AN ENHANCEMENT AND AN IMPLEMENTATION OF TCP VEGAS OVER MANETS

[1]THAKORE MITESH, [2]VANDANA VERMA

[1]Merchant Engineering College, Mehsana- Visnagar Road, Basna, India
[2]Ganpat University Kherva, Department of Computer Science
Email: mitesh.thakore@gmail.com, Vav01@ganpatuniversity.ac.in

Abstract: We examine the performance of the TCP protocol. In addition, work on using TCP (Transmission Control Protocol) to provide reliable data transmission has been performed for the purpose of smooth integration with the wired Internet. In the wired Internet, TCP-Vegas is a well-known transport protocol which takes into account existing network conditions. However, TCP-Vegas cannot be directly applied to MANETs because a route change can make the BaseRTT used over a previous path obsolete. It has been shown in the literature that the congestion control mechanism of TCP reacts adversely to packet losses due to temporarily broken routes in wireless networks. Through simulations using ns-2, we observed that TCP-Vegas-Adhoc outperforms the standard TCP-Vegas protocol especially under high mobility scenarios over both reactive and proactive ad hoc routing protocols such as AODV and OLSR.

Keywords: MANET, Wireless Adhoc Network, Performance Parameters, Retransmission, Slow Start

I. INTRODUCTION OF MANETS

Wireless cellular systems have been in use since 1980s. We have seen their evolutions to first, second and third generation's wireless systems. These systems work with the support of a centralized supporting structure such as an access point [1]. The wireless users can be connected with the wireless system by the help of these access points, when they roam from one place to the other. The adaptability of wireless systems is limited by the presence of a fixed supporting coordinate. It means that the technology cannot work efficiently in that places where there is no permanent infrastructure. Easy and fast deployment of wireless networks will be expected by the future generation wireless systems. This fast network deployment is not possible with the existing structure of present wireless systems. Recent advancements such as Bluetooth introduced a fresh type of wireless systems which is frequently known as mobile ad-hoc networks. Mobile ad-hoc networks or "short live" networks control in the nonexistence of permanent infrastructure. Mobile ad hoc network offers quick and horizontal network deployment in conditions where it is not possible otherwise. Ad-hoc is a Latin word, which means "for this or for this only." Mobile ad hoc network is an autonomous system [2] of mobile nodes connected by wireless links; each node operates as an end system and a router for all other nodes in the network. A wireless network is a growing new technology that will allow users to access services and information electronically, irrespective of their geographic position. Wireless networks can be classified in two types [2]: infrastructure network and infrastructure less (ad hoc) networks. Infrastructure network consists of a network with fixed and wired gateways. A mobile host interacts with a bridge in the network (called base station) within its communication radius. The mobile unit can move geographically while it is communicating. When it goes out of range of one base station, it connects with new base station and starts communicating through it. This is called handoff. In this approach the base stations are fixed. A Mobile ad hoc network is a group of wireless mobile computers (or nodes); in which nodes collaborate by forwarding packets for each other to allow them to communicate outside range of direct wireless transmission. Ad hoc networks require no centralized administration or fixed network infrastructure such as base stations or access points, and can be quickly and inexpensively set up as needed.MANET is an autonomous group of mobile users that communicate over reasonably slow wireless links. The network topology may vary rapidly and unpredictably over time, because the nodes are mobile. The network is decentralized [1], where all network activity; including discovering the topology and delivering messages must be executed by the nodes themselves. Hence routing functionality will have to be incorporated into the mobile nodes. MANET is a kind of wireless ad-hoc network and it is a self-configuring network of mobile routers (and associated hosts) connected by wireless links the union of which forms an arbitrary topology. The routers, the participating nodes act as router, are free to move randomly and manage themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet [1].

a. Characteristics of MANETS

Mobile ad hoc network nodes are furnished with wireless transmitters and receivers using antennas, which may be highly directional (point-to-point), unidirectional (broadcast), probably steerable, or

some combination of there. At a given point in time, depending on positions of nodes, their transmitter and receiver coverage patterns, communication power levels and co-channel interference levels, a wireless connectivity in the form of a random, multi-hop graph or "ad hoc" network exists among the nodes [1]. This ad hoc topology may modify with time as the nodes move or adjust their transmission and reception parameters.

II. TCP Vegas in MANETs

TCP Vegas is a new design for TCP that was introduced by Brakmo et al [2,3]. TCP Vegas includes a modified retransmission strategy(compared to TCP Reno) that is based on fine-grained measurements of the round-trip time (RTT) as well as new mechanisms for congestion detection during slow-start and congestion avoidance.

TCP Reno's congestion detection and control mechanisms use the loss of segments as a signal that there is congestion in the network. TCP Reno has therefore no mechanism to detect the incipient stages of congestion before losses occur and hence cannot prevent such losses. Thus, TCP Reno is reactive, as it needs to create losses to find the available bandwidth of the connection. On the contrary, TCP Vegas's congestion detection mechanism is proactive [4], that is, it tries to sense incipient congestion by observing changes in the throughput rate. Since TCP Vegas infers the congestion window adjustment policy from such throughput measurements, it may be able to reduce the sending rate before the connection experiences losses

a. New retransmission mechanism

TCP Vegas introduces three changes that affect TCP's (fast) retransmission strategy. First, TCP Vegas measures the RTT for every segment sent. The measurements are based on fine- grained clock values. Using the fine-grained RTT measurements, a timeout period for each segment is computed. When a duplicate acknowledgement (ACK) is received, TCP Vegas checks whether the timeout period has expired. If so, the segment is retransmitted1. Second, when a non-duplicate ACK that is the first or second after a fast retransmission is received, TCP Vegas again checks for the expiration of the timer and may retransmit another segment. Third, in case of multiple segment loss and more than one fast retransmission, the congestion window is reduced only for the first fast retransmission.

b. Congestion avoidance mechanism TCP Vegas does not continually increase the congestion window during congestion avoidance. Instead, it tries to detect incipient congestion by comparing the measured throughput to its notion of expected throughput. The congestion window is increased only if these two values are close, that is, if there is enough network capacity so that the throughput can actually be achieved. The congestion window is reduced if the measured.

c. Modified slow-start mechanism

A similar congestion detection mechanism is applied during slow-start to decide when to change to the congestion avoidance phase. To have valid comparisons of the expected and the actual throughput, the congestion window is allowed to grow only every other RTT Algorithms used to modify TCP Vegas are:

☐Congestion detection during slow-start

☐Congestion detection during congestion avoidance

☐More aggressive fast re-transmit mechanism

☐Additional retransmissions for non-duplicate ACKs

☐Prevention of multiple reductions of the congestion window in case of multiple segment loss

☐Reduction of the congestion window by only 1/4 after a recovery (instead of halving it as in the case of TCP Reno).

☐A congestion window size of two segments at initialization and after a timeout (TCP Reno sets the size of the congestion window to one segment in these situations)

☐Burst avoidance limits the number of segments that can be sent at once (that is, back- to-back) to three segments

☐The congestion window is not increased if the sender is not able to keep up, that is, the difference between the size of the congestion window and the amount of outstanding data is larger than two maximum-sized segments

☐Spike suppression limits the output rate to at most twice the current rate. (This algorithm is turned off by default.) Key points of TCP Vegas:

☐Modified Congestion Avoidance

☐Aggressive Retransmission (use fine grained timer)

☐With dupacks and with partial acks

☐Aggressive Congestion Window Adaptation

☐With recovery and with multiple loss

☐Modified Slow-Start

d. Modified Congestion Avoidance for TCP Vegas

TCP Vegas Calculates the expected throughput and actual throughput (Once per RTT):

Expected Throughput = WindowSize/BaseRTT

Actual Throughput = ActualSentAmount/RTT

Static Parameters:

☐ = 1 pkts/RTT

☐ = 3 pkts/RTT

Where,☐ *and*☐ are static variables for TCP Vegas

TCP transmission rate = cwnd/RTT [28] where, cwnd-congestion window and RTT - Retransmission timeout.

TCP takes congestion window updating decision once per RTT; the decision is applied throughput the next RTT for each received ACK as follows:

☐Increase the Tx Rate (Expected-Actual>☐)

• cwnd = cwnd + 1/cwnd

☐Decrease Tx Rate (Expected-Actual<☐)

• cwnd = cwnd - 1/cwnd

☐Tx Rate Unchanged (☐<Expected-Actual<☐)

• cwnd = cwnd

e. Aggressive Retransmission for TCP Vegas

☐With dupacks

• When Vegas receives the first dupack or the second dupacks, it checks the fine grained timer expiry

• If timer expirers, it retransmits immediately

☐With partial acks

• For the first two partial acks, Vegas checks whether fine grained timer expires

• If timer expires, it retransmits immediately

f. Aggressive cwnd updating for TCP Vegas

☐With recovery

• Reduce cwnd by one quarter instead of half when it enters into recovery

☐With multiple loss-

• In case of multiple segment loss from a single window, it reduces the cwnd only once

☐With Initial setting

•cwnd is set to 2 instead of 1

g. Modified Slow-Start for TCP Vegas

•Vegas Calculates (in every alternate RTT)

•*Expected Throughput* = WindowSize/BaseRTT

•*Actual Throughput* = ActualSentAmount/RTT

•Static Parameters:☐ = 1 pkts/RTT

Where☐ is static variable for TCP Vegas

•TCP keeps the congestion window fixed in every other RTT and it measures the throughput

•On every next RTT, it does the followings:

•Continue SS (Expected-Actual<☐)

•Exponential Increase.

- Cwnd = cwnd + 1 for each ACK, that is,
- Cwnd = 2 * cwnd for each RTT

• Switch to CA (Expected-Actual>☐):

- Set ssthresh = cwnd
- Follow the rules of CA

h. Condition of Increment in CA for TCP Vegas

$$(ExpectedThroughput - ActualThroughput) > \frac{\beta}{baseRTT}$$

$$\left(\frac{WindowSize}{baseRTT} - \frac{WindowSize}{RTT}\right) * baseRTT > \beta$$

$$WindowSize * \left(1 - \frac{baseRTT}{RTT}\right) > \beta$$

$$WindowSize > \frac{\beta}{\left(1 - \frac{baseRTT}{RTT}\right)}$$

i. Problems of CA for TCP Vegas

- Unfair Treatment of Old Connections
  - Old connections have smaller baseRTT
  - Vegas decreases Tx Rate if

$$WindowSize > \frac{\beta}{1 - \frac{baseRTT}{RTT}}$$

  - Vegas increases Tx Rate if

$$WindowSize < \frac{\alpha}{1 - \frac{baseRTT}{RTT}}$$

☐ Persistent Congestion

Where, BaseRTT is RTT of segment when congestion is not congested, Window Size is the number of segments in transit corresponds to the number of segments sent during the last RTT.

j. Vegas conclusion

☐Less fluctuation

- Less fluctuation in bottleneck queue and in send rate

☐Enhanced Throughput

☐Better Utilization of bottleneck capacity

☐Unfair treatment of old connection and Ineffectiveness of congestion avoidance

III. TCP Vegas - Without Modification

a. Result for Node 25

Mobility(m/s)	Throughput (kbps)	End to End Delay (ms)	PDR (%)	Packet Drop	Protocol
10	261.21	24.7533	99.939	4	AODV
20	90.09	107.79	98.9125	22	AODV
50	135.53	54.1261	99.6496	14	AODV
10	269.09	22.8216	81.3158	3	OLSR
20	0.18	41.058	0.2296	7	OLSR
50	0.18	32.3741	0.1858	7	OLSR

Table-1

Table 1 shows the output of performance parameters for TCP Vegas for ad hoc routing protocol AODV and OLSR;

we got the result for node 25 and node mobility 10, 20 and 50 m/s respectively without any modification in TCP Vegas.

b. **Results for Node 50**

Mobility(m/s)	Throughput (kbps)	End to End Delay (ms)	PDR (%)	Packet Drop	Protocol
10	118.96	54.227	99.2752	24	AODV
20	39.94	122.502	95.898	48	AODV
50	137.80	56.6602	99.2591	24	AODV
10	81.44	43.7703	25.7152	12	OLSR
20	0.9	46.6817	0.0393	5	OLSR
50	60.85	36.6385	20.0723	22	OLSR

Table-2

Table2 shows the output of performance parameters for TCP Vegas for ad hoc routing protocol AODV and OLSR; we got the result for node 50 and node mobility 10, 20 and 50 m/s respectively without any modification in TCP Vegas.

IV. TCP Vegas - With Modification

Modification for TCP Vegas:
According to characteristic of TCP Vegas, it gives the minimum packet loss for MANETs but, also reduce the throughput. We made change in slow-start phase logically and vary the congestion window, because congestion window will affect the performance of TCP Vegas.
We minimize the packet loss but also increase the throughput and also get changes in other performance parameters, endto- end delay and PDR.
For congestion avoidance phase, change in congestion window will help to minimize the congestion and due to less congestion, packet loss occur minimum and it will also help in to increase throughput and PDR as well.

a. Result for Node 25

Mobility(m/s)	Throughput (kbps)	End to End Delay (ms)	PDR (%)	Packet Drop	Protocol
10	261.66	24.8611	99.9565	1	AODV
20	91.84	67.1777	98.9125	10	AODV
50	136.16	40.2792	99.6496	15	AODV
10	269.34	20.8080	81.0607	2	OLSR
20	0.18	41.0580	0.2296	7	OLSR
50	0.18	41.058	0.2296	7	OLSR

Table-3

Table 3 shows the output of performance parameters for TCP Vegas for ad hoc routing protocol AODV and OLSR; we get the result for node 25 and node mobility 10, 20 and 50 m/s respectively with some modification in TCP Vegas.

b. Result for Node 50

Mobility(m/s)	Throughput (kbps)	End to End Delay (ms)	PDR (%)	Packet Drop	Protocol
10	123.66	44.7896	99.5959	12	AODV
20	49.22	204.716	96.2829	45	AODV
50	136.69	38.9661	99.5515	19	AODV
10	94.56	35.2648	29.2569	10	OLSR
20	0.9	46.6817	0.0393	5	OLSR
50	78.87	39.7158	22.0315	18	OLSR

Table-4

Table 4 shows the output of performance parameters for TCP Vegas for ad hoc routing protocol AODV and OLSR we got the result for node 50 and node mobility 10, 20 and 50 m/s respectively without any modification in TCP Vegas.

CONCLUSION

We have compared two ad hoc routing protocols, namely, Ad hoc On-Demand Distance Vector Routing (AODV) and Optimized Link State Routing (OLSR). The simulation of these protocols has been carried out using Ns-2.
Three different simulation scenarios are generated; the node mobility has varied from 10 m/sec, 20 m/sec and 50 /sec. for 25 nodes and 50 nodes for TCP Vegas. Other network parameters are kept constant during the simulation.
It is observed that using TCP Vegas and performance parameters like throughput, end-to-end delay, PDR (Packet Delivery Ratio) and packet loss, AODV performs well in node mobility 20 m/s. but, OLSR is not suitable for node density 25 and OLSR performs well in node mobility 50 m/s. but, AODV is not suitable for node density 50.
It would be interesting to observe the behavior of these two protocols by varying others network parameters.
It would be modify and implement other TCP Variants and Try to implement new TCP variant which will provide balance result among other TCP variants , which will help to improve the performance of AODV, OLSR and others ad hoc routing protocols for MANETs.

REFERENCES

[1]Vivek Kumar Me in Computer Science and Engineering Mr. Sumit Miglani (Lecturer) *Simulation and Comparison Of Aodv And Dsr Routing Protocols In Manets* Computer Science And Engineering Department Thapar University Patiala 147004 July 2009

[2] L. S. Brakmo and L. Peterson. *TCP Vegas: New Techniques for Congestion Detection and Avoidance*. In Proc. of ACM SIG-COMM '94, pages 24-35, London, October 1994.

[3]L.S. Brakmo and L.L. Peterson. *TCP Vegas: End to End Congestion Avoidance on a global Internet*. IEEE Journal on Selected Areas in Communications, 13(8):1465-1480, Oct 1995.

[4] U. Hengartner1, J. Bolliger1 and Th. Gross1 2 *TCP Vegas Revisited* 1Departement Informatics ET H Z¨urich CH 8092, 2Z¨urich School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213