Jun. 1996.

[10] C.-S. Wu, M.-H. Hsu, and K.-J. Chen, "Traffic shaping for TCP networks: TCP leaky bucket," in *IEEE TENCON '02*, Beijing, China, Oct. 2002.

[11] K. Ramakrishnan, T. Networks, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3168.txt

[12] Z. Li and G. Zhang, "A Novel Differentiated Marking Strategy based on Explicit Congestion Notification for Wireless TCP," in *IEEE TENCON '06*, Hong Kong, China, Nov. 2006.

[13] K. Xu, Y. Tian, and N. Ansari, "Improving TCP performance in integrated wireless communications networks," *Computer Networks*, vol. 47, no. 2, pp. 219 – 237, Feb. 2005.