ISSN No. 0976-5697

Volume 6, No. 1, Jan-Feb 2015

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Developing an Efficient Mechanism of Quality of Service on WINS Node

[1]Vishal Singh, [2]Ashish Kumar
[1](Research Scholar),[2](Asst. Professor)
Dept. Computer Science & Engineering
[1]PSIT, Kanpur Nagar India
[2]PSITCOE, Kanpur Nagar India

Abstract: On wireless computer networks, ad-hoc mode is a method for wireless devices to directly communicate with each other. To set up an ad-hoc wireless network, each wireless adapter must be configured for ad-hoc mode versus the alternative infrastructure mode. An ad-hoc network tends to feature a small group of devices all in very close proximity to each other. Performance suffers as the number of devices grows, and a large ad-hoc network quickly becomes difficult to manage. Ad-hoc networks cannot bridge to wired LANS or to the Internet without installing a special-purpose gateway. Apply some modifications on parameters of existing protocols as well as analysis on these parameters by which can improve Quality of Service.

Keywords: WINS, AODV, IP Header, DSCP, ECN, Trace graph, NS2, OMNET++

I. INTRODUCTION

Wireless integrated network sensors (WINS) provide distributed network and Internet access to sensors, controls and processors that are deeply embedded in equipment, facilities and environment [1].

Ad-hoc On-Demand Distance Vector (AODV) Routing is a routing protocol for mobile ad-hoc networks and other wireless ad-hoc network. In AODV, the network is silent until a connection is needed. It uses an on-demand approach for finding routes, in which a route is, established only when it is required by a source node for transmitting data packets. AODV uses symmetric links between neighboring nodes. It does not follow paths between nodes when one of the nodes cannot hear the other one however we may include the use of such links in future enhancements [2].

Differentiated services or DiffServ is a computer networking architecture that specifies a simple, scalable and coarse-grained mechanism for classifying and managing network traffic and providing quality of service (QoS) on modern IP networks. DiffServ uses a 6-bit differentiated services code point (DSCP) in the 8-bit Differentiated Services Field (DS Field) in the IP header for packet classification purposes [3]. The DS field and ECN field replace the outdated IPv4 TOS field. DiffServ-aware routers implement per-hop-behaviors (PHBs), which define the packet-forwarding properties associated with a class of traffic. In practice, most networks use the following commonly defined Per-Hop Behaviors:

a. Default PHB (Per hop behavior)--which is typically best-effort traffic.
b. Expedited Forwarding (EF) PHB—dedicated to low-loss, low-latency traffic.
c. Assured Forwarding (AF) PHB—gives assurance of delivery under prescribed conditions.
d. Class Selector PHBs—which maintain backward compatibility with the IP Precedence field [3].

Trace graph is a great application that comes handy to ns2 users. It eliminates the need to configure and run Perl / awk scripts over the trace file. Trace file analysis simplified.

Trace graph is third party software helps in plotting the graphs for ns2 and other networking simulation software [4].
In 1996-97, ns version 2 (ns-2) was initiated based on a refactoring by Steve McCanne. NS-2 is a discrete event simulator targeted at networking research. NS-2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks [5].
In this paper, we have describe network simulation tools that helps to define results easily and research scholar focus on their research, introduces more efficient mechanism of Quality of Service. There are many parameters available in protocol header format but some parameters are unused which are no need to send every time in protocol header format. Here we have done operations on unused parameters of protocols.

II. WINS

Wireless Integrated Network Sensors (WINS) form a new distributed information technology of combination of sensor, and processing systems. WINS nodes are autonomous, self-organized, wireless sensing and control networks. WINS nodes include micro sensors, signal processing, computation and low power wireless networking. WINS enable distributed measurements for applications ranging from aerospace system condition monitoring to distribute environmental science monitoring. WINS require a Microwatt of power. But it is very cheaper when compared to other security systems such as RADAR. It is used for short distance. It produces a less amount of delay. Hence it is reasonable faster. On a global scale, WINS will allow monitoring of land, water, and air resources for environmental monitoring [1].

The opportunities for WINS depend on the development of low cost, sensor network architecture. This requires sensor information be conveyed to the user at low bit rate with low power transceivers. Continuous sensor signal processing must be provided to enable constant monitoring of events in an environment. In contrast to conventional wireless networks, the WINS network must support large numbers of

sensors in a local area with short range and low average bit rate communication [1].

III. NS2 SIMULATOR

NS (version 2) is an object-oriented, discrete event driven network simulator developed at UC Berkely written in C++ and OTcl (Object- oriented Tcl script language). It implements many network protocols such as FTP, Telnet, Web, CBR, and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing. Network Simulator also implements multicasting and some of the MAC layer protocols for LAN simulations. To simulate any network scenario, NS-2 users need to write a simulation script, and invoke the NS2 interpreter. NS2 will simply store the results in form of trace files. One popular approach is to produce two types of trace file, i.e. network animation (NAM) trace file and normal trace file. NAM trace file is used for network animation purposes. While for trace file processing, a program can be coded using the user's own favorite software and graph plotters like GNU plot and xgraph can be used to view the results [4].

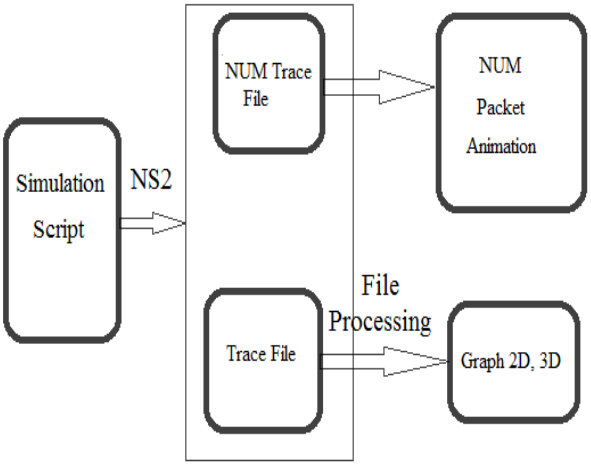

Figure 1: NS2 Simulation Process Flow

IV. TRACEGRAPH

Trace Graph provides a simple visual display of the program's trace which allows changes in execution to be easily distinguished. Trace Graph is a free network trace files analyzer developed for network simulator ns-2 trace processing and Trace Graph can support any trace format if converted to its own or ns-2 trace format [4]. Trace graph when opened, it opens three windows: (a) First window to select the trace file (.tr) that was created by NS2 (depending on the size of the trace file, the processing time also varies). (b) Second window is the main window in which you can see the graphs for various performance characteristics, like throughput, End to End Delay, and Jitter, etc in 2D and 3D facility. Even it can plot the histograms. (c) Third window is nothing but Simulation Information Windows, that you can see the packet loss, packet delivery, end to end delay, Simulation processing times, Simulation Round Trip Times. Trace graph runs under Windows, Linux, UNIX, MAC OS systems. NS2 trace file formats: wired, satellite, wireless (old and new trace), new trace and wired-wireless [4].

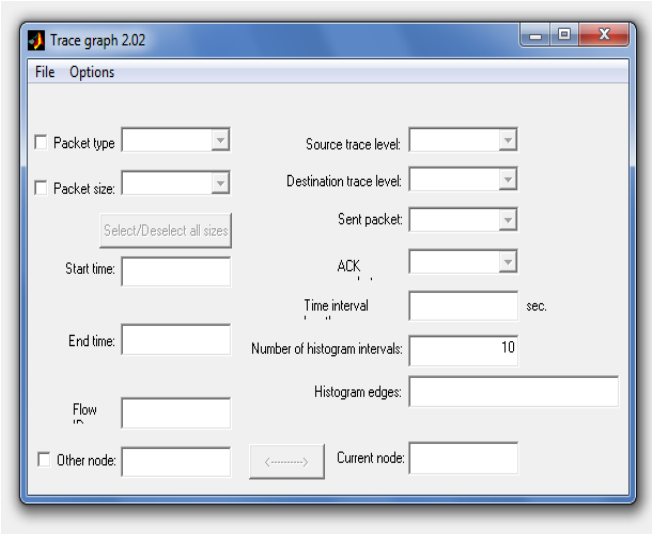

Figure 5: First Window

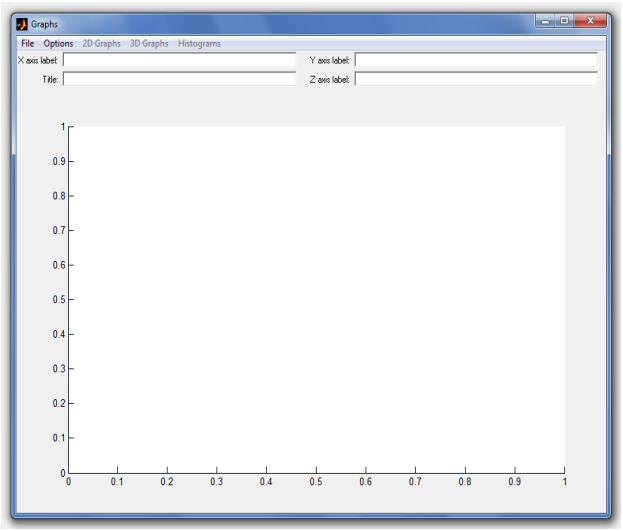

Figure 6: Second Window

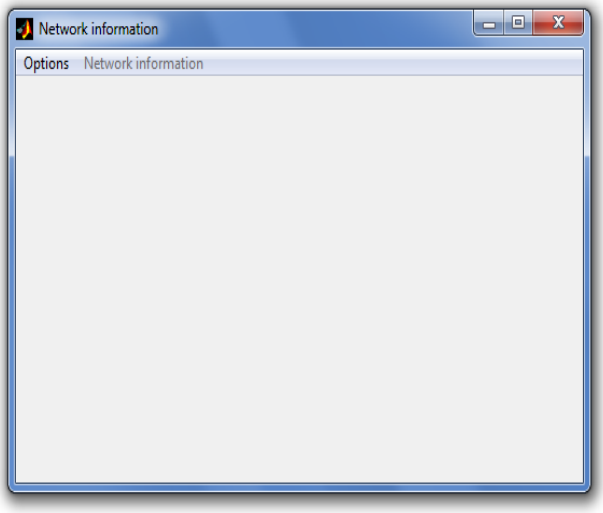

Figure 7: Third Window

V. OMNET++

OMNET++ is an extensible, component-based C++ simulation library and framework, primarily for building network simulators. Domain-specific functionality such as

support for sensor networks, wireless ad-hoc networks, Internet protocols, performance evaluation, etc. There are extensions for real-time simulation, network emulation, alternative programming languages (Java, C#), database integration, SystemC integration, and several other functions. OMNET++ is free for academic, and it is a widely used platform in the global scientific community. Commercial users must get a license from omnest.com. OMNET++ runs on Windows, Linux, Mac OS X, and other Unix-like systems. The OMNET++ IDE runs on Windows, Linux, or Mac OS X [6].

INET Framework is an open-source communication network simulation package for the OMNET++ simulation environment. The INET framework contains models for several wired and wireless networking protocols, and many others [6].

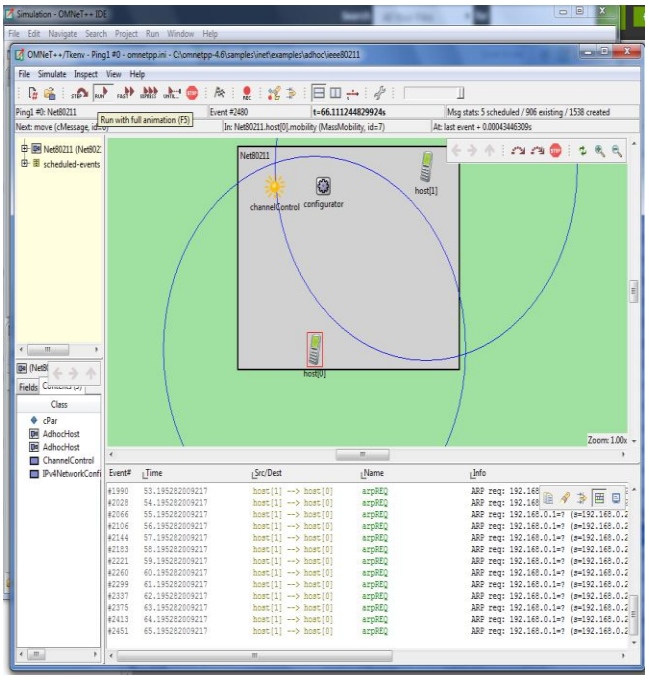

Figure 2: Ad-hoc Simulation in OMNET++

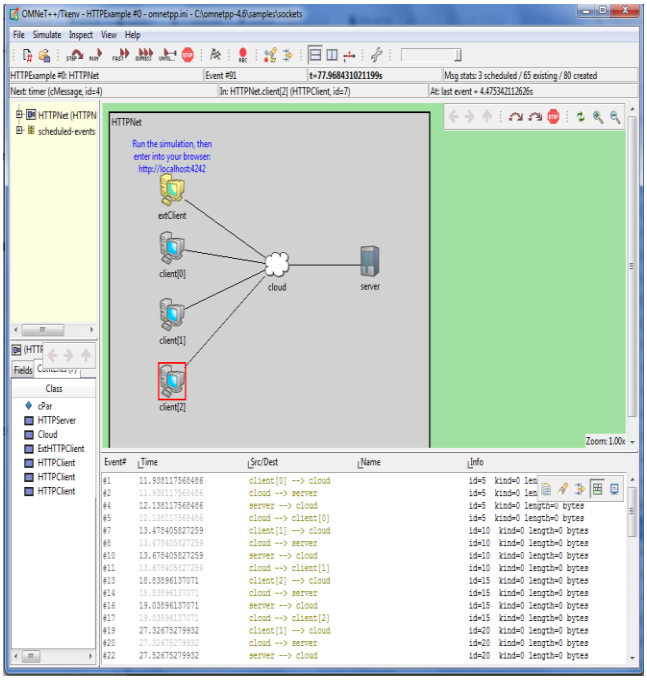

Figure 3: Sockets Simulation in OMNET++

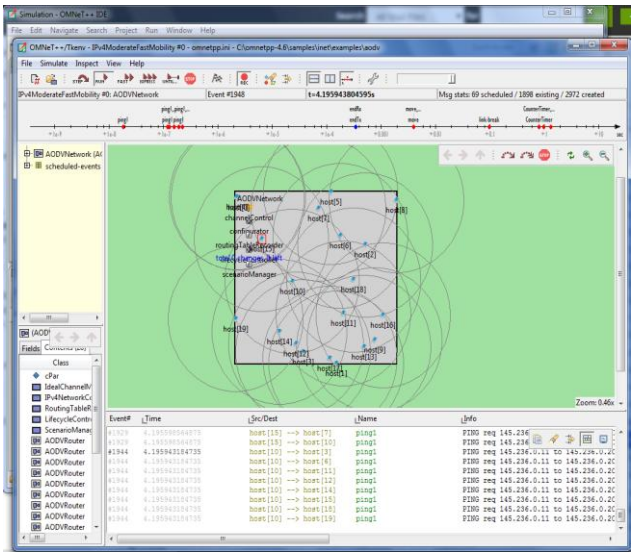

Figure 4: AODV Simulation in OMNET++

VI. AODV ROUTING PROTOCOL

Ad hoc on demand distance vector protocol is reactive routing protocol. It constructs route on demand and aims to reduce routing load [7]. AODV deals with route table management. Route table information must be kept for short-lived routes, such as are created to temporarily store reverse paths towards nodes originating RREQs. AODV the following fields with each route table entry [7]:

a. Destination IP Address
b. Destination Sequence Number
c. Valid Destination Sequence Number flag
d. Other state and routing flags (e.g., valid, invalid, repairable, being repaired)
e. Network Interface
f. Hop Count (number of hops needed to reach destination)
g. Next Hop
h. List of Precursors
i. Lifetime (expiration or deletion time of the route)

The AODV routing protocol is designed for mobile ad hoc networks with populations of tens to thousands of mobile nodes. AODV can handle low, moderate, and relatively high mobility rates, as well as a variety of data traffic levels. AODV is designed for use in networks where the nodes can all trust each other, either by use of preconfigured keys, or because it is known that there are no malicious intruder nodes. AODV has been designed to reduce the dissemination of control traffic and eliminate overhead on data traffic, in order to improve scalability and performance [7].

A. Route Request (RREQ) Message Format:

Table: 1

0	8	13	24	32
Type	J|R|G|D|U	Reserved	Hop Count	
RREQ ID				
Destination IP Address				
Destination Sequence Number				
Originator ID Address				
Originator Sequence Number				

The format of the Route Request message is illustrated above, and contains the following fields:

Type: 1

J: Join flag; reserved for multicast.

R: Repair flag; reserved for multicast.

G: Gratuitous RREP flag; indicates whether a Gratuitous RREP should be unicast to the node specified in the Destination IP Address field

D: Destination only flag; indicates only the Destination may respond to this RREQ

U: Unknown sequence number; indicates the destination sequence number is unknown.

- ***a. Reserved:*** Sent as 0; ignored on reception.
- ***b. Hop Count:*** The number of hops from the Originator IP Address to the node handling the request.
- ***c. RREQ ID:*** A sequence number uniquely identifying the particular RREQ when taken in conjunction with the originating node's IP address.
- ***d. Destination IP Address:*** The IP address of the destination for which a route is desired.
- ***e. Destination Sequence Number:*** The latest sequence number received in the past by the originator for any route towards the destination.
- ***f. Originator IP Address:*** The IP address of the node which originated the Route Request.
- ***g. Originator Sequence Number:*** The current sequence number to be used in the route entry pointing towards the originator of the route request [7] .

B. Route Reply (RREP) Message Format:

Table:2

Type	**R| A| Reserved**	**Prefix Sz**	**Hop Count**
Destination IP address			
Destination Sequence Number			
Destination Sequence Number			
Originator ID Address			
Lifetime			

The format of the Route Reply message is illustrated above, and contains the following fields:

Type: 2

R: Repair flag; used for multicast.

A: Acknowledgment required;

- ***a. Reserved:*** Sent as 0; ignored on reception.
- ***b. Prefix Size:*** If nonzero, the 5-bit Prefix Size specifies that the indicated next hop may be used for any nodes with the same routing prefix (as defined by the Prefix Size) as the requested destination [7].
- ***c. Hop Count:*** The number of hops from the Originator IP Address to the Destination IP Address. For multicast route requests this indicates the number of hops to the multicast tree member sending the RREP.
- ***d. Destination IP Address:*** The IP address of the destination for which a route is supplied.
- ***e. Destination Sequence Number:*** The destination sequence number associated to the route.
- ***f. Originator IP Address:*** The IP address of the node which originated the RREQ for which the route is supplied.
- ***g. Lifetime:*** The time in milliseconds for which nodes receiving the RREP consider the route to be valid [7].

VII. INTERNET HEADER FORMAT

Table:3

0 3 7 15 31

Version	**IHL**	**Differentiated Services**	**Total Length**	
Identification			Flags	Fragment Offset
TTL		Protocol	Header Checksum	
Source IP Address				
Destination IP Address				

Version: 4 bits, Bit 0 - Reserved [8].

The Version field indicates the format of the internet header. This document field indicates the format of the internet header.

- ***a. IHL:*** 4 bits, Internet Header Length is the length of the internet header in 32 bit words, and thus points to the beginning of the data. The minimum value for a correct header is 5.
- ***b. Type of Service:*** 8 bits, The Type of Service provides an indication of the abstract parameters of the quality of service desired. These parameters are to be used to guide the selection of the actual service parameters when transmitting a datagram through a particular network [8].

Bits 0-2: Precedence.

Bit 3: 0 = Normal Delay, 1 = Low Delay.

Bit 4: 0 = Normal Throughput, 1 = High Throughput.

Bit 5: 0 = Normal Reliability, 1 = High Reliability.

Bits 6-7: Reserved for Future Use.

0 - 2	3	4	5	6	7
Precedence	D	T	R	0	0

Precedence -

111 – Network Control

110 – Internetwork Control

101 – CRITIC/ECP

100 – Flash Override

011 – Flash

010 – Immediate

001 – Priority

000 – Routine

The use of the Delay, Throughput, and Reliability indications may increase the cost of the service.

Total Length: 16 bits, Total Length is the length of the datagram, measured in octets, including internet header and data.

Flags: 3 bits, Various Control Flags.

Bit 0: Reserved, must be zero

Bit 1: (DF) 0 = May Fragment, 1 = Don't Fragment.

Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments.

- ***a. Fragment Offset:*** 13 bits, the fragment Offset is measured in units of 8 octets (64 bits) [8].
- ***b. Time to Live:*** 8 bits, this field indicates the maximum time the datagram is allowed to remain in the internet system.
- ***c. Protocol:*** 8 bits, this field indicates the next level protocol used in the data portion of the internet datagram.
- ***d. Header Checksum:*** 16 bits, this is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header.

Source Address: 32 bits
Destination Address: 32 bits [8].

VIII. EXPERIMENTAL ANALISYS

A. Experiment on AODV Protocol :

When we talk about parameters of AODV Routing Protocol, There are various parameters available and every parameter has its specific information. But some parameters are either not required to be sent every time or their value need to be changed. First one is packet type, no need to send at request of aodv structure as well as reply of aodv structure. Here we have remove packet type from structure of aodv request and aodv reply [7]. As well as one more parameter of AODV is 8 bits reserved field that is sent as 0 and ignored at reception [7], here we have also removed it. By which we may improve the QoS of routing protocol and found more efficient routing protocol. These changes will reduce values, sizes etc. of following terms:

a. Simulation Length
b. Number of generated packets
c. Number of sent packets
d. Average packet size
e. Average packet size (current node)
f. Number of sent bytes
g. Maximal delay in simulation End2End delays
h. Average in simulation End2End delays
i. Minimal(node, PID) in simulation processing times at intermediate nodes
j. Maximal(node, PID) in simulation processing times at intermediate nodes
k. Average in simulation processing times at intermediate nodes

B. Experimental Results:

In this AODV routing protocol has set in which we have create network with 23 nodes and 2 servers. For execution of this script, we have used NS2 simulator on Linux os and simulate it by which creates NAM file and Trace file. NAM File is used for represents animation of network simulation and Trace file is used for simulation results in which we have used Trace Graph for results evaluation [4].

a) Network Simulation Information of Original AODV Protocol:

Simulation information:

Simulation length in	6.080717962
Number of	23
Number of sending nodes:	23
Number of receiving nodes:	23
Number of generated	2708
Number of sent packets:	2627
Number of lost packets:	0
Minimal packet size:	28
Maximal packet size:	1078
Average packet size:	156.7087
Number of sent bytes:	626488
Packets dropping nodes:	0 1 2 3 4 5 6 7 8

Figure 8: Simulation Information of Original AODV

b) Network Simulation Information of Changed AODV Protocol:

Simulation information:

Simulation length in	5.54142059
Number of	23
Number of sending nodes:	23
Number of receiving nodes:	23
Number of generated	2622
Number of sent packets:	2493
Number of lost packets:	0
Minimal packet size:	28
Maximal packet size:	1078
Average packet size:	140.2755
Number of sent bytes:	527130
Packets dropping nodes:	0 1 2 3 4 5 6 7 8

Figure 9: Simulation Information of Efficient AODV

c) Network Simulation Current node Information of Original AODV Protocol:

Current node information:

Number of generated	926
Number of sent	926
Number of forwarded	0
Number of received	645
Number of lost	0
Number of sent	493580
Number of forwarded	0
Number of received	25832
Minimal packet size:	28
Maximal packet size:	1078
Average packet size:	330.6251

Figure 10: Current Node Information of Original AODV

d) Network Simulation Current node Information of Changed AODV Protocol:

Current node information:

Number of generated	784
Number of sent	780
Number of forwarded	0
Number of received	596
Number of lost	0
Number of sent	413058
Number of forwarded	0
Number of received	24240
Minimal packet size:	28
Maximal packet size:	1078
Average packet size:	317.8038

Figure 11: Current Node Information of Efficient AODV

e) Simulation Processing Times at Intermediate Nodes of Original AODV:

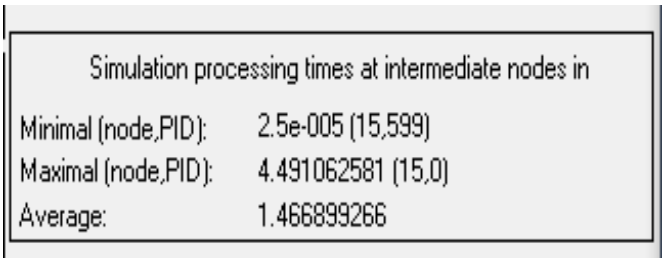

Figure 12: Simulation Processing Times at Intermediate Nodes of Original AODV

f) Result of Simulation Processing Times at Intermediate Nodes of Changed AODV:

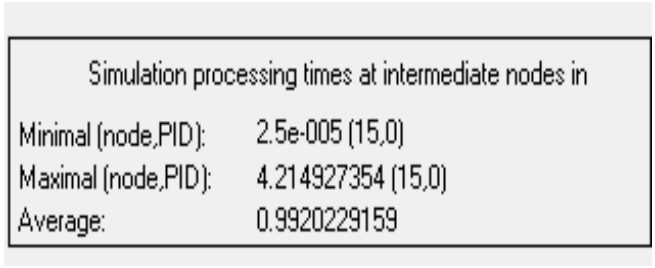

Figure 13: Simulation Processing Times at Intermediate Nodes of Efficient AODV

g) Result of Simulation End2End delays of Original AODV protocol:

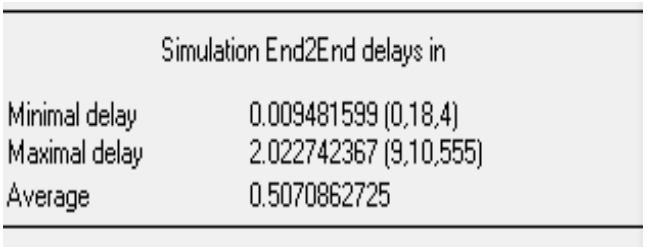

Figure 14: Simulation End2End Delays of Original AODV

h) Result of Simulation End2End delays of Changed AODV protocol:

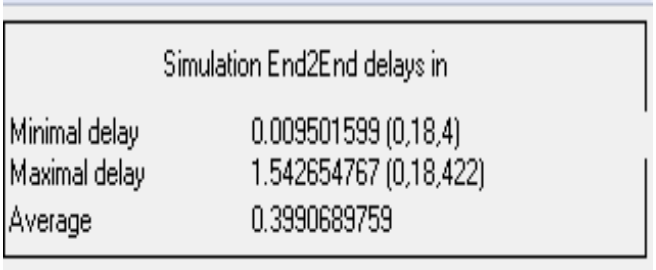

Figure 15: Simulation End2End Delays of Efficient AODV

a. Experimental Results with 2D Graphs:

In this section, describes 2D results for many terms such as:

(a). Packet Size vs. Average Simulation End2End delay.
(b). Cumulative Sum of Numbers of all the Received Packets.
(c). Throughput of Receiving Bits vs. Average Simulation Processing Time.

a) Result of Packet Size vs Average Simulation End2End delay of Original AODV:

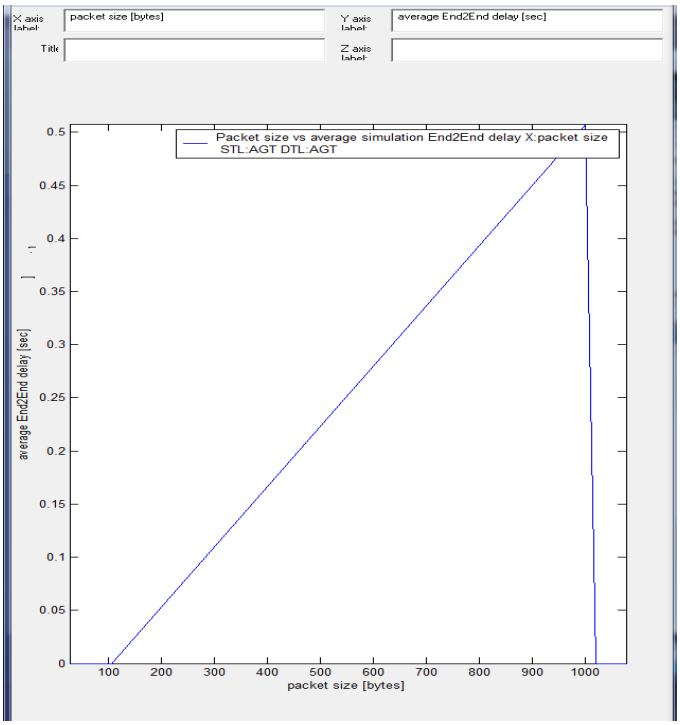

Figure 16: Packet Size vs Average Simulation End2End delay of Original AODV

b) Result of Packet Size vs Average Simulation End2End delay of Changed AODV:

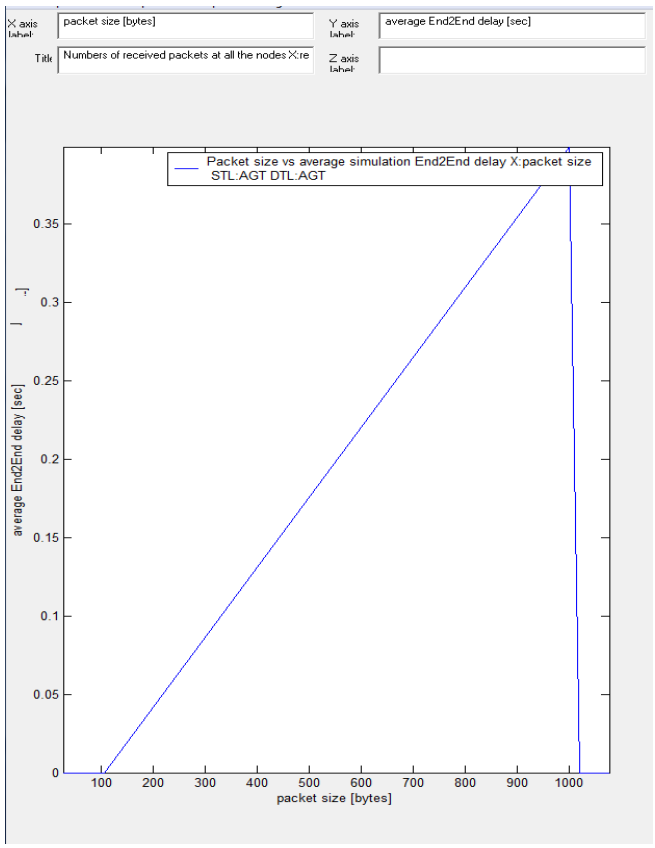

Figure 17: Packet Size vs Average Simulation End2End delay of Efficient AODV

c) Result of Cumulative Sum of Numbers of all the Received Packets of Original AODV:

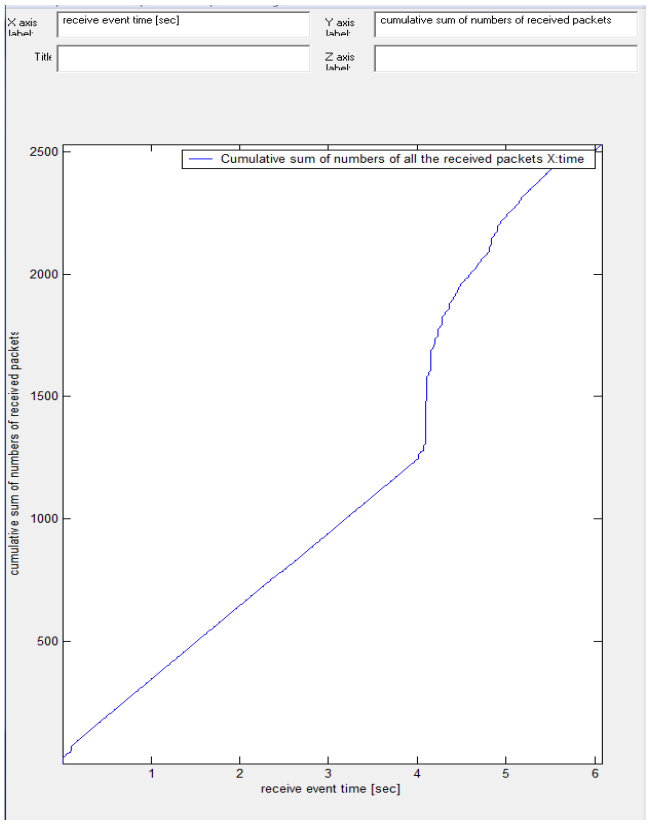

Figure 18: Cumulative Sum of Numbers of all the Received Packets of Original AODV

d) Result of Cumulative Sum of Numbers of all the Received Packets of Changed AODV:

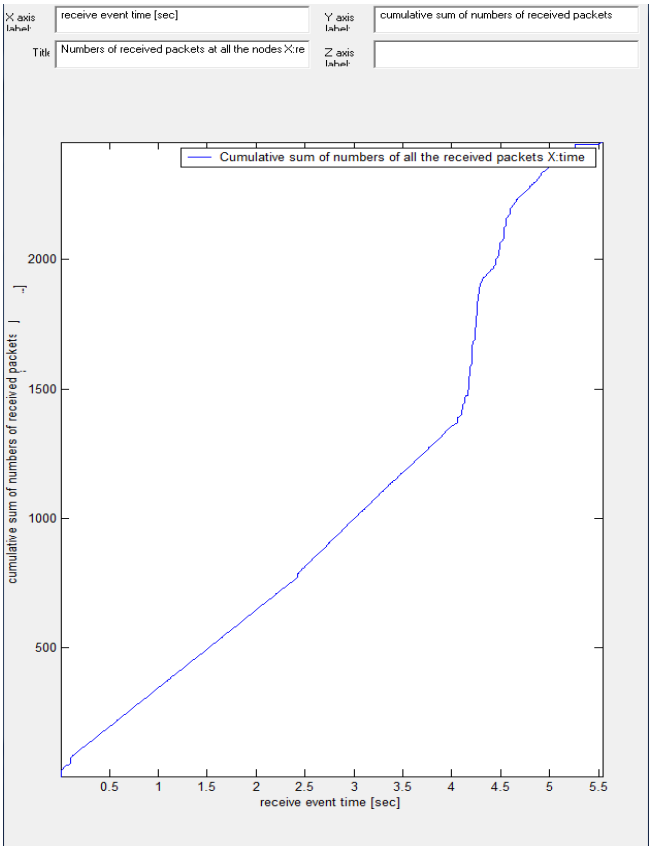

Figure 19: Cumulative Sum of Numbers of all the Received Packets of Efficient AODV

e) Result of Throughput Of Receiving Bits vs Average Simulation Processing Time of Original AODV:

This graph represents the throughput of AODV routing protocol between receiving bits and average simulation processing time.

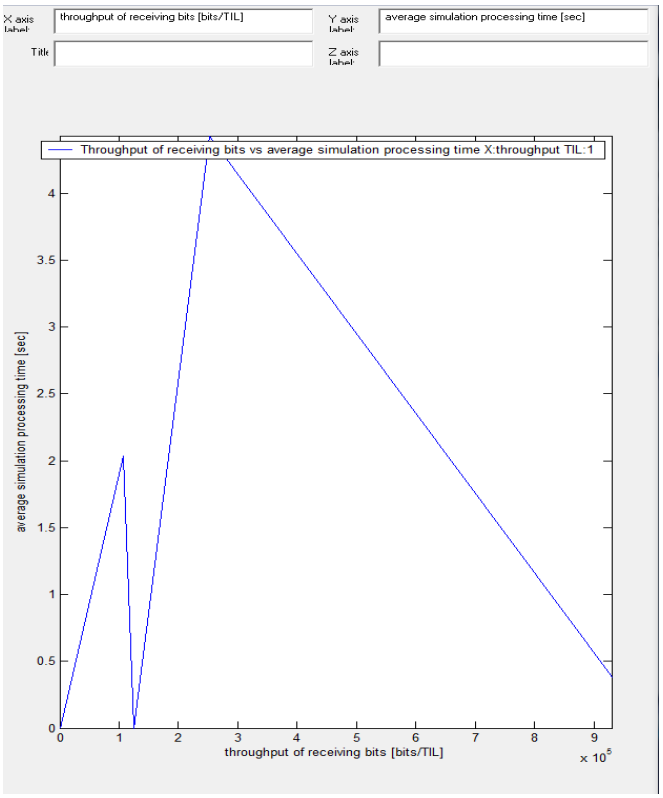

Figure 20: Throughput Of Receiving Bits vs Average Simulation Processing Time of Original AODV

f) Result of Throughput Of Receiving Bits vs Average Simulation Processing Time of Changed AODV:

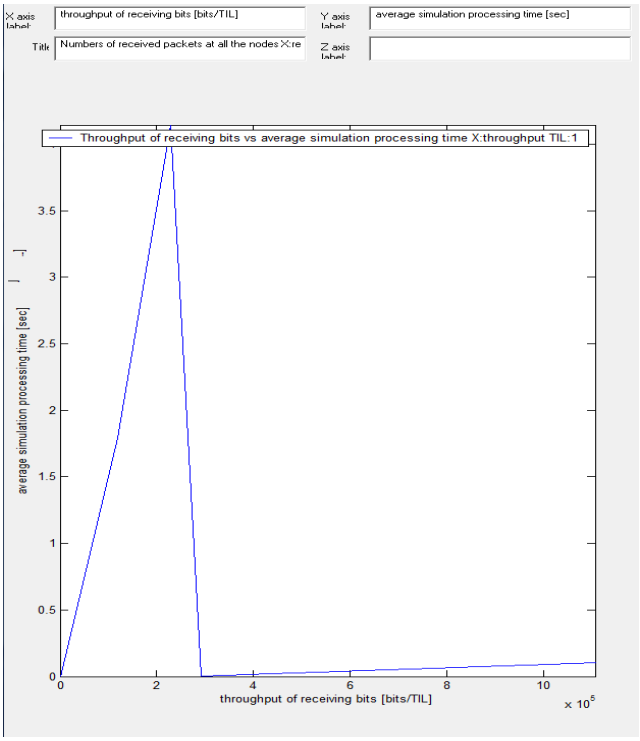

Figure 21: Throughput of Receiving Bits vs Average Simulation Processing Time of Efficient AODV

C. Experiment on IPv4 Header:

Currently, 8 bit data is passed in Differentiated Field (Type of Service) But Only 6 bit data is effectively used while remaining 2 bits are unutilized [8]. Further, neither there is any suggestion how to use it nor is it anywhere elaborated as why these 2 bits are preserved in every packet. And Version field of IP Header contains 4 bits but Bit 0 is reserved and Bit 1, 2, 3 defines IP version [8]. As well as Flags field of IP Header also contains 3 bits but first Bit is reserved [8]. The reserved bits of Version and Flags Fields no need to send with every packet. Here we have removed these reserved bits from IP Header fields, this will reduce minimum length 160 bits of IPv4 header and reduced length is 156 bits in each packet.

D. Experimental Results:

a) Result of Original IP Header fields of IPv4 header:

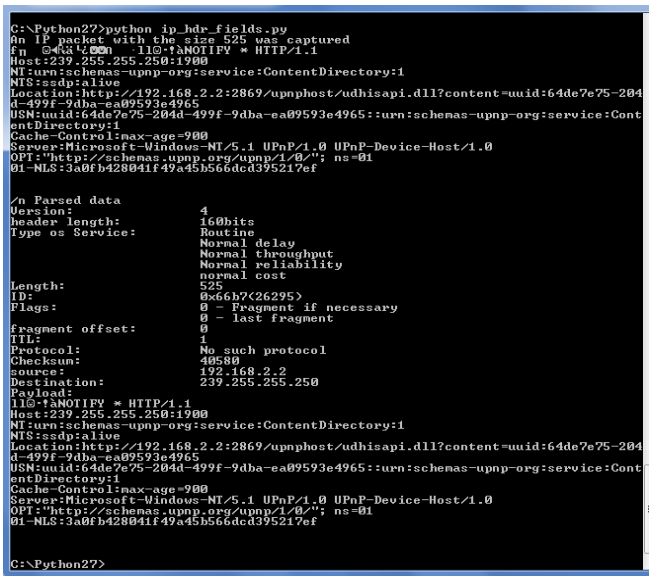

Figure 22: IP Header Fields of Original IPv4 Header

b) Result of Changed IP Header fields of IPv4 header:

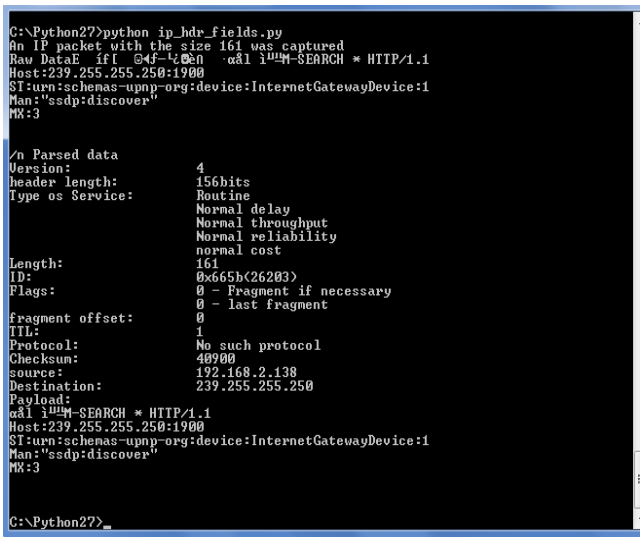

Figure 23: IP Header Fields of Efficient IPv4 Header

IX. CONCLUSION AND FUTURE WORK

In this paper, we presented some changes on existing protocols by which we may improve the quality of service on WINS node and develop more efficient protocol.

We summarize our major results presented in below:

a. We have removed the packet type from header request and reply structure of AODV routing protocol and also removed reserved field from header reply structure of AODV routing protocol. By which we have found more efficient routing protocol.

b. We have removed reserved 2 bits ECN field from differentiated service field (Type of service), removed reserved 1 bit from version field and removed reserved 1 bit from Flags field of IPv4 header format. By which we have improved Quality of Service of IPv4 protocol.

For Future work, we may remove or change more parameters in other protocols.

X. ACKNOWLEDGMENT

This research paper is made possible through the help and support from everyone. Especially, please allow me to dedicate my acknowledgment toward the following significant advisors and contributors:

First and foremost, I would like to thank Mr. Ashish Kumar (Asst. Prof.) for his most support and encouragement. He kindly read my paper and offered invaluable detailed advices on grammar, organization, and theme of the paper.

Second, I would like to thank HoD of dept. of C.S.E., M.Tech and Dean of the institute.

XI. REFERENCES

[1] Proposed work: Secure & Efficient Mechanism of Quality of Service on WINS Node, Vishal Singh, Rajendra Singh, Saurabh Sachan; NCISCL, 5-6 April-2014 in HBTI Kanpur, pp: (127-130).

[2] A Survey: QoS of MANET through cryptography and routing protocol enhancement, Vishal Songh, Ashish Kumar Saxena, IJERSTE, ISSN: 2319-7463 Vol. 3 Issue 2, February-2014, pp: (225-231).

[3] Differentiated Services, http://en.wikipedia.org/wiki/Differentiated_services, Nov 2014, pp: (1-4).

[4] Trace Analyzer for NS-2, SCOReD-2006, Shah Alam, Selangor, MALAYSIA, 27-28 June, 2006, pp: (29-32).

[5] Ns2, http://en.wikipedia.org/wiki/Ns_(simulator), Jan 2015, pp: (1-4).

[6] OMNET++ Community – www.omnetpp.org, Andras Varga and OMNET++ Team 2001-13, pp: 1

[7] AODV Routing, C. Perkins Nokia Research Center E. Belding-Royer University of California, Santa Barbara S. Das University of Cincinnati July 2003, rfc3561, pp: (4-9).

[8] Inetrnet Protocol, Information Sciences Institute University of Southern California 4676 Admiralty Way Marina del Rey, California, rfc791, pp: (11-14).