<u>CNT5008 – Computer Communication Networks Architecture</u>

<u>Back-off Protocols in 802.11 WLANs</u>

Project Report

Name – Nishit Prasad

Introduction

In 802.11 Wireless Networks, the nodes are mobile, that is, the nodes do not have specific constant location. These nodes transmit data to each other taken into consideration the status of transmission channel(s). If the channel(s) is/are busy, then there is a random delay of the packets to be delivered. This random delay is known in other words as random back-off access protocol. Each packet or a frame after a collision (due to congestion in the network) experience a back-off range interval which implies a delay for the respective packet to be retransmitted. After every collision, the new back-off range value increases based on the old back-off range value, which is defined by the given expression:

New_limit = 2*(Prev_Limit + 1) – 1

For example if the backoff range for a frame is [0,7] then a new collision will increase the backoff range of that frame to [0,15].

Simulation Package Used

Network Simulator 2 (ns-2) is a discrete event network simulator that creates an open simulation environment for computer networking. The network simulator version is ns-2.35. The source code involved are C++ and TCL (stands for - “Tool Command Language” - a scripting language which provides ability to communicate among applications).

Simulation Details and Output Results

For 802.11 WLANs, the 'backend' code of this simulation tool are present. The following are the list of files involved (present in the ns-2.35 package) for 802.11 WLANs :

- **tcl/lan/ns-mac.tcl**
- **tcl/lib/ns-lib.tcl**
- **mac/mac-802_11.h**
- **mac/mac-802_11.h**
- **tcl/lib/ns-default.tcl**

The random back-off protocol is defined via contention window, which expands more based on the more number of repeated collisions. Under the header file named **mac-802_11.h,** the declaration and definition of back-off protocol is present:

```
inline void inc_cw(){

    cw_ = (cw_ << 1) + 1;

    if(cw_> phymib_.getCWMax())

        cw_= phymib_.getCWMax();

}
```

The function `inc_cw()` has the functionality of increasing the contention window (cw_), that is, the random-back-off delay. Here as we see, bitwise operator “<<” is used which means when a collision occurs, the new limit value of cw_ increases exponentially (by 2) by shifting each bit to the left. This back-off protocol in ns2 is also known as Binary Exponential Back-off.

The following are the parameters considered for output graphs, based on the number of nodes:

- Packet-Delivery Ratio (in %) Vs. Number of Nodes
  - Calculated as

    Number of Received Packets/Number of Sent Packets
- Average End-to-End Delay (in milliseconds) Vs. Number of Nodes\
  - Calculated as

    End Time of successfully received packet – Start Time of successfully sent packet.

These outputs are plotted based on the constraint of the change in the contention window size calculation. The following are the contention window limit expressions considered:

CW_Original	**:**	**cw_ = (cw_ << 1) + 1**
CW_Modified 1	**:**	**cw_ = (cw_ << 2) + 1**
CW_Modified 2	**:**	**cw_ = (cw_ * 1.5) + 1**

The scenario is tested via TCL Script file named **"Backoff_Protocol_Test.tcl".**

This file is run via command prompt as (after reaching to the specific directory):

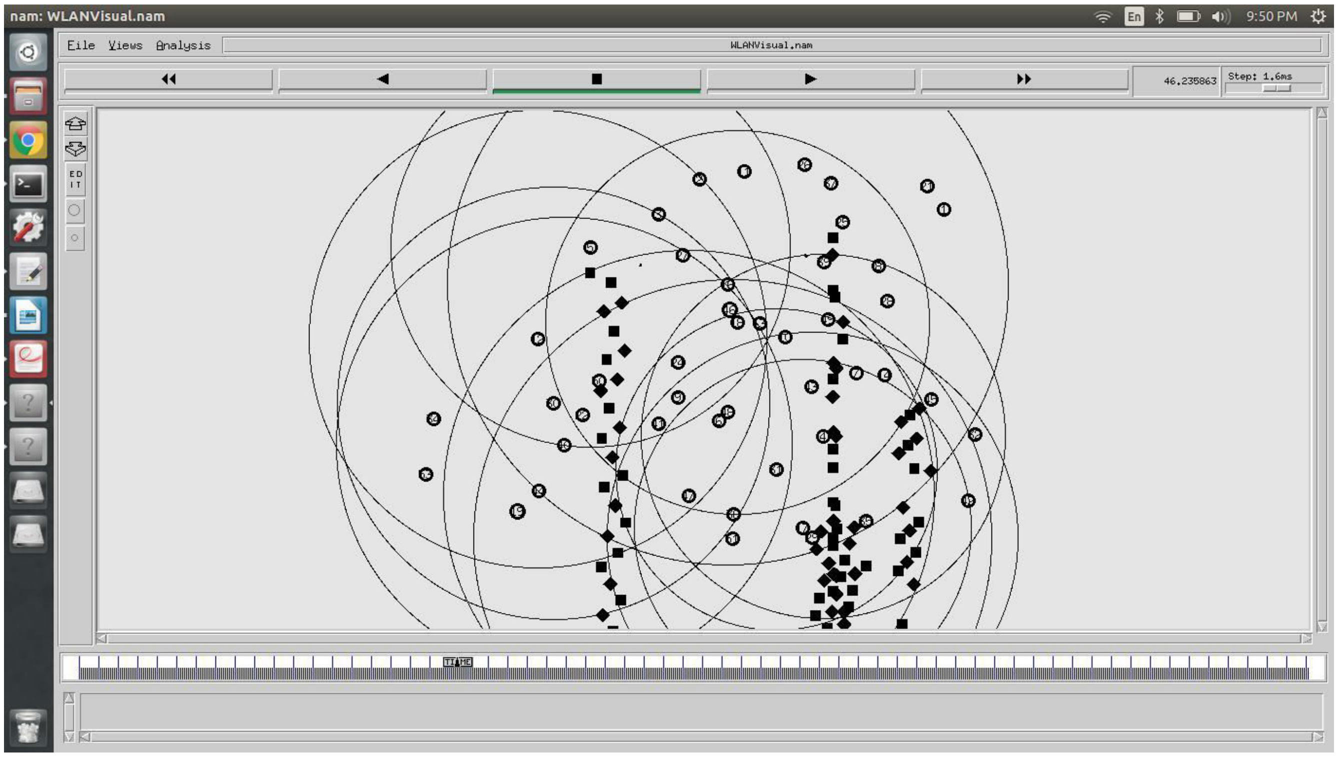
```
ns Backoff_Protocol_Test.tcl
```

After successul execution of this script, two files are generated named :

WLANTrace.tr	:	Traces the whole simulation scenario (log)
WLANVisual.nam	:	Visualizes the simulation scenario via ns-2 nam.

Each scenario is changed via number of nodes (changed within the TCL Script file) as well as the contention window size limit.

The following figure shows the visualization of wireless mobile nodes (of a particular scenario) in which the packets are transmitted as well as packets are dropped:



The calculations are made from the simulation trace files (log files) generated in each scenario. The calculation is done via AWK which is an interpreted programming language designed for text processing. Here it is used for trace file processing in order to calculate the respective performance metrics. The following is the file name followed by the command line execution for a particular simulation scenario:

Filename : **Performance_Metrics_Calc.awk**

Command Line: **gawk -f Performance_Metrics_Calc.awk WLANTrace.tr**

The following calculations are presented in table as well as in graph format:

1. Packet-Delivery Ratio (in %) Vs. Number of Nodes

Contention Window Expression (original): **cw_ = cw_<< 1 + 1**

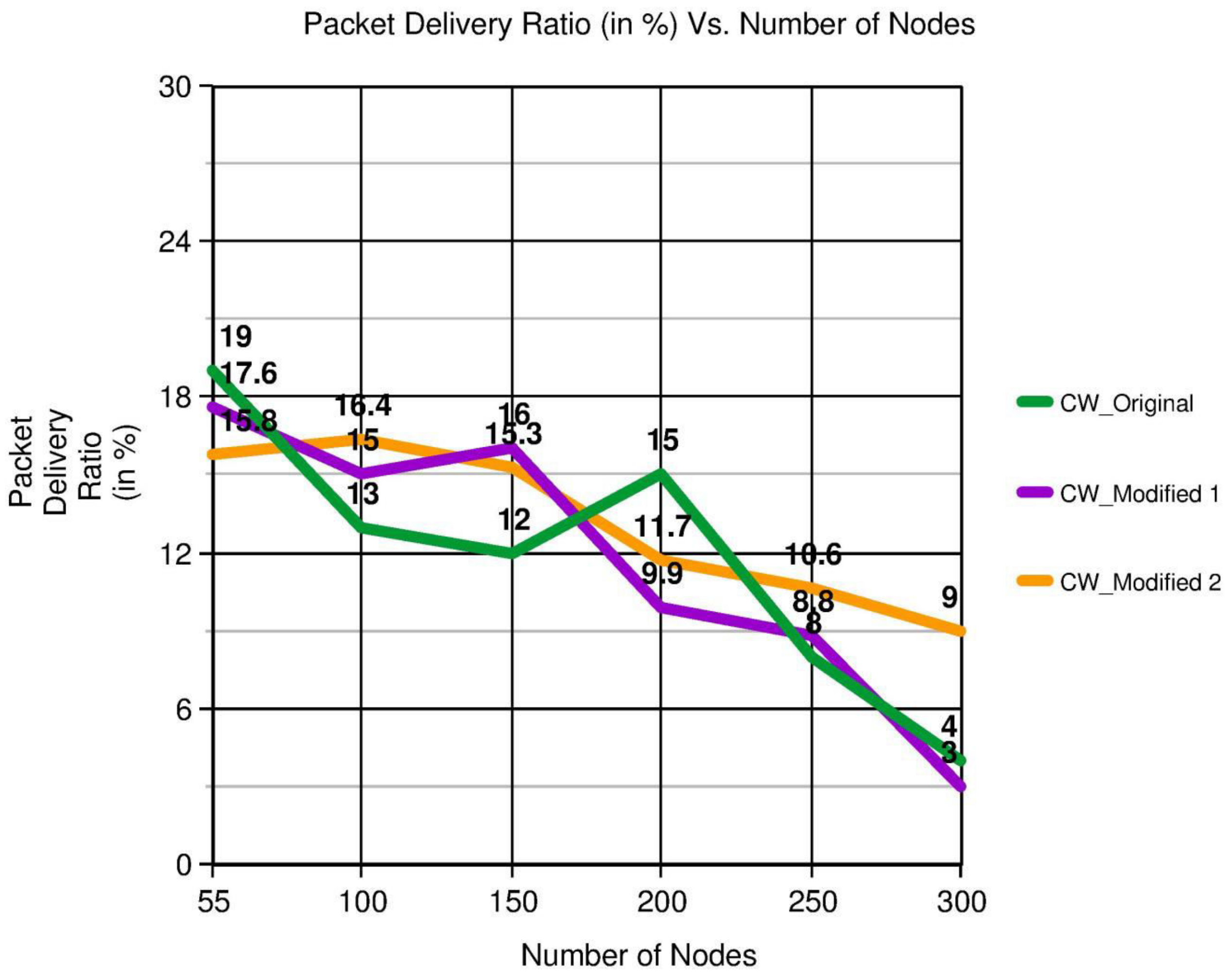
Number of Nodes	Packet-Delivery Ratio (in %)
55	19%
100	13%
150	12%
200	15%
250	8%
300	4%

Contention Window Expression (original): **cw_ = cw_<< 2 + 1**

Number of Nodes	Packet-Delivery Ratio (in %)
55	17.6%
100	15%
150	16%
200	9.9%
250	8.8%
300	3%

Contention Window Expression (modified): **(cw_ = cw_ * 1.5) + 1**

Number of Nodes	Packet-Delivery Ratio (in %)
55	15.8%
100	16.4%
150	15.3%
200	11.7%
250	10.6%
300	9.7%



1. Average End-to-End Delay(in ms) Vs. Number of Nodes

<u>Contention Window Expression (original):</u> **cw_ = cw_<< 1 + 1**

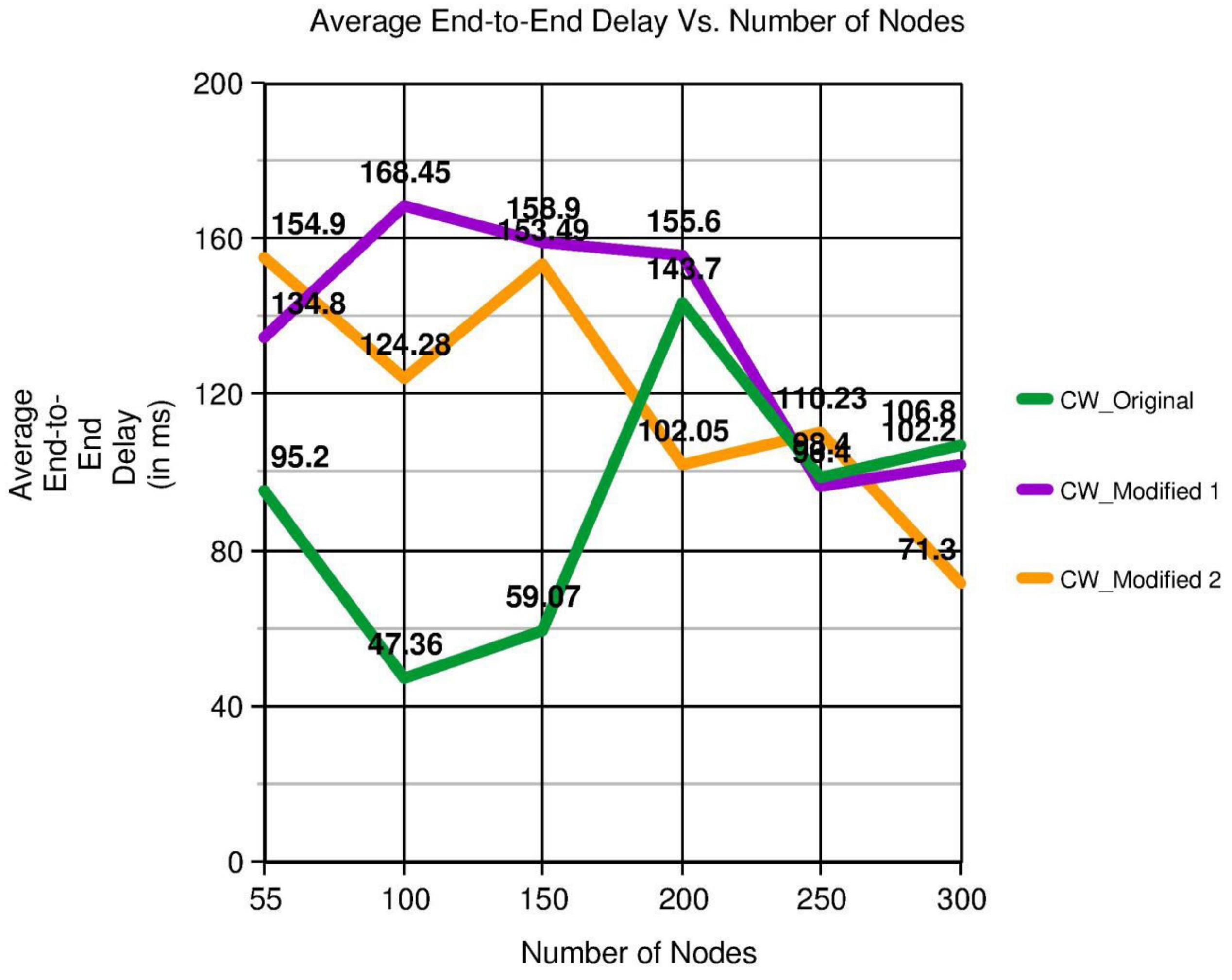
Number of Nodes	Average End-to-End Delay (in ms)
55	95.2
100	47.36
150	59.07
200	143.7
250	98.4
300	106.8

Contention Window Expression (Modified): **cw_ = cw_<< 2 + 1**

Number of Nodes	Average End-to-End Delay (in ms)
55	134.8
100	168.45
150	158.9
200	155.6
250	96.4
300	102.2

Contention Window Expression (modified): **(cw_ = cw_ * 1.5) + 1**

Number of Nodes	Average End-to-End Delay (in ms)
55	154.9
100	124.28
150	153.49
200	102.05
250	110.23
300	71.3



Conclusion

As it can be seen based on analysis, the three different contention window size change scenarios, that is, the different random backoff delay scenarios are executed and the corresponding performance metrics are plotted. We see, as the number of nodes increases, the contention in the channel increases, but due to change in the contention window sizes, the delay correspondingly decreases to some extent. In case of Packet Delivery Ratio, the number of successfully received packets decreases as the number of nodes increases (contention increases