Wireless-Aware Congestion Control for Transmission over Heterogeneous Networks

Jyun-Siou Fan, Sheng-Shuen Wang, and Hsu-Feng Hsiao
Department of Computer Science, National Chiao Tung University
1001 University Road, Hsinchu, Taiwan
gree1970@hotmail.com, {sswang, hillhsiao}@cs.nctu.edu.tw

Abstract—TCP is the most widely used transport layer protocol on Internet. The congestion control mechanism implemented in a traditional TCP uses events of packet loss as indicators of network congestion. However, this is not suitable for transmission over heterogeneous networks, since there can be wireless packet losses when data passes through wireless networks. If a congestion control algorithm takes the wireless losses caused by common channel errors due to multipath fading, shadowing, and attenuation as an index of network congestion, it will mistakenly lead to degraded performance because of the incorrect reduction of the congestion window. In this paper, we designed a packet loss classification algorithm which uses two-trend detections on relative one-way trip time to differentiate congestion losses from wireless losses. A wireless-aware end-to-end congestion control algorithm is further proposed. The congestion control algorithm discriminates wireless losses from congestion losses, and then it regulates the congestion window and slow start threshold properly. The extensive simulation results show that the proposed algorithms can significantly improve network performance when compared with the congestion control algorithms implemented in the TCP protocols.

Keywords—congestion control algorithm; transmission control protocol; heterogeneous networks.

1. INTRODUCTION

With the advances of wireless technologies and ever increasing demands for mobile communications, Transmission Control Protocol (TCP), which is the dominant transport layer protocol on Internet, provides a reliable, connection-oriented service and is being extended to wireless network. However, wireless communications are carried out in the open air and the signal may disperse and travel on different paths due to reflection, diffraction, and scattering caused by obstacles before it arrives at receivers. The resultant signal may have been significantly distorted and attenuated when compared with the original signal. Therefore, wireless losses [29] can appear randomly due to signal fading by unreliable wireless medium. Traditional TCP is not quite appropriate to be used over heterogeneous (mixed with wired and wireless networks) networks because of the mechanisms of its congestion control algorithm. TCP congestion control algorithm uses the event of packet losses as an indicator of network congestion. If the congestion control algorithm takes wireless losses as an index of network congestion, it will lead to dramatic performance degradation due to the reduction of the congestion window by mistake.

In order to solve this problem, several congestion control approaches have been suggested. There are four different types of approaches [1][2][12][25][26]: split connection, link-layer, end-to-end, and network cooperation. The main idea of split-connection approach [10] is to split connection at the base station between wired and wireless network. The base station serves as a relay node to separate congestion control functionality on wireless links from that on wired networks. When bandwidth asymmetry exists between the wired and wireless path, a huge buffer is required at the relay node to store and forward packets toward the mobile host. “Snoop” is a proxy-based link layer enhancement to cache copies of TCP data packets at the base station, and to monitor the ACKs from a receiver to its sender. Besides, some link layer approaches [26][28] attempt to reduce wireless losses using scheme of automatic repeat request (ARQ) in IEEE 802.11 MAC to recover transmission error locally by retransmitting the lost frame at the link layer. This protocol could reduce end-to-end retransmission and prevent the associated reduction in congestion window size by retransmitting the cached copy for local wireless links. ARQ is also used in 802.16 MAC (WiMAX), and also LTE protocols. For the two approaches mentioned above, the control overhead is considerable as the base station needs to maintain a significant amount of state information for each TCP connection.

Instead of the support from base station, the network cooperation approach [16][17] requires the assistance from intermediate routers to send the information about network condition to end-systems for the improvement of congestion control efficiency in presence of wireless losses. The intermediate routers do not require per-flow states. However, it is expensive to deploy all the intermediate routers to enable this function.

The end-to-end methods are promising since significant gains can be achieved without extensive support at the
network layer at routers and base stations. The main concept of end-to-end approaches is to distinguish wireless losses from congestion losses at end-system and no explicit modifications are required for the intermediate routers, which are out of the control of the end users. This approach has attracted extensive research attention because this approach can be deployed easily and gradually. Besides, the end-to-end approach can be further classified into reactive and proactive congestion control in [20] according to whether feedback information is used to reallocate network resources.

A packet loss classification based on the work in [6] is proposed in this paper to differentiate packet loss classes. According to the classification result, the congestion control algorithm can adjust the sending rate effectively and proactively based on congestion losses instead of wireless losses. Furthermore, an improved congestion control algorithm based on the TCP congestion control mechanism in NewReno was developed.

In section 2, several packet loss classification (PLC) algorithms proposed in the literature are discussed, as well as several congestion control algorithms that are commonly used in TCP protocols. In section 3, a new PLC algorithm based on the trend detection of relative one-way trip time is proposed. The wireless-aware congestion control algorithm for TCP is further proposed. In section 4, we evaluate the performance of the proposed algorithms and the competing algorithms in the literature, followed by the conclusions in section 5.

2. RELATED WORK

In real network environment, there are two classes of packet losses in heterogeneous networks: one is congestion loss, and the other is wireless loss. A packet loss due to network congestion is called a congestion loss. The other kind of packet loss due to shadowing and/or signal attenuation over wireless networks is called a wireless loss. If a wireless loss is used as an indicator of network congestion, the congestion window will be reduced incorrectly.

Biaz scheme [3] uses packet inter-arrival time to differentiate congestion losses from wireless losses at the receiver side using threshold. Biaz assumes that only the last link along the path is wireless and it is also the bottleneck, and the scheme classifies lost packets according to the temporal range. On the other hand, ZigZag [5] increases the classification threshold according to the number of losses encountered because a more severe loss is associated with higher congestion and higher relative one-way trip time (ROTT). In the delay trend scheme [6], the delay trend scheme could explicitly classify this packet loss as congestion loss or wireless loss, respectively, when the ROTT of the packet received after a loss occurs is relatively large or relatively small. If the ROTT of packets falls in an ambiguous region, the delay trend scheme classifies the packet loss according to the variation of ROTT.

In order to solve the performance degradation of TCP when used in heterogeneous network, there are many versions of TCP protocols, including NewReno [9], Vegas [21], Veno [22], Westwood[2][19], New Jersey [16][17], and SACK [11]. There are four phases [7][8] in a typical TCP congestion control algorithm: slow start, congestion avoidance, fast retransmission, and fast recovery. During the slow start phase, a TCP increases congestion window (cwnd) exponentially for each acknowledgment received. When cwnd reaches or exceeds slow start threshold (ssthresh), the control algorithm will enter the congestion avoidance phase. In the congestion avoidance phase, cwnd is at most increased by a segment per round-trip time. When congestion occurs, one-half of the current cwnd is saved in ssthresh. Additionally, if the congestion is indicated by a timeout, cwnd is set to one segment.

If three duplicate ACKs are received by the sender, it is a strong indication that a packet has been lost. So TCP performs a fast retransmission on the lost packet without waiting for the retransmission timer to expire. After the missing packets are transmitted in the fast retransmission phase, the fast recovery phase is performed to control the cwnd until a non-duplicate ACK arrives. The fast recovery phase is an improvement that allows high throughput under moderate congestion, because the received ACKs mean that other packets can still be received. When the third duplicate ACK is received, ssthresh is set to half of the amount of data that has been sent but not yet acknowledged. The lost segment is then transmitted and cwnd is set to ssthresh plus three maximum segment sizes (MSS). The cwnd is increased by one MSS whenever each additional duplicate ACK is received. The congestion window is inflated in order to reflect the additional segment that has left the network.

However, in TCP Reno [30], fast recovery is terminated while the next new ACK arrives, even if the ACK just acknowledges some but not all of the transmitted packets before the fast retransmission. Therefore, NewReno [9] keeps the highest sequence number before retransmission as recover to improve the fast recovery algorithm of Reno that incorporates a response into partial acknowledgments. In NewReno, fast recovery is terminated with either a retransmission timeout or an ACK that acknowledges all of the data that was outstanding at the start of fast recovery procedure. There are two cases for receiving a new ACK. If it acknowledges all of the data up to and including recover, fast recovery procedure is terminated in NewReno and cwnd is set to ssthresh. Then the congestion avoidance phase is performed. Otherwise, if this ACK is a partial acknowledgment, NewReno retransmits the unacknowledged segment. After that, it reduces the cwnd by the amount of new data acknowledged. Due to the improved mechanisms mentioned above, our proposed TCP congestion control algorithm is based on the NewReno to incorporate our new design.

TCP Westwood [2][19][30] (TCPW for short) is a modified version of TCP Reno. TCPW sender makes an end-to-end estimate of the available bandwidth along the connection by measuring the rate of returning ACKs. When a packet loss event is observed, TCPW uses the bandwidth estimate to set the cwnd and the ssthresh. The sender updates the ssthresh by the measured bandwidth and the minimum relative trip time instead of the half of the congestion window.
In [2], it shows that TCPW has better throughput than TCP Reno when there are wireless losses. This is because TCPW uses the current estimated rate as reference to reset the congestion window, but TCP Reno simply halves the congestion window. Besides, there are some derivations based on TCPW, such as TCP WestwoodNR [23] with NewReno feature and TCP Westwood+[24] with low-pass filter for the rate of returning ACKs to increase throughput and fairness over wireless links.

TCP Vegas [21] [30] utilizes the minimum RTT sample value observed during the connection to estimate the number of backlog packets in the buffer of bottleneck link by the difference between the expected throughput and the actual throughput. Two thresholds are used to estimate if the network experiences too many or few backlog packets followed by either decreasing or increasing the congestion window linearly to deal with the problem.

TCP Veno [22][30] improves the performance of TCP Reno by utilizing bandwidth estimation scheme of Vegas. In addition, it differentiates wireless losses from congestion losses by checking if the number of backlog packets is less than a threshold when a packet loss is detected. It implicitly means that the available bandwidth is still not fully utilized due to wireless losses.

TCP Jersey [16] consists of two key components, available bandwidth estimation (ABE) and congestion warning (CW) router configuration. CW is like Explicit Congestion Notification (ECN) configuration for network routers to mark the packets when the average queue length exceeds a threshold so as to help a sender to effectively differentiate packet losses caused by congestion, instead of wireless link errors. The ABE algorithm computes the current available bandwidth based on the time interval of ACK packets, the measured RTT, and the size of data that nth ACK acknowledges.

Instead of inter-ACK gap at the sender, TCP New Jersey [17] enhances TCP Jersey by computing ABE using \( t_n \) as the arrival time of the nth packet at the receiver. Thus the time interval \((t_{n}-t_{n-1})\) of data segments at the receiver can overcome the problem of ACK compression/delay in the reverse path. In addition, ESTCP [27] cooperates between the dynamic AIMD window controller at the source side and traffic controller at the network bottleneck node to achieve better fairness. However, not all routers have the ECN-like support in reality and the performance will degrade dramatically if the router of bottleneck link does not have the ECN-like algorithm implemented. Furthermore, bandwidth estimation algorithms that exploit the gap of inter-ACK or RTT do not work well, especially in asymmetric network where the bottleneck link may be on the reverse path.

3. THE PROPOSED PACKET LOSS CLASSIFICATION AND WIRELESS-AWARE CONGESTION CONTROL

In this section, a new packet loss classification algorithm extended from delay trend scheme is proposed with both increasing and decreasing trends. Besides, an improved congestion control for TCP transmission is proposed. The congestion control algorithm is based on the congestion control in NewReno with the assistance of the proposed PLC algorithm to regulate the transmitted bandwidth fairly and efficiently.

3.1. Packet Loss Classification Algorithm

The ROTT of received packets is exploited to assist packet loss classification, and two-trend detection method is utilized when it is ambiguous to distinguish the class of packet losses. The ROTT is measured as the time difference between the receiving time and the packet sending timestamp recorded in packet header plus a fixed bias. The end-to-end packet delay can be modeled as the summation of propagation delay, queuing delay, transmission delay, and router processing delay. Propagation delay is the time for the electromagnetic waves to traverse along the path. Router processing delay is required for the router to perform multiplex, reassembling, and packet forwarding. Transmission delay is the time required to send a packet into the link. They are usually constant for a given end-to-end path and the same packet length. The remainder, queuing delay, is the time for a packet to stay in a queue. The model for the end-to-end packet delay \( T_d \) is shown in (1):

\[
T_d = \sum_{i} (T_{d,i} + P_{i} + T_{r,i}) + \frac{d}{s}
\]

where \( T_{d,i} \) is the queuing delay of link \( i \), \( T_{r,i} \) is the router processing delay, \( P_{i} \) is the packet size, and \( C_{i} \) is the capacity of link \( i \). The final term is propagation delay where \( d \) is the length of physical link and \( s \) is the propagation speed of EM waves. We use measured ROTT and the trend of the received ROTTs which carries the information about the status of the queue in the bottleneck to classify packet losses in our proposed method.

The flow chart of the proposed method is shown in Fig. 1. Two thresholds \( TG^{up} \) and \( TG^{low} \) as in [6] are adopted to segment three regions to present the upper and lower bound of gray zone. When ROTT is larger than \( TG^{up} \), it means that ROTT is larger than the time that is required when buffer is filled, and we classify the packet loss as a congestion loss. Otherwise, when ROTT is smaller than \( TG^{low} \), the packet loss is classified as a wireless loss. When the measured ROTT falls in the gray zone between \( TG^{up} \) and \( TG^{low} \), “full search” method is used to calculate the two trends, increasing trend index \( incr_{\text{trend}} \) and decreasing trend index \( decr_{\text{trend}} \), as shown in (2) and (3).

\[
\text{Fig. 1 Flow chart of the proposed packet loss classification algorithm.}
\]
addition, if the ROTT of the current received packet falls in the ambiguous area, the increasing trend index and decreasing trend index are compared with the thresholds. We take the increasing trend index and decreasing trend index into consideration to estimate the network condition. If the increasing trend index is larger than an empirical threshold (e.g., 0.5), it means that ROTTs have an increasing trend caused by growing up queuing delay in the bottleneck and the packet loss is caused by network congestion with high probability. In addition, if the decreasing trend index is larger than the threshold (e.g., 0.5) which means the queue in the bottleneck is not fully utilized; the packet loss is classified as a wireless loss. Otherwise, if ROTT is neither increasing nor decreasing, the packet loss is classified as the same as the last PLC result. Although clock skew may make the measurement of ROTT imprecise, the result of increasing/decreasing trend detection is not affected because relative $D_t$ is used during the trend detection.

### 3.2. The Proposed Wireless-Aware Congestion Control Algorithm

With the convenience brought by mobile devices, the heterogeneous networks with wired and wireless channels are more and more common in reality. In heterogeneous networks there are congestion losses caused by network congestion, and wireless losses caused by wireless channel error due to shadowing and attenuation. For the TCP congestion control algorithms which reduce the congestion window regardless of the nature of packet losses, it will mistakenly lead to performance degradation when a wireless loss occurs.

In order to solve this problem, the proposed PLC algorithm is utilized to classify packet losses, and then the TCP congestion control algorithm is modified in response to the packet losses caused by wireless errors to avoid unnecessary performance degradation. The wireless loss flag and packet loss number could be recorded in the reserved field of TCP header. The first bit of the reserved field is used to record the flag. This flag is kept constant for the packets that have the same acknowledgement number until the next packet loss event occurs. The remainder of the reserved field records the difference between the discontinuous sequence numbers. According to the flag and the packet loss number, the sender knows that wireless loss occurs and how many packets are dropped, and then the modified congestion control algorithm can be adopted to adjust the congestion window. In response to wireless losses, the retransmission policy remains the same. That is to say, the sender still retransmits the lost packet when three duplicate ACKs or partial ACKs are received, or when timeout of the retransmission timer occurs. However, the congestion window is increased as if a new ACK was received.

The proposed wireless-aware congestion control algorithm is shown in Fig. 2. The $seqno$ means the sequence number of the next packet requested by the receiver. The $lastack$ is defined as the ACK number of the last received one. After the third duplicate ACK is received, $recover$ is recorded as the highest sequence number (highest_seq) transmitted, and the fast recovery phase starts. The fast recovery phase means that one packet loss has occurred and different policy is used to recover successive packet losses in one window. The main idea of the proposed congestion control is to maximize the $ssthresh$ after timeout occurs and also to maximize the size of $cwnd$ after fast recovery ends if the loss is caused by wireless channel errors.

![Fig. 2 A flow chart of the proposed wireless-aware congestion control for TCP](image)

When the sender receives a packet, it checks whether this packet is a new ACK by comparing $seqno$ with $lastack$. If $seqno$ is equal to $lastack$, this ACK is a duplicate ACK. According to the number of the duplicate ACKs (dupack), different strategies are taken. If dupack is less than three, we check whether the loss is caused by network congestion or wireless error. If the loss is classified as a wireless loss, cwnd is increased by $(loss_num+1)$ where $loss_num$ is defined as the number of lost packets at the lost event and it is recorded in the header of corresponding ACK packet. Otherwise, cwnd is constant and this action is the same as the control in NewReno.

When the sender receives the third duplicated ACK, the control algorithm keeps recover, sets $ssthresh$ to be the 0.8*cwnd if a wireless loss is detected. Otherwise, the value of $ssthresh$ is set as 0.5*cwnd. We set higher $ssthresh$ in case of wireless losses so as to acquire larger congestion window in the end of fast recovery. Then cwnd is set to be one-half of the current congestion window. The lost packet with sequence number $lastack$ is retransmitted, and the fast recovery phase is initiated.

In addition, the variable $dupwnd$ is utilized to keep track of...
the increment of sending window whenever the duplicated ACK indicates network congestion, because the value of cwnd is still used to reset ssthresh when timeout occurs during fast recovery phase. Therefore, the sending window size is determined by the sum of cwnd and dupwnd during fast recovery phase. If the loss is caused by wireless error, we keep the accumulated loss_num and increase cwnd by (loss_num+1). Otherwise, the dupwnd is increased by 1 as in NewReno. The same action is also applied when the number of dupack is greater than three. The cwnd is increased if a wireless loss occurs and dupwnd is increased if congestion loss occurs.

If the seqno of new ACK is greater than last_ack during fast recovery phase, the control algorithm enters partial ACK action or exits fast recovery phase according to the comparison between seqno and recover. If seqno is less than or equal to recover (which means partial ACK), we retransmit the lost packet and reduce dupwnd by the difference between seqno and last ack. Moreover, we set cwnd to the sum of ssthresh and loss_num, and reset loss_num to 0 when seqno is more than recover. Therefore, under the situation of wireless losses, ssthresh can be larger than the original one by loss_num/2 after timeout occurs, while the size of cwnd is increased by loss_num after fast recovery phase ends.

4. Performance Evaluation

The performance of the proposed PLC algorithm in terms of classification accuracy is presented first. The proposed wireless-aware congestion control algorithm for TCP is then compared with several congestion control algorithms used in different TCP protocols. Extensive simulation results are conducted at different wireless error rates/traffic patterns, and different network topologies.

4.1. Performance for the Parking-Lot Topology with Multiple Bottlenecks

We use the parking-lot topology with multiple bottlenecks to evaluate the proposed PLC algorithm and the delay trend scheme in literature. The new topology is shown in Fig. 3 and the link delay and link capacity for each link are also indicated. The wireless link is between node W8 and node M0.

![Fig. 3 Parking-lot topology with multiple bottlenecks](image)

There are three TCP flows and all of them are FTP. The traffic is set as the following. TCP1 is from node W0 to node M0 during 0 to 100 seconds. TCP2 is from node W1 to node W5 during 20 to 60 seconds. TCP3 is from node W4 to node W7 during 40 to 80 seconds. The total simulation time is 100 seconds. The error model of the wireless links described by a two-state Markov model of Gilbert-Elliott channel [14] is turned on at the second 60 and its average error rate is equal to 0.034.

Ac [3], which is defined as the ratio of the number of congestion losses correctly classified to the total number of congestion losses, is used to evaluate the accuracy of congestion loss discrimination. Aw is defined as the ratio of the number of wireless losses correctly classified to the total number of wireless losses and it is used to evaluate the accuracy of wireless loss discrimination. A is defined as the ratio of the number of total packet loss correctly classified to the number of total packet losses for the evaluation of the accuracy of overall discrimination. The accuracy of flow TCP1 is shown in Table 1. The proposed PLC algorithm shows better accuracy when there can be more than one bottleneck in the network.

<table>
<thead>
<tr>
<th></th>
<th>Ac</th>
<th>Aw</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay trend scheme [6]</td>
<td>0.58</td>
<td>0.8</td>
<td>0.68</td>
</tr>
<tr>
<td>The proposed PLC</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
</tbody>
</table>

4.2. The Performance of the Wireless-Aware Congestion Control Algorithm

We use ns-2 as the simulation environment to compare the performance of the proposed algorithm with the congestion controls algorithms implemented in several TCP protocols. The performance metrics used here are throughput, network utility, and fairness index as defined in [13]. The packet size is set to 1000 bytes and queue size is set to 50 packets in these simulations.

4.2.1. Performance at Various Wireless Error Rates

The simulation topology is shown in Fig. 4 with link capacity and delay for each link indicated. There are two FTP flows that use the same TCP protocol. One flow TCP1 is from node S1 to node D1; the other flow TCP2 is from node S2 to node D2. The bottleneck with capacity 1.3Mb is the link between node N1 and node N2. The total simulation time is 100 seconds, and both of the two flows exist during the entire simulation time. When the simulation time reaches second 40, the wireless error model is activated to generate wireless losses. We compare total goodput, which is defined as the effective amount of data rate in application layer, of the proposed algorithm to several TCP variants: WestwoodNR, NewReno, Westwood, Westwood+, New Jersey, and Veno at different error rates, and the results are shown in Fig. 5. The proposed algorithm can discriminate wireless losses from congestion losses to avoid unnecessary performance degradation. Therefore, it outperforms other TCP protocols that also do not require congestion control support from the routers. The performance of the proposed algorithm is almost the same as the one of the TCP-New Jersey which differentiates wireless losses by the help of the intermediate routers to give a signal of congestion.
receivers at mobile hosts. The capacity for each link is generated according to heavy-tailed distribution in the range of 2Mbps and 10Mbps. The wireless error model with average error rate 0.1 is activated to simulate wireless channel error at second 40. Besides, there are 5 UDP flows with 500 Kbps as background traffic. The effective throughputs are shown in Table 4. The total goodput of the proposed algorithm is better than other TCP variants according to the simulation results.

We measure the data flow at the bottleneck link according to different flows separately, and the average values of the simulations at different error rates are listed in Table 2 and 3. In general, the proposed wireless-aware congestion control algorithm outperforms other TCP variants with respect to the throughputs and network utility. We can find that the proposed method shows better utility of the bottleneck link regardless of the wireless error rates. The results of fairness index of these methods are close. The proposed algorithm is better than the others, especially when the error rate is high. The main reason of the better performance is due to the increase of $cwnd$ after fast recovery ends as well as the increase of $ssthresh$ after timeout occurs if a wireless loss occurs.

### 4.2.2 Performance with Larger Topology

In this Section, we use Brite [15] topology generator to generate the network topology which is composed of 50 nodes, as shown in Fig. 6. There are 33 nodes (the square ones) in the core network and the others are leaf nodes. There are 3 WiFi base stations connected to the leaf nodes and there are 3 WiFi mobile hosts for each base station. Three leaf nodes are taken as senders to transmit TCP packets to their receivers at mobile hosts. The capacity for each link is generated according to heavy-tailed distribution in the range of 2Mbps and 10Mbps. The wireless error model with average error rate 0.1 is activated to simulate wireless channel error at second 40. Besides, there are 5 UDP flows with 500 Kbps as background traffic. The effective throughputs are shown in Table 4. The total goodput of the proposed algorithm is better than other TCP variants according to the simulation results.

4.2.2. Performance with Larger Topology

In this Section, we use Brite [15] topology generator to generate the network topology which is composed of 50 nodes, as shown in Fig. 6. There are 33 nodes (the square ones) in the core network and the others are leaf nodes. There are 3 WiFi base stations connected to the leaf nodes and there are 3 WiFi mobile hosts for each base station. Three leaf nodes are taken as senders to transmit TCP packets to their receivers at mobile hosts. The capacity for each link is generated according to heavy-tailed distribution in the range of 2Mbps and 10Mbps. The wireless error model with average error rate 0.1 is activated to simulate wireless channel error at second 40. Besides, there are 5 UDP flows with 500 Kbps as background traffic. The effective throughputs are shown in Table 4. The total goodput of the proposed algorithm is better than other TCP variants according to the simulation results.

We measure the data flow at the bottleneck link according to different flows separately, and the average values of the simulations at different error rates are listed in Table 2 and 3. In general, the proposed wireless-aware congestion control algorithm outperforms other TCP variants with respect to the throughputs and network utility. We can find that the proposed method shows better utility of the bottleneck link regardless of the wireless error rates. The results of fairness index of these methods are close. The proposed algorithm is better than the others, especially when the error rate is high. The main reason of the better performance is due to the increase of $cwnd$ after fast recovery ends as well as the increase of $ssthresh$ after timeout occurs if a wireless loss occurs.

#### Table 2 Performance comparisons over bottleneck link (error rate 0.06)

<table>
<thead>
<tr>
<th></th>
<th>The proposed</th>
<th>NewReno</th>
<th>WestwoodNR</th>
<th>Veno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Utility</td>
<td>0.94</td>
<td>0.72</td>
<td>0.84</td>
<td>0.72</td>
</tr>
<tr>
<td>Fairness</td>
<td>0.998</td>
<td>0.999</td>
<td>0.998</td>
<td>0.997</td>
</tr>
</tbody>
</table>

#### Table 3 Performance comparisons over bottleneck link (error rate 0.12)

<table>
<thead>
<tr>
<th></th>
<th>The proposed</th>
<th>NewReno</th>
<th>WestwoodNR</th>
<th>Veno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Utility</td>
<td>0.84</td>
<td>0.59</td>
<td>0.61</td>
<td>0.62</td>
</tr>
<tr>
<td>Fairness</td>
<td>0.998</td>
<td>0.999</td>
<td>0.997</td>
<td>0.997</td>
</tr>
</tbody>
</table>

5. CONCLUSION

A new packet loss classification algorithm is proposed with two-trend delay detection to differentiate congestion losses from wireless losses in the ambiguous region of ROTT distribution. A wireless-aware congestion control algorithm for TCP is further proposed. The proposed algorithm takes advantage of the packet loss classification to react correctly for real congestion situations. The algorithm adjusts the congestion window after fast recovery ends and increases the slow start threshold after timeout occurs if a wireless loss occurs.

---

**Fig. 4** Network topology used for different wireless error rates

**Fig. 5** Total goodput at different error rates. Note that New Jersey needs extra information from routers and it is not an end-to-end protocol.

**Fig. 6** The large scale network topology generated by Brite

**Table 4** Goodput at the receiver

<table>
<thead>
<tr>
<th></th>
<th>The proposed</th>
<th>NewReno</th>
<th>WestwoodNR</th>
<th>Veno</th>
</tr>
</thead>
<tbody>
<tr>
<td>M11</td>
<td>493.88</td>
<td>412.18</td>
<td>567.27</td>
<td>393.54</td>
</tr>
<tr>
<td>M12</td>
<td>486.23</td>
<td>406.27</td>
<td>362.68</td>
<td>387.72</td>
</tr>
<tr>
<td>M13</td>
<td>538.23</td>
<td>414.51</td>
<td>386.97</td>
<td>457.61</td>
</tr>
<tr>
<td>M21</td>
<td>563.19</td>
<td>481.07</td>
<td>515.52</td>
<td>428.9</td>
</tr>
<tr>
<td>M22</td>
<td>557.03</td>
<td>440.64</td>
<td>470.09</td>
<td>494.05</td>
</tr>
<tr>
<td>M23</td>
<td>556.28</td>
<td>422.75</td>
<td>543.64</td>
<td>355.19</td>
</tr>
<tr>
<td>M31</td>
<td>421.42</td>
<td>335.55</td>
<td>344.46</td>
<td>307.93</td>
</tr>
<tr>
<td>M32</td>
<td>408.44</td>
<td>306.43</td>
<td>417.09</td>
<td>379.98</td>
</tr>
<tr>
<td>M33</td>
<td>454.78</td>
<td>336.05</td>
<td>341.63</td>
<td>335.8</td>
</tr>
<tr>
<td>Total</td>
<td>4479.44</td>
<td>3555.42</td>
<td>3949.29</td>
<td>3540.69</td>
</tr>
</tbody>
</table>
New Jersey, which, unlike our approach, is not an end-to-end algorithm and TCP New Jersey needs router support. Although the results are encouraging, the further evaluation is desired to take the mobility of the mobile host and handoff between base stations into consideration, and also prove the efficiency in real wired-wireless network environments.

ACKNOWLEDGMENT

This work was supported in part by the Ministry of Economic Affairs of Taiwan under Grant 102-EC-17-A-03-S1-214, and by the National Science Council of Taiwan under Grant 101-2221-E-009-086-MY3.

REFERENCES