LOSS DIFFERENTIATION ALGORITHMS VS. CONGESTION CONTROL SCHEMES: DYNAMICS AND PERFORMANCE

Aleksandar Milenkoski[1] and Biljana Stojcevska[1]

[1]School of Computer Science and Information Technology, University American College, Skopje, Republic of Macedonia
{milenkoski,stojcevska}@uacs.edu.mk

ABSTRACT

This paper carefully analyzes the behaviour of different congestion control schemes when used in combination with Loss Differentiation Algorithm. Three types of congestion schemes are discussed: delay-based, bandwidth estimation and AIMD, with one TCP variant representing each congestion scheme. We simulated two network scenarios with diverse link and traffic properties and evaluated the congestion schemes with integrated Loss Differentiation Algorithm in each of them. The integrated Loss Differentiation Algorithm is ideal, i.e. it makes no errors in its judgement. The behaviour of the schemes is analyzed from aspect of: the properties of the employed mathematical functions, the effect of presence or absence of additional network load (reverse and background traffic), and the achieved throughput. The results show very diverse scene and pinpoint the importance of the careful and delicate design of the congestion avoidance action when a non-congestion loss is detected.

KEYWORDS

Loss Differentiation Algorithms, Congestion avoidance, Congestion control scheme

1. INTRODUCTION

With the increasing popularity of wireless networks, a few problems of the standard TCP implementation have become evident. Among them, the inability to cope with losses due to link errors has proven to be one of the greatest obstacles [1] when it comes to high throughput performance in networks with wireless links. Many TCP variants blindly reduce their congestion window because their TCP schemes always suppose that any loss is caused by network congestion. As a result, a great degradation in performance is noted.

As an answer to this problem, many Loss Differentiation Algorithms (referred to as LDAs from this point) have been designed. Their primary goal is to make a two-way decision: is a packet loss due to congestion, or due to corruption? It has been shown that losses due to link error can be detected both in the transport [2] and the link layer [3]. These two approaches have been accompanied by cross-layer designs [4], [5]. The varying efficiencies of the existing LDAs have been observed by several LDA evaluation papers [6], [7], [8]. Their evaluation methods usually include testing the LDA from aspect of accuracy and frequency of congestion loss prediction. The excellent study presented in [6] goes further and defines metrics for misclassification rate, stressing the possible effect on the performance of the protocol when the LDA makes misclassifications. Additionally, the same paper outlines the performance of the protocols put under evaluation in terms of throughput and fairness.

However, a certain trend has been noticed: very few papers explicitly define the way in which the used LDA is integrated into the TCP congestion control scheme. In most cases, it can be implicitly concluded that the undertaken action when a packet error loss is detected is simply avoiding the original congestion control response.

To fill the gap, this paper closely investigates the relationship between the LDA and the congestion control scheme in which it is integrated. Three different types of congestion schemes are investigated: AIMD scheme, bandwidth estimation scheme and delay-based scheme. We investigate the following representative protocols of these schemes: TCP NewReno [9], TCP Westwood [10] and TCP Vegas [11] respectively. In each scheme we integrate ideal LDA with zero misclassification rate. The performances of the protocols are compared in terms of throughput and briefly on fairness. We observed very closely the dynamics of the protocols and the impact of the way in which the LDA is implemented in the congestion scheme.

The paper is organized as follows: Section 2 contains the related work in this field. In Section 3 a brief overview of the underlying algorithms behind TCP NewReno, TCP Westwood and TCP Vegas is given; in Section 4 we describe the structure of the LDA in use, while in Section 5 the simulation environment is discussed. In Section 6 the performances and dynamics of the schemes is elaborated, in Section 7 we present guidelines for future work, and finally in Section 8 we give our conclusions.

2. RELATED WORK

Although proposing loss differentiation solutions with high precision, many of the published papers that present protocols aimed for operation in networks with high random loss do not implement change in the standard congestion schemes. The most utilized actions when wireless errors occur are no change at all or changing the reduction of the congestion window by a factor smaller than 1/2. For instance, the authors of [12] clearly state that no congestion window adjustment is needed when wireless errors are detected. Moreover, TCP Feno [13], based on its predecessor TCP Veno [14], simply reduces the window by 1/5 for each recognized wireless loss. The same action is undertaken by [15].

In contrary, [16] proposes congestion scheme designed for best efficiency when loss differentiator is used and analyzes in detail the performance of the scheme with and without error discriminator. The authors introduce congestion window cut policy in order to minimize unnecessary congestions caused by error discriminator mismatches. Additionally, they draw attention to the importance of designing adaptable and dynamic congestion schemes, suitable for LDA integration. Similarly, [17] proposes efficient congestion action when multiple non-congestion losses take place. The authors emphasize the possible harm to the network's performance caused by inadaptable congestion window actions combined with loss differentiator. In addition, [18] closely observes the impact of LDA accuracy on the performance of TCP. It concludes that accurate LDA and appropriate reaction to wireless loss is often not enough for improved performance, but that the LDA information should be used even in designing retransmission timeout recovery algorithms.

A proposal which implements flexible action triggered upon a detected wireless loss is presented in [19]: a switching mechanism based on queuing delay is used to decide the appropriate congestion action after a wireless loss. Normally, when wireless loss is detected the window is kept unchanged. But, if big queuing delay is present, the protocol treats the wireless loss as congestive and reduces the congestion window.

Following the arguments presented in [16] and [17], this research thoroughly analyzes the consequences of simple and not dynamic reaction to packet drops due to wireless errors.

3. CONGESTION SCHEMES

In this section we briefly overview the mathematical model used by the congestion schemes of TCP NewReno, TCP Westwood and TCP Vegas. The accent is put on the reaction of the TCP modification when *n* consecutive duplicate acknowledgements are received; therefore we closely observe the position where LDA is deployed.

3.1. TCP NewReno

TCP NewReno is an effective modification of the original congestion avoidance algorithm [20]. The modification is an improvement of the Fast Recovery phase. However, the congestion avoidance scheme is still an AIMD scheme, since the congestion window is increased additively and decreased in a multiplicative fashion.

When three duplicate acknowledgements are received by the sender, TCP NewReno halves the congestion window along with the Slow Start threshold (Eq. 1 and Eq. 2).

$$ssthresh = \frac{WindowSize}{2} \qquad (1)$$

$$WindowSize = \frac{WindowSize}{2} \qquad (2)$$

ssthresh is the Slow Start threshold, while *WindowSize* is the size of the congestion window at the moment of receiving three duplicate acknowledgements.

3.2. TCP Westwood

TCP Westwood uses bandwidth estimation in order to achieve admirable protocol performance in mixed wired-cum-wireless networks. One of the key advantages of this protocol over TCP NewReno is the adaptive adjustment of the congestion window. TCP Westwood estimates the available link bandwidth, based on Eq. 3 and Eq. 4:

$$SampleBwe = \frac{AckdSize}{AckdInterval} \quad (3)$$

$$Bwe = Bwe \times 0.9047 + (Bwe + LastSampleBwe) \times 0.0476 \qquad (4),$$

where *AckdSize* is the total size of the acknowledged windows, *AckdInterval* is the length of the time slot between the last received acknowledgement and the moment of bandwidth estimation,. Bandwidth estimation is performed at each received acknowledgement. *Bwe* is the new estimated bandwidth, and *LastSampleBwe* is the previous *SampleBwe*.

Finally, the congestion action is defined in Eq. 5:

$$ssthresh = \frac{(Bwe \times RTT_{\min})}{SegSize} \qquad (5).$$

Only if the current window size is greater then the new Slow Start Threshold, new window size is calculated (Eq. 6):

$$WindowSize = ssthresh \qquad (6).$$

RTT_{min} is the minimal RTT observed during the connection and *SegSize* is the length of the TCP segment, expressed in bits. It should be noted that TCP Westwood has successful mechanism for handling delayed and cumulative acknowledgements.

3.3. TCP Vegas

Similarly to TCP Westwood, TCP Vegas differs greatly from the AIMD scheme. The congestion avoidance actions in TCP Vegas can happen in two cases: one action takes place at the moment when new acknowledgement is received. The congestion window linearly decreases, increases or keeps its previous size. The pseudo code is given below.

```
Actual sending rate = Last window size / RTT for the last window

Expected = Current window size / minimalRTT

Difference = Expected – Actual

if (Difference < α)        linearly increase window;

if (Difference > β)        linearly decrease window;

else                       leave window unchanged;
```

minimalRTT is the minimal RTT observed during the connection.

The other action takes place when *n* duplicate acknowledgements are received. It manipulates the window in more aggressive fashion: the window may be halved, set to three fourths of its current size or directly set to size of 2. The way in which the congestion window will be changed depends on the number of the retransmissions of the packet which is lost. The pseudo code follows:

```
if (Current window size <= 3)    Current window size = 2;

else if(Number of retransmissions > 1)    Current window size = Current window size/2;

else Current window size = Current window size*3/4;
```

When it receives three duplicate acknowledgements, TCP Vegas also readjusts its timeout value: either it increases it two times when the number of transmissions is greater than one, or it increases it by one eighth of its value in any other case. The timeout value is crucial for deciding whether the sender should wait for three duplicate acknowledgements or retransmit the lost packet immediately.

4. THE IDEAL LDA

It has been previously stated that this research uses ideal LDA algorithm. In this section we briefly discuss the reasons for using such LDA and the way it is implemented.

There are two reasons for constructing an ideal LDA:

(a) it is the most appropriate way to investigate the congestion scheme – LDA relationship. Any LDA clearly has a certain misclassification rate, which could seriously alter the final results. Additionally, the architecture of each LDA invokes algorithm-specific inefficiencies. For example, [6] concludes that two loss differentiation schemes have classification problems when multiple streams share the same wireless link. The usage of ideal LDA guarantees a great deal of "*cleanliness*" in the simulation scenario; it certainly eliminates the chances for classification mistakes. Therefore, the advantages or disadvantages of integrated loss classification mechanism are unbiased from misclassifications;

(b) an ideal LDA certainly represents the final goal of the LDA authors. Each LDA has been designed with maximum classification rate in mind. Interestingly enough, this research clearly shows that many congestion schemes would have achieved better performance if the employed LDA made a certain number of mistakes. However, such improved performance may be safely considered as unexpected behaviour of the protocol when compared with the final goal of the creation of the loss differentiation scheme.

By the properties of the ideal LDA, it can be easily concluded that it does not use a specific algorithm. The design of loss differentiation algorithm with perfect classification is still the ultimate goal. The LDA which we use is implemented in two places of the ns-2 [21] code: the first one is in the used error model code. At each detected drop event the LDA writes the *packet sequence number – flow id* pair in a file. The written pair is chosen specifically to ensure uniqueness of the record of the dropped packet. The second part of the ideal LDA is implemented at the TCP sender's side. It consists of a function which searches the specific record file at each *n* duplicate acknowledgement event. If the combination of the acknowledged sequence number which triggered the loss action and the appropriate flow id is found in the record file, then the loss is classified as loss due to link error. Otherwise, the loss is considered to be caused by network congestion. Inspired by the previously discussed LDA evaluation papers, the action of the ideal LDA when loss due to corruption is detected is simply avoiding the original congestion action when three duplicate acknowledgements are received. In that way we observe the behaviour of the protocol in a situation when it does not activate its congestion reaction (Section 2) when there is no real congestion event present.

The design of the ideal LDA has the disadvantage of many reading and writing actions to a file which increases the time needed for a simulation set to finish. However, storing the *sequence number – flow id* pair in the ns-2 code has proven to be unreliable. The great number of simulation runs in order to achieve good statistical accuracy can easily drain the available system memory. Additionally, the complexity of the code makes the transfer of the information on the dropped packet from the error model to the TCP agent very hardly feasible. At the end, the more time-consuming process was chosen, since reliability must be kept at high level.

5. SIMULATION ENVIRONMENT AND METHODOLOGY

In order to investigate the behaviour of the congestion schemes in realistic manner, we simulated a number of different scenarios in ns-2 network simulator. We used tcp-eval [22] as traffic and topology generator. In this section we give detailed explanation of the simulated scenarios.

From aspect of general network congestion level, we created two different scenarios:

- **Scenario 1 - Not congested network**
  - Number of forward long-lived flows: 1
  - No reverse traffic
  - No background traffic
- **Scenario 2 - Congested network**
  - Number of forward long-lived flows: 20
  - Number of reverse long-lived flows: 5
  - Number of voice streaming flows: 5

- Number of forward video streaming flows: 5
- Number of reverse video streaming flows: 5
- Video streaming rate: 640 Kb/s
- Video streaming packet size: 840 bytes
- HTTP generation rate: 5 Kb/s

Scenario 1 was created to investigate the impact of LDA integration into the congestion scheme when the number of packet loss due to congestion is low and when the protocol of the long-lived flow does not compete with other flows. Therefore, all outside factors which could possibly disturb the efficient performance of the congestion scheme put under evaluation were eliminated. Scenario 2 represents more realistic scenario. Traffic in both directions is simulated, accompanied by additional background CBR UDP flows and TCP short-lived, ON/OFF flows. With this complex traffic scheme we create network situation which creates sudden congestion and decongestion, ACK compression, delayed and out-of-order acknowledgements, and eliminates possible traffic-phase effects [23].

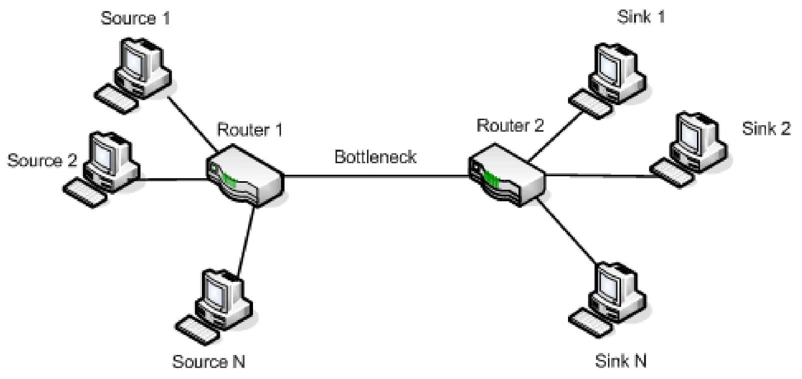


Figure 1. Single-bottleneck topology

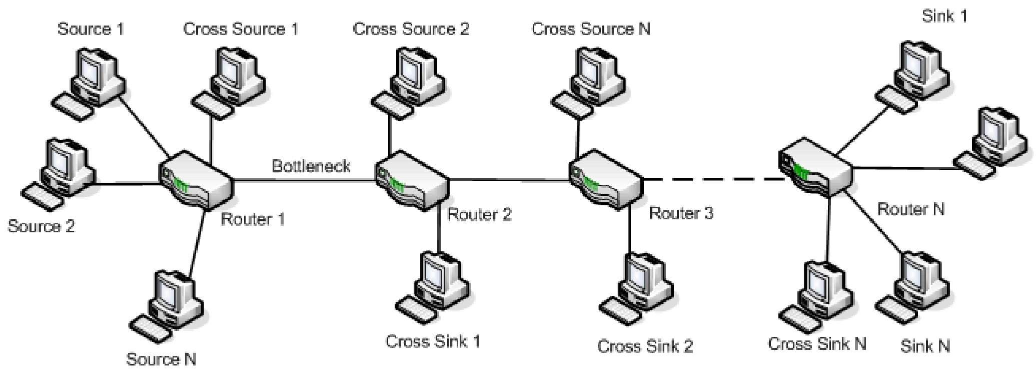


Figure 2. Multiple-bottleneck topology

The traffic scheme explained in Scenario 1 and Scenario 2 was deployed in different network topologies with various link characteristics. We simulated two network topologies: single-bottleneck (Fig. 1) and multiple-bottleneck topology (Fig. 2) for each traffic scenario. The multiple-bottleneck topology was modelled with three bottlenecks. Additionally, two queuing schemes were tested, FIFO and RED [24]. At the bottleneck queues a Bernoulli error model was used to drop packets. We tested the congestion schemes in a packet error rate interval of 0.01 to 0.09. The bottleneck bandwidth was set to 10 Mbps and the queue length was equal to the Bandwidth Delay Product (BDP). The non-bottleneck links had 1.5×10 Mbps bandwidth. The achieved statistical precision is 5% with 95% confidence interval. The simulation time was set to 100 sec. Careful examination of the protocols' dynamics showed that they reached stable state after 100 sec of traffic generation.

6. THE OBSERVATIONS

In this section we present the observations of the congestion schemes dynamics specified in Section 2. We compared the protocols' achieved throughput when FIFO and RED queuing schemes were deployed at the routers and concluded very similar results. Therefore, RED is not thoroughly discussed in the following sections. We also measured and compared the fairness of the specified TCP modifications with the fairness of their LDA-integrated counterparts and noticed very small changes. In all investigated scenarios, maximum improvement and worsening of the fairness of 2.50% and 3%, respectively, was noted. Hence, we concentrate our analyzing effort on the protocols' achieved throughput when FIFO queuing disciplines are deployed.

The values of the number of actions presented in the tables in this paper are averaged values obtained from a series of simulations.

6.1. TCP NewReno

Since TCP NewReno does not receive feedback from the network to adjust its sending rate, the impact of the LDA integration is the simplest of all discussed schemes to observe. It becomes clear that the most important factor of the protocol's behaviour is the number of performed different reactions to packet losses. Therefore, we observe three different categories of actions: actions of increasing, decreasing or keeping the congestion window unchanged. We further investigate the number of such actions and try to find correlation between their number and the achieved throughput.

Fig. 3a and Fig. 3b depict the percentage of improvement of TCP NewReno with integrated LDA as function of packet error rate. The new protocol is referred to as TCP NewRenoLDA. Fig. 3a and Fig. 3b show the throughput improvement in single-bottleneck and multiple-bottleneck network topology respectively, in Scenario 1. Figure 3a clearly shows that the highest improvement is achieved at packet error rate of 0.01, followed by almost linear performance degradation to 0.09 error rate. The linear characteristics of the improvement are mainly due to the simple congestion reaction used by TCP NewReno, accompanied by the monotonous dynamics of the single-bottleneck topology.

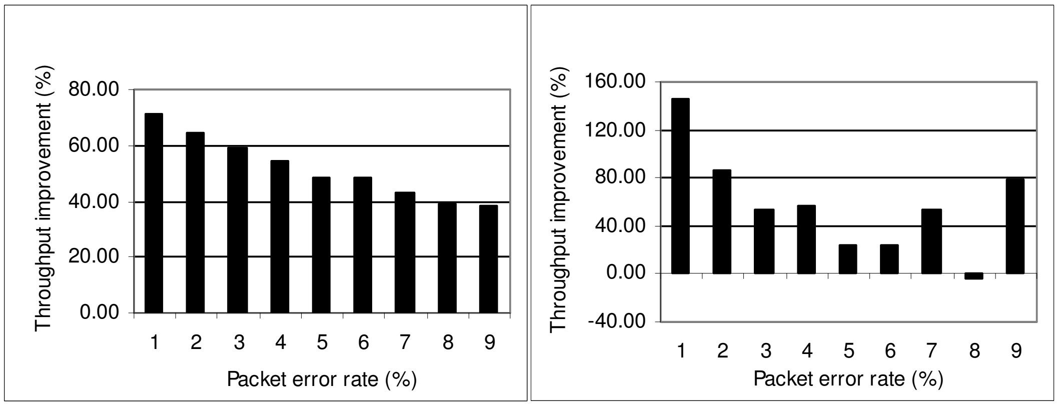


Figure 3. Percentage of throughput improvement of NewRenoLDA in a) single-bottleneck topology b) multiple-bottleneck topology in Scenario 1.

By observing the number of congestion events at the NewReno agent presented in Table 1, we concluded that the degradation of the throughput is caused by too aggressive action when packet loss due to corruption is detected. Keeping the window unchanged seems to be more aggressive action as the number of corruption events increases. However, the overall outcome is positive - the minimum value of improvement is almost 40%.

Table 1. Number of losses due to corruption at the sender's side (single-bottleneck topology, Scenario 1)

Protocol	**Packet error rate**	**No. of congestion events**
NewReno	0.01	124
NewRenoLDA	0.01	121
NewReno	0.09	116
NewRenoLDA	0.09	125

We noted the increased number of congestion events at packer error rate (referred to as PER) of 0.09 (9%). When PER is 0.01 the NewRenoLDA agent detected slightly less congestion events than original NewReno. In contrary, when PER is 0.09, the situation is vice versa – the NewRenoLDA agent has experienced more losses than the NewReno sender. It can be concluded that keeping the window unchanged causes additional congestion in the network for high error rates, therefore decreasing the overall efficiency.

Figure 3b depicts the different behaviour of the protocol in a multiple-bottleneck topology. We observe a decreasing trend until PER of 0.05 is reached. For PER values greater than 0.06, the general trend of increasing performance is not stable. In order to explain the sudden decrease at PER of 0.08, we observed steep forward throughput degradation at both NewReno and NewRenoLDA agents when packet error loss is introduced in the multiple-bottleneck topology (Fig. 4). Hence, a big improvement or worsening of the throughput of NewRenoLDA is calculated from a very small overall number of packets. An important factor of the diminished throughput is the highly discrete and not adaptive behaviour of the investigated AIMD congestion scheme. As more time is spent in retransmitting corrupted packets, the congestion window shrinks or expands at low rate, resulting in bad network capacity utilization. Additionally, the time spent per retransmission in a three bottleneck topology is considerably long, which directly decreases the achieved throughput in a fixed time interval of 100 sec.

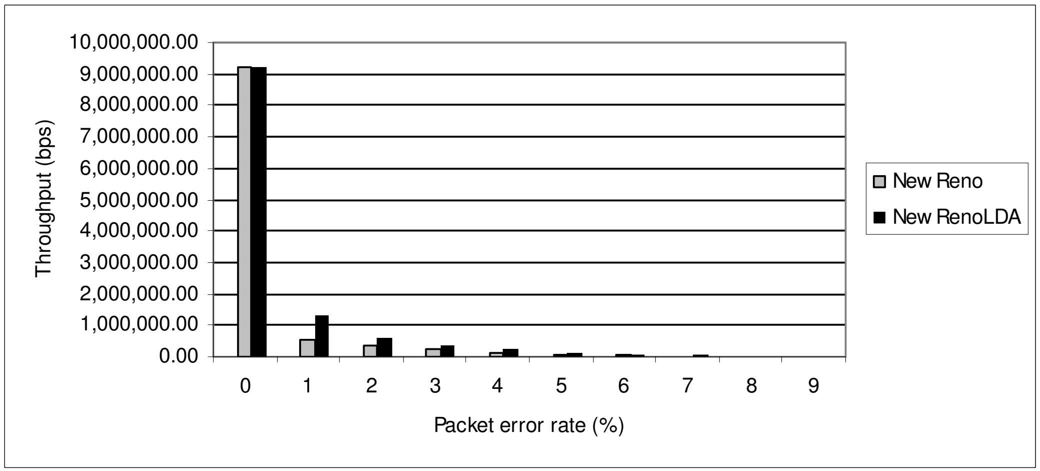
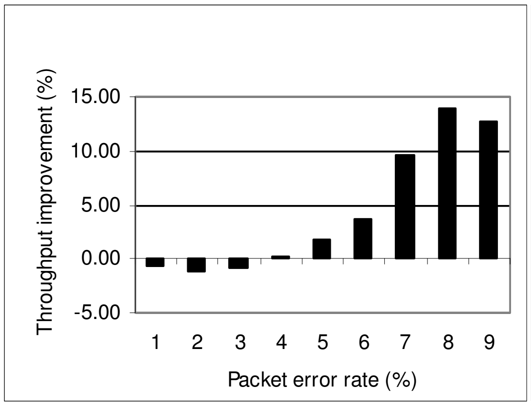


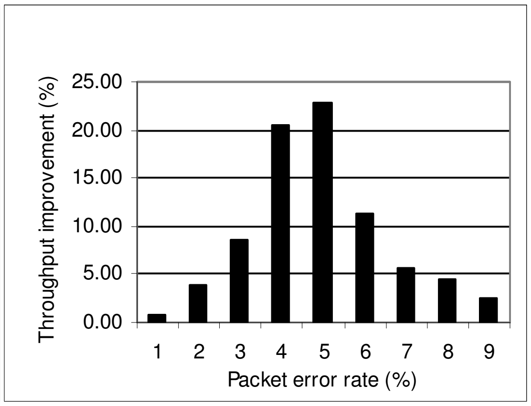
Figure 4. Throughput of forward long-lived flow in multiple-bottleneck topology

Fig. 3b is a proof that the increased number of performed congestion actions is not always a cause for lowered protocol performance of the NewRenoLDA sender. We noticed trend of increased congestion actions at the NewrenoLDA agent as PER has greater value in multiple-bottleneck technology, similar to the results presented in Table 1. Therefore, the number of congestion epochs in the network can perform different roles in different topologies – it can be a factor for either increased or decreased total throughput.

Fig. 5a and Fig. 5b display the throughput improvement of NewRenoLDA in Scenario 2.



a)



b)

Figure 5. Percentage of throughput improvement of NewRenoLDA in a) single-bottleneck topology b) multiple-bottleneck topology in Scenario 2.

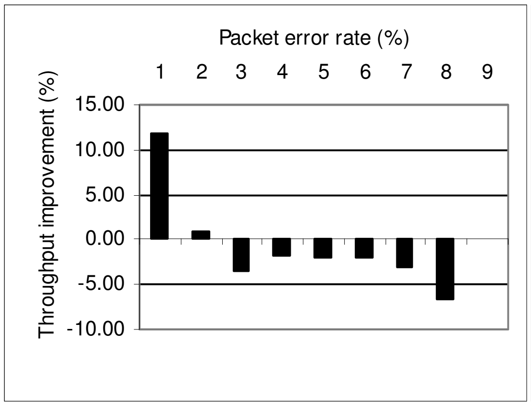
We observed almost complementary protocol behaviour when compared with the results presented in Fig. 3a and Fig. 3b. From Fig. 5a it can be concluded that the LDA integration does not seem to make any significant throughput improvement until PER of 0.06 is reached. The advantage of using LDA is evident after the *threshold* of 0.06 error rate. We noticed that the average bottleneck utilization decreases as the value of PER increases. Considering the high congestion state of the network, keeping the congestion window unchanged pays off after a certain freeing of the network capacities is performed by high PER values. Furthermore, by conducting a set of additional experiments, we noticed that the value of the mentioned error rate *threshold* is strongly dependant on the congestion level of the network. Moreover, we recorded greater number of congestion events at the NewRenoLDA sender than the NewReno sender at PER of 0.02 and 0.03, which was certainly not the case in Scenario 1 (Table 1). An interesting fact is that the same PER *threshold* is present in the multiple-bottleneck topology, but this time the *threshold* marks the beginning of a throughput decreasing trend (Fig. 5b). Because the bottleneck link capacity is relieved of stress by dropping packets, we recorded performance improvement until the *threshold* is reached. Congestion window analysis showed that when the *threshold* is passed, the averaged congestion window of NewRenoLDA is too large when compared with the averaged congestion window of NewReno. Hence, NewRenoLDA is too aggressive for the congestion state of the network, so the performance is decreased in almost the same fashion as it was increased.

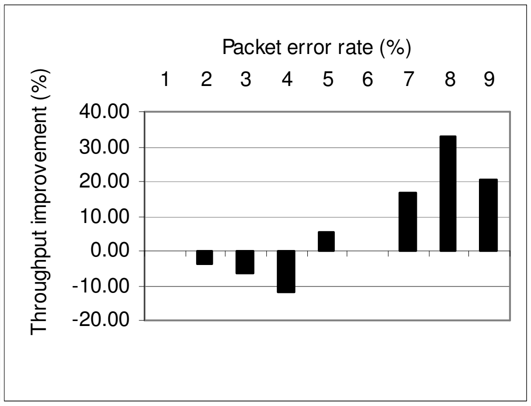
6.2. TCP Westwood

TCP Westwood uses bandwidth estimation to control its sending rate at each received acknowledgement. In [10] the authors state that the bandwidth estimation of the network is performed in both slow start and congestion avoidance phase. Therefore, additional network phenomenon as ACK compression or ACK delay may perform much important role in the overall protocol behaviour. In contrary to TCP NewReno, we observe much greater continuality in the protocol's congestion scheme. We argue that the most appropriate way to implement LDA may be more complex than the implementation in the AIMD scheme. Currently, the integrated ideal LDA simply does not perform the bandwidth estimation operation and the possible decreasing of the congestion window when three duplicate acknowledgements are received.

In [6] it has been stated that TCP Westwood strongly relies on the TCP acknowledgement scheme, so we further broaden our study and include analysis of the acknowledgements in order to completely understand the behaviour of bandwidth estimation algorithms.

Fig. 6a and Fig. 6b show the throughput improvement of TCP WestwoodLDA (TCP Westwood with integrated LDA) when compared with TCP Westwood in Scenario 1.





a) b)

Figure 6. Percentage of throughput improvement of WestwoodLDA in a) single-bottleneck topology b) multiple-bottleneck topology in Scenario 1.

According to the action undertaken by the ideal LDA, as the PER increases, the number of bandwidth estimations performed by the WestwoodLDA agent decreases. In other words, for big values of PER, we disrupt the continuality of bandwidth estimation to a large scale. It has proved to be of more benefit for the simulated multiple-bottleneck topology.

When single-bottleneck topology is simulated, the value of *AckdInterval* (Section 2.2) takes smaller values when compared with the values of *AckdInterval* when multiple-bottleneck topology is used in simulations. The greater value of *AckdInterval* in multiple-bottleneck topology is due to the longer path which the acknowledgement must travel. We measured averaged values of 0.004194 and 0.005499 of *AckdInterval* for single and multiple-bottleneck topology respectively. We also recorded constant value of both *AckdSize* and *RTTmin* in the both topologies. Therefore we conclude that the value of *AckdInterval* has the greatest impact over the continuous bandwidth estimation and finally at the achieved throughput. *AckdInterval* and *SampleBwe* are reverse proportional; hence we recorded smaller values of *SampleBwe* in the multiple-bottleneck topology than in the single-bottleneck topology. The reverse proportionality of *AckdInterval* and *SampleBwe*, accompanied by the different magnitudes of *AckdInterval* in the two topologies, lead to the almost inverted behaviour of WestwoodLDA presented in Fig. 6a and Fig. 6b. It should be noted that, according to Eq. 4, greater values of *SampleBwe* may lead to consecutive greater values of *Bwe*. The increasing value of *Bwe* results in more aggressive WestwoodLDA sender, which proves to be inefficient in single-bottleneck topology after PER of 0.02.

We observe the two extremes in Fig. 6a, at PER of 0.01 and PER of 0.08. At PER of 0.01 we record improvement of TCP WestwoodLDA of 11.91%, while at PER of 0.08 degradation of 6.58% was noticed. Also, two extremes are depicted in Fig. 6b - at PER of 0.04 (degradation of 11.91%) and 0.08 (improvement of 33.24%). We prove the variability in aggressiveness of the Westwood and the WestwoodLDA sender from another point of view – the number of packet losses due to congestion recorded at the sender in Scenario 1(Table 2 and Table 3).

Table 2. Number of losses due to corruption at the sender's side (single-bottleneck topology, Scenario 1)

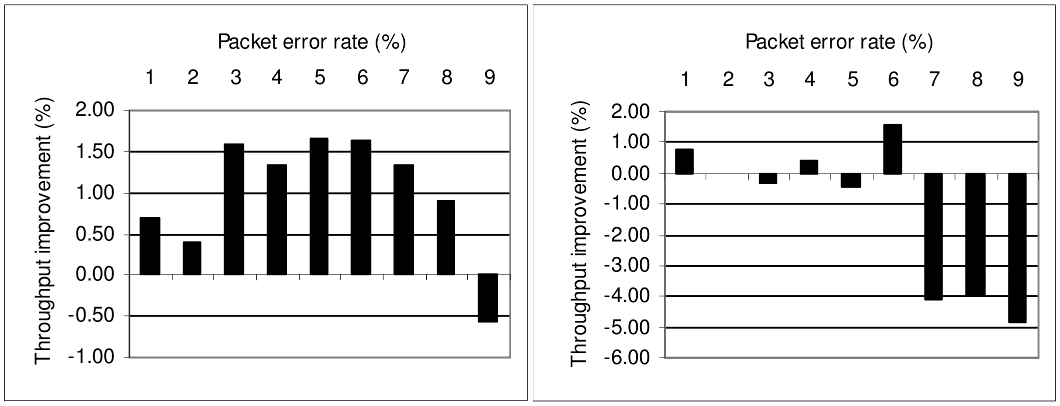
Protocol	**Packet error rate**	**No. of congestion events**
Westwood	0.01	994
WestwoodLDA	0.01	795
Westwood	0.08	306
WestwoodLDA	0.08	130

Table 3. Number of losses due to corruption at the sender's side (multiple-bottleneck topology, Scenario 1)

Protocol	**Packet error rate**	**No. of congestion events**
Westwood	0.04	249
WestwoodLDA	0.04	72
Westwood	0.08	50
WestwoodLDA	0.08	4

The decreased number of congestion events recorded at both the Westwood and WestwoodLDA agents indicates that the multiple-bottleneck topology is much less prone to congestion than the single-bottleneck topology. The WestwoodLDA agent recorded only 4 packet losses due to congestion in a simulation time of 100s at PER of 0.08. The small number of congestion events is an indicator that WestwoodLDA did not decrease the congestion window almost 46 times more than the Westwood sender. Moreover, WestwoodLDA did not cause additional congestion in the network. From our observations of the Westwood and Vegas LDA variants we concluded that due to the network adaptive behaviour, non-AIMD schemes with integrated LDA never experience more congestion packet losses than the original congestion schemes. This is certainly not the case with the AIMD scheme used by TCP NewReno (Table 1).

Fig. 7a and Fig. 7b show the throughput improvement of TCP Westwood over TCP WestwoodLDA when Scenario 2 is simulated. We observed general throughput improvement of WestwoodLDA in the single-bottleneck topology, in contrary of the worsened achieved throughput depicted in Fig. 6a. However, the throughput improvement is very small; the maximum improvement of 1.65% is noted at PER of 0.05. We also observed inverted behaviour of the protocol in multiple-bottleneck topology (Fig. 6b and Fig. 7b). The window size aggressiveness of the WestwoodLDA sender produces increased averaged throughput only because of the great number of congestion losses. The congestion window is reduced many times during the simulation due to heavy congestion, so occasional keeping of the size of the congestion window brings only a small benefit until PER of 0.09 is reached (Fig. 7a). However, that is not the case when multiple-bottleneck topology is simulated, since the WestwoodLDA sender is not so aggressive due to high values of *AckdInterval*. The values of *AckdInterval* are additionally increased because of the three routers where heavy congestion occurs. Opposite of the WestwoodLDA sender, the original Westwood agent is prone to adjusting its congestion window to smaller sizes. Fig. 7b shows that keeping the congestion window unchanged results in worsened throughput due to heavy congestion, for PERs greater than 0.07.



a) b)

Figure 7. Percentage of throughput improvement of WestwoodLDA in a) single-bottleneck topology b) multiple-bottleneck topology in Scenario 2.

An important factor of the non-linear characteristics of Fig. 7a and Fig. 7b may be the fact that TCP Westwood (and hence TCP WestwoodLDA) relies on the TCP acknowledgement mechanism. The reverse traffic accompanied by the cross traffic created ACK compression of great scale. In Fig. 8 we present the recorded values of *AckdInterval* in WestwoodLDA sender in single-bottleneck topology in 2 seconds of simulation. As expected, we observed even greater ACK compression in the multiple-bottleneck topology.

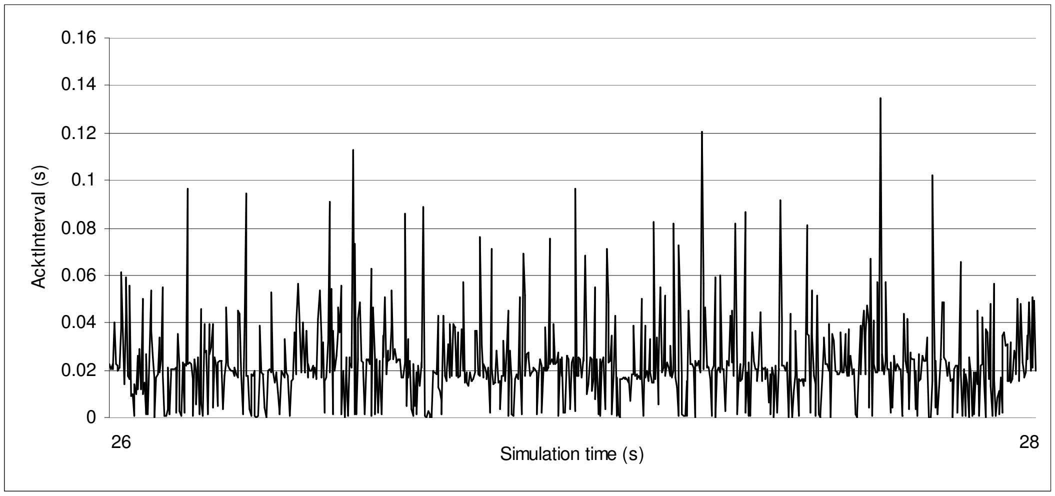
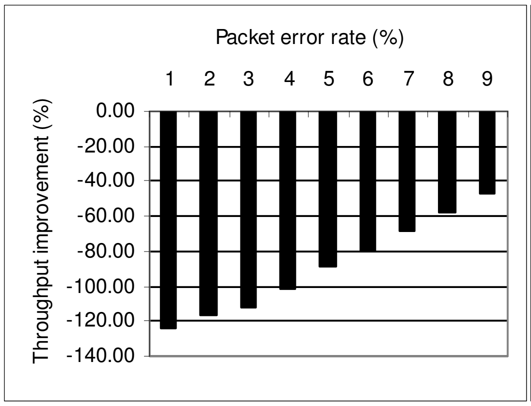


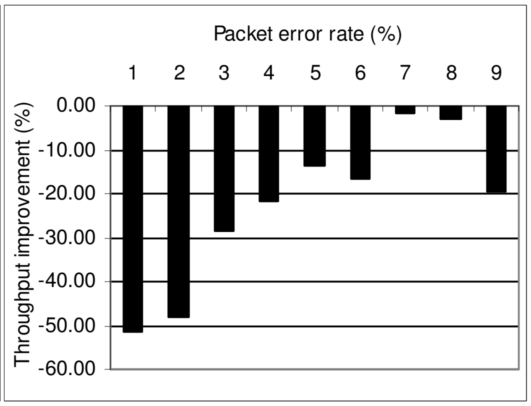
Figure 8. ACK compression recorded at the WestwoodLDA sender

The consecutive low values of *AckdInterval* are clear indicators that a series of acknowledgement packets are received in a very tight time slot, often followed by large delay until next acknowledgement is received. Therefore, if a packet loss due to congestion happens after a long ACK compression event, the Westwood and WestwoodLDA agents set their slow start threshold and possibly the congestion window to a slightly greater value. If packet loss due to corruption happens sometime after that sequence of events, the congestion window is not readjusted by WestwoodLDA. That creates high probability for new congestion events. We observe non-linear characteristics of the averaged throughput because of the unpredictability of such events.

6.3. TCP Vegas

The congestion scheme of TCP Vegas is the most complex scheme so far, because the congestion window is modified at many places (Section 2). Therefore, we analyze three kinds of actions performed by the Vegas (and the VegasLDA) sender: Action of increasing, decreasing or keeping the congestion window unchanged. Decreasing the congestion window is performed when three duplicate acknowledgements are received; all other actions take place at each received acknowledgement. Once again, the ideal LDA simply avoids the decreasing of the congestion window and resetting the timeout value when three duplicate acknowledgements are received.





a) b)

Figure 9. Percentage of throughput improvement of VegasLDA in a) single-bottleneck topology b) multiple-bottleneck topology in Scenario 1.

The behaviour of TCP VegasLDA in Scenario 1 is not positive. We observe linear improvement of the negative throughput as the PER increases in Fig. 9a. The same trend is evident in Fig. 9b until PER of 0.09 is reached. However, the overall achieved throughput is greatly worsened. We explain the worsened throughput by observing two possible factors:

a) we noticed that the VegasLDA (and the Vegas) sender received small values of RTTs in the not utilized network in Scenario 1. Because of the reverse proportionality of the values of *Actual* and *RTT for the last window* (Section 2.3) we recorded high values for *Actual* and small values of *Difference*. From the algorithm deployed in TCP Vegas we concluded that small values of *Difference* led to bigger chances for increased congestion window;

b) although small, the number of detected erroneous packets disrupts the process of continuous increase of the timeout value which Vegas implements. It should be noted that the timeout value is crucial for faster retransmits. Hence, for small enough timeout values, a Vegas sender would never have to wait for three duplicate acknowledgements in order to perform retransmit [11]. Since VegasLDA avoids prolonging its timeout value, the agent produced smaller timeout values than the original Vegas agent and hence he never waited for three duplicate acknowledgements. That behaviour is too aggressive for single-bottleneck (Fig. 9a) and multiple-bottleneck topology (Fig. 9b).

In Table 4 and Table 5 we present the number of different actions performed upon the congestion window in single and multiple-bottleneck topology, respectively:

Table 4. Number of actions upon the congestion window at the sender's side (single-bottleneck topology, Scenario 1)

	Vegas/ 0.01 PER	**VegasLDA/ 0.01 PER**	**Vegas/ 0.09 PER**	**VegasLDA/ 0.09 PER**
NKU	53	12	112	73
NI	1012	764	491	386
ND	11	0	5	0
NOC	166	0	183	0

Table 5. Number of actions upon the congestion window at the sender's side (multiple-bottleneck topology, Scenario 1)

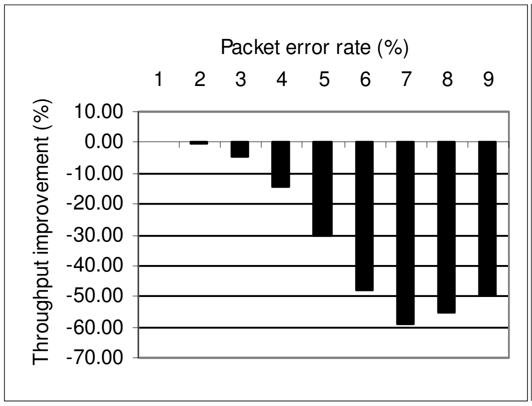
	Vegas/ 0.07 PER	**VegasLDA/ 0.07 PER**	**Vegas/ 0.09 PER**	**VegasLDA/ 0.09 PER**
NKU	50	38	27	35
NI	214	187	120	162
ND	0	0	0	0
NOC	20	0	7	0

NKU stands for Number of Keeping the window Unchanged, NI for Number of Increasing the window, ND for Number of Decreasing the window and NOC for Number of Original Congestion actions. We observed the values of PER where Fig. 9a and Fig. 9b depicted maximum improvement or worsening of the throughput, or when sudden change in the trend was noticed. As previously stated, the VegasLDA agent never manages to perform the original congestion action of decreasing the congestion window after three duplicate acknowledgements, making its behaviour too aggressive (note that NOC is equal to 0).

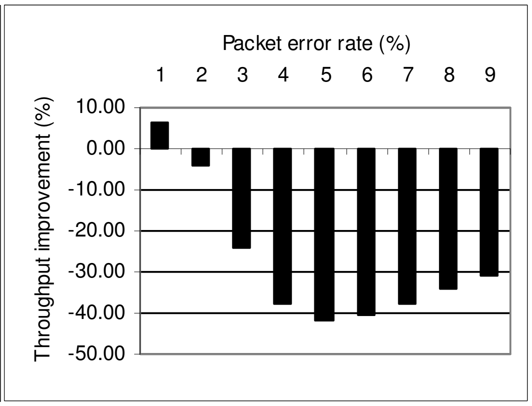
At multiple-bottleneck topology we observe deviation from the linear trend of negative throughput improvement. This deviation can be explained by the change in the behaviour of the agent noticeable at the PER of 0.09 in multiple-bottleneck topology when compared with the behaviour of the same PER in single-bottleneck topology. In Table 4 we observe smaller values of NKU and NI at the VegasLDA agent than the original Vegas sender. In Table 5, however, the situation is opposite. The additional aggressiveness of the VegasLDA in multiple bottleneck topology results in worsening of the achieved throughput.

In Fig. 10a and Fig. 10b we present the improvement of the throughput of the TCP VegasLDA protocol in Scenario 2. Similar to TCP WestwoodLDA (Section 5.2) we observe better performance of Vegas LDA in a congestion network scenario. The heavy congestion limits the congestion window to low sizes, never allowing excessive aggressiveness of the VegasLDA agent. The presented results are complementary with the results depicted in Fig. 9a and Fig. 9b; for high error rates the VegasLDA agent in Scenario 1 records throughput improvement, in contrary to the VegasLDA agent in Scenario 2.

Until PER of 0.03 is reached, we observe almost same behaviour of both Vegas and VegasLDA in the single-bottleneck topology (Fig. 10a) and even slight improvement (PER 0.01) in the multiple-bottleneck topology (Fig. 10b). We recorded trend of lowering bottleneck utilization as PER increases. Therefore, the VegasLDA sender managed to partially regain its high window sizes which we observed in Scenario 1; hence the complementarities of the presented charts in this section.



a)



b)

Figure 10. Percentage of throughput improvement of VegasLDA in a) single-bottleneck topology b) multiple-bottleneck topology in Scenario 2.

Furthermore, we observed very similar value of the sum of original congestion actions and detected packets lost due to errors at VegasLDA and number of congestion actions at Vegas sender (Table 6). Hence, the PER of 0.03 is a PER threshold after which the balance is disturbed and the throughput of VegasLDA is significantly worsened – we observe that for high PER rates the VegasLDA sender does not decrease its congestion window (NOC is equal to 0).

Table 6. Number of actions upon the congestion window at the sender's side (single-bottleneck topology, Scenario 2)

	Vegas/ 0.01 PER	**VegasLDA/ 0.01 PER**	**Vegas/ 0.07 PER**	**VegasLDA/ 0.07 PER**
Detected losses due to errors	0	894	0	2530
NOC	1045	48	4833	0

Since specific congestion action is performed at each receipt of acknowledgement, the effect of the compressed acknowledgements has been expressed through the presented numbers of different performed actions. Also, the small time slots between the compressed acknowledgements does not cause drastic changes in the behaviour of TCP VegasLDA since the variables *Expected* and *Actual* do not rely on it; they rely on the overall RTT.

7. FUTURE WORK

This work can be continued in several directions: First, a more detailed analysis can be performed on each discussed protocol. The effect of the faster retransmits algorithms deployed in both TCP WestwoodLDA and TCP VegasLDA can be thoroughly discussed. For TCP VegasLDA, analysis on the behaviour of RTT should also be taken into consideration, since its algorithm relies greatly on the RTT values. Second, few more representatives can be picked out of each type of congestion scheme in order to find common behaviour patterns. If similar characteristics are discovered, then we could consider designing LDA schemes appropriate for each type of congestion mechanism.

8. CONCLUSIONS

We observed very different and complex dynamics of the three investigated congestion schemes. For example, we noticed that TCP Westwood produced inverted throughput achievement in congested and not congested networks. Moreover, TCP VegasLDA produced complementary throughput improvement charts for dumb-bell and parking-lot topology, making the output even more diverse.

At the AIMD scheme employed by TCP NewReno and TCP NewRenoLDA we observed linear characteristics of the throughput improvement charts. We conclude that the greatest impact on the behaviour of TCP NewReno and TCP NewRenoLDA has the number of congestion losses – number of error losses relationship.

We witnessed unpredictability of the behaviour of the bandwidth estimation scheme used in TCP Westwood. We showed that changes in the inter-arrival times of the acknowledgements cause improvement or worsening of the throughput in rather unpredictable ways. Additionally, we disclosed the sensitivity of TCP Westwood and TCP WestwoodLDA of the *AckdInterval* variable. We also concluded that occasional events like ACK compression can cause excessive aggressiveness in the WestwoodLDA agent. However, in order to achieve improved behaviour of WestwoodLDA in different network scenarios, we conclude that the LDA must take into consideration the estimated bandwidth and make possible changes after the estimation is performed.

As expected, we observed huge sensitivity of the delay-based scheme of the timeout variable. Avoiding increasing the timeout value when packer loss due to error is detected results in greatly worsened throughput. We also noticed that if we do not increase the timeout value, even for a few times, the whole dynamics of Vegas is disturbed – it never manages to increase it again in a time interval of 100 sec. Therefore, we conclude that the timeout value must be increased no matter of the type of the detected loss.

It is evident that there is no single way of integrating LDA into a congestion scheme. We have often witnessed inverted and complementary behaviour of the bandwidth estimation and delay-based protocols. That leads to the conclusion that in complex congestion schemes we might need several ways of integrating LDA in one congestion scheme, each way best for different network situations. However, constructing an LDA which performs different set of actions in one congestion scheme might be cumbersome – a delicate switching mechanism would have to be designed.

REFERENCES

[1] G. Xylomenos, G.C. Polyzos, P. Mähönen, M. Saaranen, “TCP Performance Issues over Wireless Links,” IEEE Communications Magazine, vol. 39, no. 4, pp. 52-58, April 2001.

[2] C. Paras, J.J. Garcia-Luna-Acevez, “Differentiating Congestion vs. random loss: a method for improving TCP performance over wireless links,” Proceedings of IEEE WCNC, pp. 90-93, 2000.

[3] A.M. Sakib, F.B. Lugman, “Improving TCP Performance over wired-wireless networks,” Computer Networks, vol. 51, no. 13, pp. 3799-3811, Sept. 2007.

[4] S. Lohier, Y Ghamri Doudane, G. Pujolle, "Cross-Layer Differentiation Algorithms to improve TCP Performance in WLANs," Telecommunication Systems, vol. 36, no. 1, pp. 61-72, Nov. 2007.

[5] Y. Ji-Hoon, "Cross-Layer Explicit Link Status Notification to Improve TCP Performance in Wireless Networks," EURASIP Journal on Wireless Communications and Networking, vol. 2009, Article ID 617818, 15 pages, 2009. doi:10.1155/2009/617818.

[6] S. Cen, P.C. Cosman, G.M. Voelker, "End-to-end Differentiation of congestion and wireless losses," IEEE/ACM Transactions on Networking, vol. 11, no. 5, pp. 703-717, Oct. 2003.

[7] S. Biaz, N.H. Vaidya, "Distinguishing Congestion Losses from Wireless Transmission Losses: A Negative Result," Proceedings of the International Conference on Computer Communications and Networks, p. 722, 1998.

[8] A. Boukerche, G. Jia, R.W.N. Pazzi, "Performance evaluation of packet loss differentiation algorithms for wireless networks," Proceedings of the 2nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks, pp. 50-52, 2007.

[9] S. Floyd, T. Henderson, A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 3782, April 2004.

[10] C. Casetti, M. Gerla, S. Mascolo, M.Y. Sanadidi, R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," Proceedings of ACM Mobicom, pp. 287-297, 2001.

[11] L. Brakmo, L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," IEEE Journal on Selected Areas in Communication, vol. 13, no. 8, pp. 1465-1480, Oct. 1995.

[12] A. Seddik-Ghaleb, Y. Ghamri-Doudane, S.M. Senouci, "TCP WELCOME TCP variant for Wireless Environment, Link losses, and COngestion packet loss ModEls," Proceedings of First International Communication Systems and Networks and Workshops, pp. 243-250, 2009.

[13] H. Jae-Hyun, Y. See-Hwan, Y. Chuck, "TCP Feno: Enhancement for higher accuracy of loss differentiation over small buffer heterogeneous networks," Proceedings of IEEE 34th Conference on Local Computer Networks, pp. 249-252, 2009.

[14] C.P. Fu, S.C. Liew, "TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks," IEEE Journal on Selected Areas in Communications, vol. 21, no. 2, pp. 216-228, Feb. 2003.

[15] K.W. Lien, J.S. Reeve, Y.J. Lee, "Improving TCP performance over wireless networks," Proceedings of International Symposium on Telecommunications, pp. 424-428, 2008.

[16] M. Alnuem, J. Mellor, R. Fretwell, "New Algorithm to Control TCP Behavior over Lossy Links," Proceedings of the 2009 International Conference on Advanced Computer Control, pp. 236-240, 2009.

[17] M. Alnuem, J. Mellor, R. Fretwell, "Tcp multiple drop action for transmission errors," The 9th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, 2008.

[18] M.Y. Park, S.H. Chung, "Analyzing effect of loss differentiation algorithms on improving TCP performance," Proceedings of The 12th International Conference on Advanced Communication Technology, pp. 737-742, 2010.

[19] P. Papadimitriou, V. Tsaoussidis, C. Zhang, " End-to-end loss differentiation for video streaming with wireless link errors," Telecommunication Systems, vol. 43, no. 3, pp. 295-312, 2009, April 2010.

[20] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit," RFC 2001, Jan. 1997.

[21] "The Network Simulator – ns-2,". Available: http://www.isi.edu/nsnam/ns/.

[22] "An NS2 TCP Evaluation Tool,". Available: http://labs.nec.com.cn/tcpeval.htm.

[23] S. Floyd, V. Jacobson, "On Traffic Phase Effects in Packet-Switched Gateways," Internetworking: Research and Experience, vol. 3, no. 3, pp. 115-156, Sept. 1992.

[24] S. Floyd, V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, vol. 1, no. 4, pp. 397-413, Aug. 1993.

Authors

Aleksandar Milenkoski received his Diploma in Electrical Engineering at Faculty of Electrical Engineering and Information Technology Skopje and is currently an MSc candidate at University American College Skopje. He currently works as teaching assistant in University American College Skopje where he teaches computer network related subjects. His general interests are simulation methods, loss differentiation in wireless and ad hoc networks and network modeling.

Biljana Stojcevska works as a teaching assistant at the School of Computer Science and IT at the University American College Skopje. She received her MCs degree in the field of Computer networks at the Institute of Informatics, Faculty of Natural Sciences and Mathematics in Skopje. The areas of her interest are Computer Networks, Network Congestion Management and Operating Systems.