Journal of Computer Science 7 (12): 1824-1830, 2011
ISSN 1549-3636

Congestion Control in Wireless Ad Hoc Networks by Enhancement of Transmission Control Protocol

[1]N. Premalatha and [2]A.M. Natarajan
[1]Department of CSE,Kongu Engineering College,
Anna University of Technology, Tamil Nadu, India
[2]Bannari Amman Institute of Technology,Sathyamangalam, India

Abstract: Problem statement: Transmission Control Protocol is the Internet's most widely used transport control protocol. TCP's strength lies in the adaptive nature of its congestion avoidance and control algorithm and its retransmission mechanism. In TCP Vegas rerouting (a path), which change the propagation delay of the connection and this may be able to result in a substantial decrease in through put. An enhanced algorithm for TCP in wireless Ad Hoc networks is needed to obtain a fairer share of the available bandwidth, tackle re-routing problems and solve the problems associated with older TCP Vegas flows. **Approach**: TCP-Vegas uses an Estimation of the propagation delay, base RTT, to adjust its window size and it is very important for a TCP Vegas connection to be able to have an accurate estimation. **Results:** One of the issues is rerouting (a path), which change the propagation delay of the connection and this may be able to result in a substantial decrease in through put. The issue identified with TCP Vegas was reinvestigated and to address them, a modification to TCP Vegas's congestion avoidance algorithm is proposed. **Conclusion:** This modified TCP enhancement algorithams is shown to obtain a fairer share of the available bandwidth, tackle re-routing problems and solve the problems associated with older TCP Vegas flows.

Key words: Congestion avoidance, Performance Evaluation, packet retransmissions, Transmission Control Protocol (TCP), Vegas connection, available bandwidth, higher throughput, Round Trip Time (RTT)

INTRODUCTION

The Transmission Control Protocol (TCP) was proposed and implemented to prevent the future congestion collapses. TCP had gone through several phases of improvement and many new features such as fast retransmit and fast recovery have been added.

High bits error of wireless channel, various kinds of mistakes of link layers and the asymmetric link and route will all affect TCP performance.Reference (Kirubanand and Palaniammal, 2011) study mainly focuses on M/M (a,b)/1 markovian model with adaboost algorithm and user selection algorithms to find performance on wired and wireless technologies in terms of service rate, arrival rate, Expected waiting time and Busy period. When comparing the wireless technologies with wired technologies in term of inter-arrival and inter-service time it has been found that the wireless technologies are better. Reference shows that the degradation of TCP performance mainly occurs in MAC layer where competition and conflict cause the instability and deterioration of TCP performance.

For TCP AIMD is optimal and is a necessary condition for a congestion control mechanism to be stable. Although TCP was initially designed and optimized for wired networks, the growing popularity of wireless data applications has lead third generation wireless networks such as CDMA2000 and UMTS networks to extend TCP to wireless communications as well. The initial objective of TCP was to efficiently use the available bandwidth in the network and to avoid overloading the network (and the resulting packet losses) by appropriately throttling the senders' transmission rates. Network congestion is deemed to be the underlying reason for packet losses. Consequently, TCP performance is often unsatisfactory when used in wireless networks and requires various improvement techniques. A key factor causing the unsatisfactory performance is that the radio link quality in wireless networks can fluctuate greatly in time due to channel fading and user mobility, leading to a high variability of transmission time and delay. High delay variability is also due to retransmissions at the link level and use of opportunistic schedulers that give preferential service to

Corresponding Author: N. Premalatha, Department of CSE Kongu Engineering College, Perundurai, Tamil Nadu, India

terminals with good radio links, thus causing additional delay to terminals with relatively poor radio quality. Furthermore, large delay variability can be incurred during handoff from one cell to a neighboring cell. A form of high delay variability, referred to as delay spike, is a sudden, drastic increase in delay for a particular packet or a few consecutive packets, relative to the delay for the preceding and following packets. When TCP is employed for data transport in such environments, highly variable RTTs and delay spikes can induce spurious timeouts, although the involved packet actually is not lost but simply delayed. Regardless of the actual cause, when a timeout occurs, the TCP congestion window is reduced to 1, thus unnecessarily degrading the throughput.

In recent techniques a dynamic power adjustment protocol is needed for sending the periodical safety message. It is based on the analysis of the channel status depending on the channel congestion and the power used for transmission.

Wireless mobile networks have many weaknesses related to bit error, network congestion and weak signals that cause segments losses as well as handoff process. For this reason, wireless mobile network TCP cannot distinguish between losses caused by these weaknesses or by the handoff process. So in handoff case, segments losses will trigger congestion control algorithms that reduce the TCP connection's throughput performance. An improved performance envisaged if these control schemes adjust dynamically to the varying ABR bandwidth capacity in a stochastic manner instead of conventional deterministic approach. The performance difference between setting explicit rate deterministically for transmitting ABR sources and doing the same stochastically using a learning automaton is of particular interest.

In this study, we have proposed different versions of TCP called TCP NewVegas and EnhancedVegas which achieves higher efficiency and causes much fewer packet retransmissions and which is not biased against the connections with longer round trip times.

TCP vegas: TCP Vegas use a sophisticated bandwidth estimation scheme to proactively gauge network congestion (Ahn, 1995; Brakmo *et al*., 1994). The reasoning is that when the actual throughput of a connection approaches the value of the Estimated maximum throughput, it may not be utilizing the intermediate routers and buffer space efficiently and hence should increase the flow rate. On the other hand, when the actual throughput is much less than the Estimated throughput, the network is likely to be congested and hence the connection should reduce the flow rate.

TCP Vegas use a conservative algorithm to decide how and when to vary its congestion window. It assumes that an increase in the RTT value is always due to the presence of competing traffic and rules out other possibilities like rerouting which is not a reasonable assumption. TCP Vegas can become unstable in the presence of network delays and proposes modifications to stabilize the system.

During the congestion avoidance phase, a TCP Vegas sender does:

cwnd = cwnd + 1 if diff <(α /baseRTT)
cwnd = cwnd if (α/baseRTT) <= diff <= (β/baseRTI')
cwnd = cwnd- 1 if (β /baseRTI') < diff,

Where:

diff	=	expected rate-actual rate >= 0, by definition
Expected rate	=	data in transit/baseRTT
baseRTT	=	The minimum of all measured RTTs, typically, the RTI' of a packet when the router queue is empty or when the flow is not congested (in seconds)
Actual cite	=	(next send sequence number-segment timed)/ average R'IT
RTT	=	Observed or actual round trip time (in seconds)
α, β =		Some constant thresholds

Features of TCP vegas: First, TCP Vegas sets Base RTT to the smallest measured round trip time and the Estimated throughput will be computed (Ahn, 1995; Brakmo *et al*., 1994). Second, with each packet being sent, Vegas records the sending time of the packet by checking the system clock and computes the Round Trip Time (RTT) by computing the elapsed time before the ACK comes back. It then computes Actual throughput using the Estimatedd RTT.Studies (Srijith *et al*., 2000; Srijith, 2003) have shown that TCP Vegas performs badly when it competes for bandwidth and is unfair towards older connections, does not handle rerouting well and has fairness bias against connections with higher bandwidth.

The reasoning is that when the actual throughput of a connection approaches the value of the Estimated maximum throughput, it may not be utilizing the intermediate routers and buffer space efficiently and hence should increase the flow rate. On the other hand, when the actual throughput is much less than the Estimated throughput, the network is likely to be congested and hence the connection should reduce the flow rate.

In this paper, the discussions are on these problems and proposed a solution to them. The issues identified with TCP Vegas were reinvestigated and to address them, a modification to TCP Vegas's congestion avoidance algorithm is proposed.

This paper proposes some modification to the congestion avoidance mechanism of TCP Vegas by three different methods. These modified algorithms obtains a fairer share of the available bandwidth, tackle re-routing problems and solve the problems associated with older TCP Vegas flows.

By decomposing TCP Vegas into its individual algorithms and addressing the effect of each of these algorithms, performance had shown that the congestion avoidance mechanism of Vegas has only a minor influence on throughput while at the same time being responsible for the issues identified with the protocol.

NewVegas algorithm: The main idea behind TCP NewVegas is that rather than fixing static values, they be made dynamically changeable and adaptive. At the start of a connection, the variables will have a fixed value. These values are then changed dynamically depending on the network conditions. Reference (Boutremans and Le Boudec, 2000) in the paper, give the idea of selection of α and β values Another way of looking at this modification is that we are trying to bring the network probing capability. While slow start and congestion recovery algorithms of NewVegas are the same as that of Vegas, we propose congestion window grow.

Wireless Ad Hoc networks are not infrastructure networks In the present implementation of TCP NewVegas, the values of the variables are increased and decreased together to maintain their relationship with each other, as in the original implementation. Here a small value of difference need not necessarily imply that the bandwidth utilization is poor. It might be that the dynamically changing value of a variable has grown to a large value because of which when congestion occurs, even a small throughput can still make difference numerically less than a variable.

TCP NewVegas improve TCP Vegas in two aspects: fairness and quick adaptation to the network condition. Based on competitiveness for newcomer flows, speed of acknowledgement returns and acceleration of speed, three revisions are proposed to improve the performance of TCP Vegas.

Here, the values of α and β are fixed and the effect of congestion control is limited. NewVegas algorithm is presented for the improvement of Vegas, it makes α and β value adjust automatically. This method can improve the congestion control mechanism, so the NewVegas algorithm can adapt to the change of the network automatically. The improvement of the algorithm is mainly displayed in congestion avoidance period.

The slow start and congestion recovery algorithms of NewVegas are the same as that of Vegas. However, NewVegas uses the modified congestion avoidance mechanism.

Terms used:

Th(t) = Actual throughput at time t

Th(t-rtt) = Actual throughput at previous rtt

The definitions of Estimated_rate, actual_rate and diff are the same as those in Vegas:

```
if β > diff > α {
if Th(t) > Th(t-rtt) {
cwnd = cwnd +1
α= α+1, β= β+1
}
else if Th(t) <= Th(t-rtt) {
no update of cwnd, α, β
}
}
else if diff < α {
if α >1 and Th(t) > Th(t-rtt) {
cwnd = cwnd +1
}
else if α >1 and Th(t) < Th(t-rtt) {
cwnd = cwnd –1, α= α-1, β= β-1
}
else if α ==1
cwnd = cwnd+1
}
else if diff > β {
cwnd= cwnd-1, α= α-1, β= β-1
}
else {
no update of cwnd, α, β
}
```

Even though diff >α, the throughput has been increasing. This indicates that the network is not fully utilized and that network bandwidth is still available. Hence, the sending rate can be increased, to probe the network.

As throughput is increasing over time, diff is decreasing. Hence α are increased to help congestion window grow. Here, they are preventing the connection from making use of the available present

implementation, α are increased and decreased at the same time to maintain their relationship with each other, as in the original implementation.

In NewVegas, a small value of diff needs not necessarily imply that the bandwidth utilization is poor. It might be that the dynamically changing value of α has grown to a large value and when congestion occurs, even a small throughput can still make diff less than α. Hence cwnd and the inflated α and β needs to be decreased.

EnhancedVegas algorithm: An another approach called EnhancedVegas algorithm for GEO satellite networks is also discussed in this paper to improve the performance of Ad Hoc networks. The improvement idea of EnhancedVegas algorithm comes from Vegas algorithm and NewVegas algorithm; it is the harmonization of two algorithms. During the course of congestion avoidance, the α and β values of this algorithm can be adjusted automatically, but the adjustment strategy of the values is different from NewVegas algorithm. Under the three main conditions of congestion avoidance, the target of EnhancedVegas algorithm and the target of Vegas algorithm are mostly same. That is to say, when $diff<\alpha$ the congestion window will be increased; if it can't be increased, it will be kept unchanged; and when $diff>\beta$ the congestion window will be decreased or kept unchanged; but when $\alpha<diff<\beta$ the congestion window will be kept constant or increased.

When $diff<\alpha$ congestion window will be increased or kept constant. The reason lies in that if $diff<\alpha$, it is already means that the Estimated throughput is too low, the actual network is comparatively expedite, so we shouldn't decrease congestion window under this circumstances; we should increase congestion window or keep congestion window constant.

When $diff>\beta$, the congestion window will be decreased or kept constant; the reason lies in that if the $diff>\beta$, it is already means that the Actual throughput is too high, the actual network condition is not very expedite, so we shouldn't increase congestion window under this circumstances; we should decrease congestion window or keep congestion window constant.

When the $\alpha<diff<\beta$, the congestion window will be kept constant or increased; the reason lies in that if the $\alpha<diff<\beta$, it is already means that the Actual throughput is just right, the actual network condition is general, so we shouldn't decrease congestion window under this circumstances; we should keep congestion window constant or increase congestion window. Under these circumstances, congestion window may also be increased, the purpose of this action is to let EnhancedVegas algorithm has the stronger ability to make use of bandwidth, attain to higher throughput.

At the slow start stage, improvement measure is to adjust congestion window in every RTT interval. The purpose of this action is to make EnhancedVegas algorithm more sensitive to the topology change of ad hoc network.

This algorithm can be described as follows:

Terms used:

Th(t) = Actual throughput at time t
Th(t-rtt) = Actual throughput at previous rtt

If $\alpha < diff < \beta$

If (Th(t) > Th(t-rtt))

```
if β > diff > α {
    if Th(t) > Th(t-rtt) {
    cwnd = cwnd +1
    α= α+1, β= β+1
    }
    else if Th(t) <= Th(t-rtt) {
    no update of cwnd, α, β
    }
    }
    else if diff < α {
    if α >1 and Th(t) > Th(t-rtt) {
    cwnd = cwnd +1
    }
    else if α >1 and Th(t) < Th(t-rtt) {
    cwnd = cwnd –1, α= α-1, β= β-1
    }
```

If $diff < \alpha$

If(α>1&& Th(t) > Th(t-rtt))
then { *cwnd*=*cwnd*+1;
$\alpha=\alpha+1$;
$\beta=\beta+1$; }

If(α>1&& Th(t) <= Th(t-rtt))
then *cwnd* = *cwnd*;

If(α==1)
then *cwnd* = *cwnd* + 1;

If $diff > \beta$

If (Th(t) <= Th(t-rtt))
then {
cwnd=*cwnd*-1;
if*(cwnd<2)*
then {*cwnd*=2};

if(α>1)
then { α=α-1;
β=β-1}
}

If (Th(t) >Th(t-rtt))
then *cwnd* = *cwnd*;

The merits of EnhancedVegas algorithm: In this part, we attempt to point out the merits of EnhancedVegas algorithm more thoroughly. Firstly, we can explain that EnhancedVegas algorithm is superior to Vegas algorithm and NewVegas algorithm in throughput. Reference (Samios and Vernon, 2003) analyzed characteristic of Vegas algorithm, presented approximate formula of TCP throughput as follows:

$$\text{Throughput} \propto \frac{\beta}{\text{RTT} - \text{baseRTT}} \quad (1)$$

In above formula, RTT is the average round trip time of a TCP connection; baseRTT is the minimum round trip time of this TCP connection; β is a parameter of Vegas algorithm. We approximatively consider that the TCP throughput of EnhancedVegas and NewVegas algorithms can be expressed as formula (1) also.

From above formula (1), we can get a conclusion that EnhancedVegas algorithm is superior to Vegas algorithm in throughput, because EnhancedVegas algorithm may have big value of β.

We can explain that EnhancedVegas algorithm is superior to NewVegas algorithm in throughput also. EnhancedVegas algorithm has more algorithm enhancement than NewVegas algorithm, thus EnhancedVegas algorithm can dispose more complex network conditions than NewVegas algorithm did. In ad hoc networks, EnhancedVegas algorithm can adapt to the change of network conditions rapidly, it can also tackle the route change and re-route process successfully. Especially, the frequent change of ad hoc network topology may cause TCP congestion window *(cwnd)* becoming one or two frequently. When network condition becomes straightway, in comparison with Vegas algorithm and NewVegas algorithm, EnhancedVegas algorithm can increase its congestion window more rapidly and then it can get more throughput.

Secondly, we can compare the complexity of the EnhancedVegas, Vegas and NewVegas algorithms as follows: our analysis is only on the numbers of embranchments of each algorithm. Because the workload of each algorithm embranchment is approximately same, the operation is increasing or decreasing *cwnd* by 1; thus the complexity of the algorithms rest with the numbers of algorithm embranchments. During congestion avoidance, Vegas algorithm has three embranchments, NewVegas algorithm has seven embranchments and EnhancedVegas algorithm has nine embranchments. This means that EnhancedVegas algorithm has the maximal complexity of three algorithms. Though EnhancedVegas algorithm has the maximal space complexity, above three algorithms are nearly same in time complexity. The reason lies in that each network condition does not need traversing every algorithm embranchments. At any network conditions, TCP only executes one or two algorithm embranchments to transmit its data; the time of algorithm execution is nearly same. The little increments of space complexity about EnhancedVegas algorithm will not affect normal function of ad hoc nodes also. In one words, the increments of complexity about EnhancedVegas algorithm was little and acceptable, if EnhancedVegas algorithm will also bring the increments of TCP throughput.

From above analysis, we can get a conclusion that EnhancedVegas algorithm has many advantages in comparison with Vegas algorithm and NewVegas algorithm, at the price of little increments of algorithm complexity.

Network environments and simulation results:
Network environments: The simulation tool is NS2 software (UCN/LBL/VINT, 2004).
The simulation environment is set up as follows:

Physical layer: Two kinds of environments. First, the area is of 500m×500 m; second, the area is of 1000×1000 m. Set up 20 move nodes in each kind of area. Propagation model is TwoRayGround; the distance of effective communication of each node is 250 meters. The link bandwidth is 2Mbps. The bit error ratio of wireless channel is configured to 0.02.
Data link layer: Utilize 802.11 protocols.

Network layer: The routing algorithm is chosen as DSDV routing algorithm.

Transport layer: Divide 20 nodes into 10 groups, each group has two nodes. Set up 10 independent TCP connections. The size of TCP segment is 1000 Bytes. We have simulated Vegas, EnhancedVegas and NewVegas algorithm.

Application layer: Choose persistent FTP as the data source of transport layer.

Simulation time: 100 sec.

Simulation results: The comparison of algorithm performance is based on the average throughput of each connection. The scene of each average speed should produce 10 times repeatedly for smoothing the influence of accidental factor and then get the average of 10 times results.

The result of our simulation shows that the EnhancedVegas algorithm is superior to the Vegas and NewVegas algorithm in throughput. Though EnhancedVegas algorithm uses the NewVegas idea of adjustment parameter dynamically for reference, the throughput of EnhancedVegas is higher NewVegas at all speed conditions. The throughput of EnhancedVegas is higher Vegas also.

In addition, in order to analyze the quality of different algorithms more deeply, we can also compare the transmission efficiency of various congestion control algorithms, namely the ratio of received TCP segment numbers to the sent TCP segment numbers. The analysis purpose of this ratio is to utilize ad hoc network's precious bandwidth resource effectively, because this ratio reflects the number of discarded TCP segment.

In 500×500 m area, 20 mobile nodes simulation results indicate that, under this environment, the EnhancedVegas algorithm is superior to Vegas and NewVegas algorithm in throughput and the difference of TCP transmission efficiency is not in evidence. Figure 1 draws the curve of TCP throughput as a function of node speed and Fig. 2 draws the curve of the ratio of received segment number to sent segment number as a function of node speed. With the increment of node speed, the throughput is increasing first, decreasing later. But the TCP transmission efficiency is digressive all the while. In 500m×500m area, the performance change is mild.

In 1000×1000m area, 20 mobile nodes simulation results indicate that, under this environment, the EnhancedVegas algorithm is superior to Vegas and NewVegas algorithm in throughput. EnhancedVegas algorithm is nearly as same as NewVegas algorithm in TCP transmission efficiency. Figure 3 draws the curve of TCP throughput as a function of node speed and Fig. 4 draws the curve of the ratio of received segment number to sent segment number as a function of node speed. With the increment of node speed, both the throughput and the TCP transmission efficiency are degrades all the while. In 1000×1000 m area, the TCP performance degradation is severe.

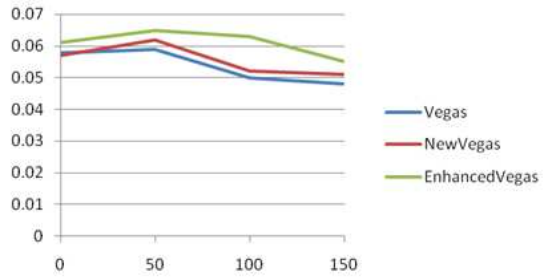

Fig.1: TCP throughput as a function of speed. (500m×500m)

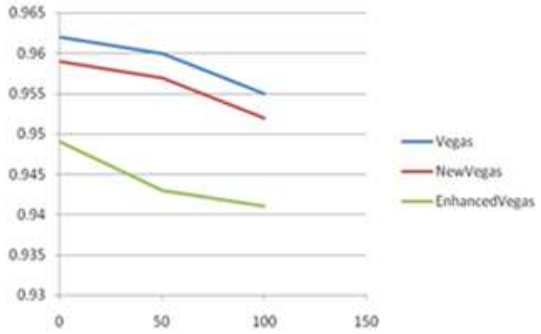

Fig 2: Received - sent ratio as a function of node speed. (500×500 m)

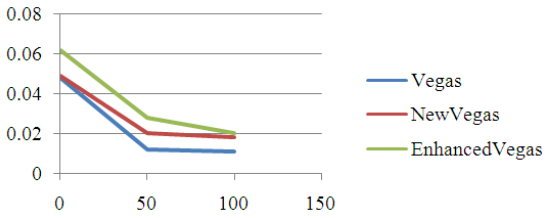

Fig 3: TCP throughput as a function of node speed. (1000×1000 m)

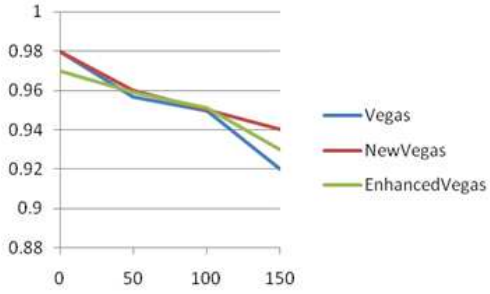

Fig. 4. Received - sent ratio as a function of node speed (1000×1000 m)

Summary and concluding remarks: In this paper, we have discussed a few issues of TCP Vegas. We have shown that TCP Vegas could cause a strange behavior when there is a rerouting in the network and connections do not detect such change. We have demonstrated that a simple scheme that updates base RTT when the round trip delay is consistently much larger than base RTT results in a much better performance for the connections that experience change in propagation delays. We have also shown that TCP Vegas could lead the network to a persistent congestion if connections start at different times when the network is congested.

TCP Vegas was proposed to go beyond the earlier work on TCP congestion control. Its performance was seen to be better than TCP Reno in terms of throughput and retransmissions. However, various problems associated with TCP Vegas have been identified. These deficiencies have been a deterrent in using TCP Vegas widely in the Internet. In this paper we examined the problems of TCP Vegas in detail and proposed a new algorithm (called TCP-Vegas). The main idea of the new algorithm is that instead of assigning static values for the protocol parameters, they are allowed to change in real time, allowing the connection to utilize the available bandwidth fully.

In future, we will try to improve the algorithm which utilizes the available bandwidth efficiently and effectively.As an improvement of TCP congestion control algorithm, the performance of Vegas3 algorithm gives us satisfaction results; it indicates that the measure of improvement is right and reasonable. The performance exaltation of the algorithm shows that, during congestion avoidance, improvement measure to divide communication conditions of network into more particular states, can suit the actual communication conditions of the ad hoc networks, this kind of detail depiction is more reasonable. And to take special counter measure for special communications conditions, to practice the different alpha and beta adjustment strategy, can also adapt actual variety of the network state. This also expresses that, to keep the mainly outline of Vegas3 algorithm in accordance with the Vegas algorithm, is a reasonable improvement project.

Only in the transport layer to improve the TCP congestion control protocol, is still a kind of effective way. The exaltation of the algorithm performance shows that, the simulation of the improvement research on Vegas algorithm has positive meaning; the Vegas algorithm still has the potential of the improvement.

REFERENCES

Ahn, J.S., P.B. Danzig, Z. Liu and L. Yan, 1995. Evaluation of TCP vegas: Emulation and experiment. Proceedings of the Conference on Applications, Technologies, Architectures and Protocols for Computer Communication (ATAPCC'95), ACM, New York, NY, USA, 185-195. DOI: 10.1145/217382.217431

Boutremans, C. and J.Y. Le Boudec, 2000. A note on the fairness of TCP Vegas. Proceedings of the Inernational Zurich Seminar on Broadband Commununication, Feb. 15-17, IEEE Xplore Press, Zurich, Switzerland, pp. 163-170. DOI: 10.1109/IZSBC.2000.829247

Brakmo, L.S., S. O'Malley and L.L. Peterson, 1994. TCP vegas: New techniques for congestion detection and avoidance. ACM SIGCOMM Comput. Commun. Rev., 24: 24-35. DOI: 10.1145/190809.190317

Kirubanand, V.B. and S. Palaniammal, 2011. Study of performance analysis in wired and wireless network. Am. J. Applied Sci., 8: 826-832. DOI: 10.3844/ajassp.2011.826.832

Samios, C., M.K. Vernon, 2003. Modeling the throughput of TCP vegas. ACM SIGMETRICS Performance Evaluat. Rev., 31: 71-81. DOI: 10.1145/885651.781037