REACTION MECHANISM FOR PACKET SIZE-BASED MISBEHAVIOR
IN WIRELESS NETWORKS

A Thesis by
Sree Sharanya Govindarajulu
Bachelors of Technology, Amrita Institute of Technology, 2008

Submitted to the Department of Electrical Engineering and Computer Science
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

May 2011
The following faculty members have examined the final copy of this thesis for form and content, and recommend that it be accepted in partial fulfillment of the requirement for the degree of Master of Science with a major in Electrical Engineering.

________________________________________________________________________

Neeraj Jaggi, Committee Chair

________________________________________________________________________

Bin Tang, Committee Member

________________________________________________________________________

Hamid Lankarani, Committee Member
ACKNOWLEDGEMENTS

First and foremost, I thank my research advisor, Dr. Neeraj Jaggi, for his patience and timely guidance throughout my progress in this research work. Without his advice and effort, completing this thesis would have been only a dream. I also convey my regards to Dr. Ravi Pendse, who developed my interest in the field of networking.

I also extend my thanks to Vamshi and other research associates for sharing their ideas and also giving valuable suggestions for improvising this thesis. For more than six years of friendship, I am grateful to Janani who helped me with my thesis and for all the fun we had during this time.

Last and most importantly, I wish to thank my parents, Govindarajulu and Manonmani, who bore me, raised me, supported me, and loved me. I would also like to thank Selva, who gave me support and confidence to complete my thesis, and the emotional strength to overcome all the hurdles I faced. To them I dedicate this thesis.
ABSTRACT

Since the field of wireless technology is growing rapidly, security is becoming a major concern. A variety of security problems are being addressed, and much research work is taking place in order to provide adequate security to prevent hackers from disrupting network service. Wireless networks follow the IEEE 802.11 standard to transmit and receive packets. The IEEE 802.11 MAC protocol is designed in such a way to provide an equal share of throughput among all nodes in a network. Users who misbehave could modify the IEEE 802.11 MAC protocol, thus causing major security threats including substantial bandwidth degradation of other users.

This thesis addresses the misbehavior of a node caused by altering the packet size. For a node to acquire higher throughput compared to other genuine nodes in the network, its packet size could be set higher than that of the genuine nodes. In order to protect against this sort of misbehavior, a special algorithm, which is a slight modification of the IEEE 802.11 MAC protocol, was developed.

This algorithm is based on the notion of receiver-assigned backoff, which has already been used to deal with other types of misbehavior. The packet size-based misbehavior was modeled mathematically using queuing theory, and an appropriate reaction strategy was deduced from the analytical results. It was shown that the proposed approach reduces the effectiveness of misbehavior and leads to fairness in the network.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>INTRODUCTION</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Classification of Wireless Networks</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Wireless MAC Protocols</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Types of Misbehaviors and Prevention Mechanisms</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>Contributions of Thesis</td>
<td>6</td>
</tr>
<tr>
<td>1.4</td>
<td>RELATED WORK</td>
<td>8</td>
</tr>
<tr>
<td>2.</td>
<td>PACKET SIZE-BASED MISBEHAVIOR</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>Range of Packet Size Considered</td>
<td>13</td>
</tr>
<tr>
<td>3.3</td>
<td>Experimental Setup</td>
<td>14</td>
</tr>
<tr>
<td>3.4</td>
<td>Effectiveness of Misbehavior</td>
<td>15</td>
</tr>
<tr>
<td>3.5</td>
<td>Summary</td>
<td>17</td>
</tr>
<tr>
<td>3.6</td>
<td>MODELING AND ANALYSIS</td>
<td>18</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>18</td>
</tr>
<tr>
<td>4.2</td>
<td>Wireless LAN Model</td>
<td>18</td>
</tr>
<tr>
<td>4.3</td>
<td>M/M/1/K Queuing System</td>
<td>20</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Parameters</td>
<td>21</td>
</tr>
<tr>
<td>4.3.2</td>
<td>State Transition Diagram</td>
<td>22</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Steady-State Equations</td>
<td>25</td>
</tr>
<tr>
<td>4.4</td>
<td>Effectiveness of Packet Size-Based Misbehavior</td>
<td>25</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Three-Node Scenario</td>
<td>25</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Nine-Node Scenario</td>
<td>30</td>
</tr>
<tr>
<td>4.5</td>
<td>Model to Reduce Impact of Misbehavior</td>
<td>35</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Parameters</td>
<td>35</td>
</tr>
<tr>
<td>4.5.2</td>
<td>State Transition Diagram</td>
<td>37</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Steady-State Equations</td>
<td>38</td>
</tr>
<tr>
<td>4.6</td>
<td>Impact of Using Varied Service Rates</td>
<td>39</td>
</tr>
<tr>
<td>4.7</td>
<td>Summary</td>
<td>43</td>
</tr>
<tr>
<td>5.</td>
<td>DESIGN AND IMPLEMENTATION OF REACTION MECHANISM</td>
<td>44</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>44</td>
</tr>
<tr>
<td>5.2</td>
<td>Algorithm to Reduce Effect of Misbehavior</td>
<td>44</td>
</tr>
<tr>
<td>5.3</td>
<td>Flowchart for Reaction Mechanism</td>
<td>46</td>
</tr>
<tr>
<td>5.4</td>
<td>IEEE 802.11 MAC Protocol Implementation in NS-2</td>
<td>47</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>Code Changes in NS-2 ................................................................. 48</td>
</tr>
<tr>
<td>5.6</td>
<td>Impact of Reaction Mechanism ......................................................... 50</td>
</tr>
<tr>
<td>5.7</td>
<td>Summary .......................................................................................... 54</td>
</tr>
<tr>
<td>6.</td>
<td>CONCLUSIONS ..................................................................................... 55</td>
</tr>
<tr>
<td>6.1</td>
<td>Future Work ...................................................................................... 56</td>
</tr>
<tr>
<td>REFERENCES .......................................................... 58</td>
<td></td>
</tr>
<tr>
<td>APPENDICES ......................................................... 60</td>
<td></td>
</tr>
<tr>
<td>A.</td>
<td>MATLAB Code .................................................................................. 61</td>
</tr>
<tr>
<td>B.</td>
<td>TCL Script ....................................................................................... 65</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Three Nodes with Well-Behaving Node’s Packet Size Set to 512 bytes</td>
<td>26</td>
</tr>
<tr>
<td>2.</td>
<td>Three Nodes with Well-Behaving Node’s Packet Size Set to 1,024 bytes</td>
<td>28</td>
</tr>
<tr>
<td>3.</td>
<td>Three Nodes with Well-Behaving Node’s Packet Size Set to 2,048 bytes</td>
<td>29</td>
</tr>
<tr>
<td>4.</td>
<td>Nine Nodes with Well-Behaving Nodes’ Packet Size Set to 512 bytes</td>
<td>31</td>
</tr>
<tr>
<td>5.</td>
<td>Nine Nodes with Well-Behaving Nodes’ Packet Size Set to 1,024 bytes</td>
<td>33</td>
</tr>
<tr>
<td>6.</td>
<td>Nine Nodes with Well-Behaving Nodes’ Packet Size Set to 2,048 bytes</td>
<td>34</td>
</tr>
<tr>
<td>7.</td>
<td>Well-Behaving Nodes’ Packet Size Set to 512 Bytes with Varied Service Rates</td>
<td>40</td>
</tr>
<tr>
<td>8.</td>
<td>Well-Behaving Nodes’ Packet Size Set to 1,024 Bytes with Varied Service Rates</td>
<td>41</td>
</tr>
<tr>
<td>9.</td>
<td>Well-Behaving Nodes’ Packet Size Set to 1,024 Bytes with Varied Service Rates</td>
<td>42</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure | Description                                                                 | Page
-------|-----------------------------------------------------------------------------|------
1.     | DCF mechanism                                                               | 4    
2.     | MAC frame format                                                            | 14   
3.     | Well-behaving nodes’ packet size set to 512 bytes                           | 15   
4.     | Effectiveness achieved by misbehaving node when fixed packet size is 512 bytes | 15   
5.     | Well-behaving nodes’ packet size set to 1,024 bytes                         | 16   
6.     | Effectiveness achieved by misbehaving node when fixed packet size is 1,024 bytes | 16   
7.     | Well-behaving nodes’ packet size set to 2,048 bytes                         | 16   
8.     | Effectiveness achieved by misbehaving node when fixed packet size is 2,048 bytes | 16   
9.     | Weighted fair queuing system                                                | 20   
10.    | State transition diagram                                                    | 23   
11.    | Three nodes with well-behaving node’s packet size set to 512 bytes          | 27   
12.    | Three nodes with well-behaving node’s packet size set to 1,024 bytes        | 28   
13.    | Three nodes with well-behaving nodes’ packet size set to 2,048 bytes        | 30   
14.    | Nine nodes with well-behaving nodes’ packet size set to 512 bytes           | 32   
15.    | Nine nodes with well-behaving nodes’ packet size set to 1,024 bytes         | 33   
16.    | Nine nodes with well-behaving nodes’ packet size set to 2,048 bytes         | 34   
17.    | Queuing system with varied service rate                                     | 35   
18.    | State transition diagram with varied service rate                           | 38   
19.    | Well-behaving nodes’ packet size set to 512 bytes with varied service rate  | 41   
20.    | Well-behaving nodes’ packet size set to 1,024 bytes with varied service rate | 42   
21.    | Well-behaving nodes’ packet size set to 2,048 bytes with varied service rate | 43   

ix
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.</td>
<td>Flowchart for reaction mechanism</td>
<td>47</td>
</tr>
<tr>
<td>23.</td>
<td>IEEE 802.11 mechanism with fixed-packet size of 512 bytes</td>
<td>50</td>
</tr>
<tr>
<td>24.</td>
<td>Reaction mechanism with fixed-packet size of 512 bytes</td>
<td>51</td>
</tr>
<tr>
<td>25.</td>
<td>Effectiveness achieved by misbehaving node without reaction mechanism (512 bytes)</td>
<td>51</td>
</tr>
<tr>
<td>26.</td>
<td>Effectiveness achieved by misbehaving node with reaction mechanism (512 bytes)</td>
<td>51</td>
</tr>
<tr>
<td>27.</td>
<td>IEEE 802.11 mechanism with fixed-packet size of 1,024 bytes</td>
<td>52</td>
</tr>
<tr>
<td>28.</td>
<td>Reaction mechanism with fixed-packet size of 1,024 bytes</td>
<td>52</td>
</tr>
<tr>
<td>29.</td>
<td>Effectiveness achieved by misbehaving node without reaction mechanism (1,024 bytes)</td>
<td>52</td>
</tr>
<tr>
<td>30.</td>
<td>Effectiveness achieved by misbehaving node with reaction mechanism (1,024 bytes)</td>
<td>52</td>
</tr>
<tr>
<td>31.</td>
<td>Standard IEEE 802.11 with fixed-packet size of 2,048 bytes</td>
<td>53</td>
</tr>
<tr>
<td>32.</td>
<td>Reaction mechanism with fixed-packet size of 2,048 bytes</td>
<td>53</td>
</tr>
<tr>
<td>33.</td>
<td>Effectiveness achieved by misbehaving node without reaction mechanism (2,048 bytes)</td>
<td>54</td>
</tr>
<tr>
<td>34.</td>
<td>Effectiveness achieved by misbehaving node with reaction mechanism (2,048 bytes)</td>
<td>54</td>
</tr>
</tbody>
</table>
LIST OF ABBREVIATIONS

ACK  Acknowledgement
AP   Access Point
BEB  Binary Exponential Backoff
CBR  Constant Bit Rate
CSMA/CA Career Sense Multiple Access/Collision Avoidance
CTS  Clear-to-Send
CW   Contention Window
DCF  Distributed Coordination Function
DIFS DCF Inter-Frame Space
EIFS Extended Inter-Frame Space
IFS  Inter-Frame Space
LAN  Local Area Network
MAC  Media Access Control
NAV  Network Allocation Vector
NS-2 Network Simulator
PCF  Point Coordination Function
PHY  Physical
PIFS PCF Inter-Frame Space
PRB  Predictable Random Backoff
RTS  Request-to-Send
QOS  Quality of Service
SIFS Short Inter-Frame Space
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>WFQ</td>
<td>Weighted Fair Queuing</td>
</tr>
</tbody>
</table>
The field of computer networks has grown extensively in the last three decades, and wireless networking has become one of the upcoming and interesting fields of research. Before the emergence of wireless networks, all computers were connected via physical links. To eliminate wiring, researchers explored the use of electromagnetic waves, such as radio waves, which led to the development of wireless networks. Examples of wireless technology include cellular telephones, global positioning systems, sensors, satellite television, and wireless local area networks (LANs). As these technologies keep emerging, security has become a major concern. Wireless networks are easy to breach, and it is important that organizations with wireless networks take appropriate measures to guard against these security threats and protect important information. This chapter provides an overview of the classification of wireless networks, wireless media access control (MAC) protocols such as IEEE 802.11 for sharing a wireless channel, security threats, and measures to control the security threats.

1.1 Classification of Wireless Networks

Wireless networks are broadly classified into two types,

- **Infrastructure-Based Network**: An infrastructure-based network has access points (APs) that are connected via existing networks. APs can interact with wireless nodes and also with a backbone wired network. All wireless nodes communicate via these access points.

- **Adhoc Network**: Contrary to the infrastructure-based network, an adhoc network is a decentralized wireless network, i.e., it does not have a base station or access points. All nodes communicate directly with each other or forward messages through other nodes that are directly accessible.
1.2 Wireless MAC Protocols

In wireless networks, nodes share a common broadcast channel. The bandwidth available for communication in such wireless networks is limited. As a result, access to the shared medium is controlled in such a way that all nodes receive an equal share of bandwidth. To address this problem media access control protocols have been designed for wireless networks.

The IEEE 802.11 MAC protocol describes how the nodes present in a wireless LAN should access the broadcast channel for transmitting data to other nodes. This protocol has two contention-resolution mechanisms to control the nodes competing for channel access: distributed coordination function (DCF) and point coordination function (PCF). The DCF does not use any kind of centralized control, while the PCF uses a centralized controller to coordinate the activity of all nodes and is used only in infrastructure-based networks. Most wireless LANs use the DCF mode of operation, whereas the PCF is optional.

The time interval between the transmissions of two consecutive frames is called the inter-frame space (IFS). The IEEE 802.11 standard defines four types of IFSs: short inter-frame space (SIFS), DCF inter-frame space (DIFS), PCF inter-frame space (PIFS), and extended inter-frame space (EIFS).

This research considered the distributed coordination function mode of operation in the wireless LAN topology. According to Murthy and Manoj [9], in the DCF mode of operation, all nodes contend for channels simultaneously. Career sense multiple access/collision avoidance (CSMA/CA) with the binary exponential backoff (BEB) algorithm was used to reduce collisions in a wireless LAN medium.
• **Carrier Sense Multiple Access/Collision Avoidance**: CSMA/CA is designed to reduce the collision probability among nodes in a network. Here, two carrier-sensing mechanisms, physical and virtual, are defined as follows:

  o **Physical (PHY) Carrier Sensing**
    
    ➢ Depending on the PHY layer, this mechanism senses the availability of the carrier frequency.

  o **Virtual Carrier Sensing**
    
    ➢ This logical carrier sensing occurs at the MAC layer.
    
    ➢ Every packet (with some exceptions) announces the duration for which current transmission will hold the channel, i.e., network allocation vector (NAV).
    
    ➢ All stations monitoring the channel read the MAC header, which contains the NAV. They all “back off” from the NAV before starting the contention for the next transmission.

If the node is about to transmit in a channel and detects transmission, then the listening node enters a deferral period. This is the CA portion of the access mechanism.

• **Binary Exponential Backoff Algorithm**: If node A wants to transmit to node B, then the following occurs:

  o If node A senses the channel to be idle, then node A waits for a DIFS period and invokes a backoff timer, which is given as

    \[
    \text{Backoff time} = \text{rand} \ (0, \ CW-1) \ * \ \text{slottime} \tag{1}
    \]

    where slottime includes the following: time needed for a node to detect a frame, propagation delay, time needed to switch from the receiving state to the transmitting state, and time to signal the state of the channel to the MAC layer. Then, rand \ (0,
CW-1) returns a pseudo-random integer from an interval (0, CW-1). The contention window (CW) value plays a vital role in determining the backoff value. The minimum value is CW_{min} and maximum is CW_{max}. If a collision occurs, then the CW value is doubled.

- If node A senses that a channel is busy, then it freezes the backoff counter value and starts decrementing it only after it senses that the channel is idle for DIFS amount of time.
- In the same scenario, if another node C apart from node A also senses the channel to be idle and both of them transmit an RTS frame at the same time, then collision occurs. In this case, both nodes double the contention window and start the backoff procedure again.

- Distributed Coordination Function Mechanism: Figure 1 represents the DCF mechanism.

![DCF mechanism diagram](image)

Figure 1. DCF mechanism

If node A senses the channel to be idle for DIFS amount of time, it waits for the backoff period and then transmits a request-to-send (RTS) and waits for the clear-to-send (CTS) from node B. If there is no interference from other nodes, then node B acknowledges the RTS
received from the sender by sending a CTS frame. After receiving the CTS, node A sends the data followed by an acknowledgement (ACK) frame from node B. Upon hearing the RTS/CTS, other nodes must adjust their network allocation vector to defer their transmission for the duration of time specified in the RTS/CTS in order to avoid collision.

1.3 Types of Misbehaviors and Prevention Mechanisms

The host misbehaviors in wireless networks can be classified in two categories: selfish and malicious.

- **Selfish Misbehavior:** This type of misbehavior is defined as using the MAC protocol in order to gain greater network resources than that of well-behaved hosts [1,2]. The node can benefit from this behavior by the following:
  - Obtaining a larger portion of channel capacity (and consequently higher throughput).
  - Achieving improved quality of service, i.e., low network latency.
  - Conserving energy by dropping packets from well-behaved nodes rather than forwarding them to destined nodes in the network.

- **Malicious Misbehavior:** Unlike selfish misbehavior, malicious misbehavior disrupts the normal operation of the network. Malicious nodes unnecessarily send data in order to deplete the channel capacity for the well-behaving nodes [1,2].

Immense research is going on in the field of misbehavior detection and reaction in wireless networks. Researchers have addressed various types of misbehaviors that alter the backoff value [1, 2], SIFS parameter, oversized NAV, and DIFS parameter in IEEE 802.11 [3]. These investigations have suggested detection and also prevention mechanisms for these types of misbehaviors.
Kyasanur and Vaidya [2] researched the misbehavior of a node by altering its backoff value. To address this kind of misbehavior, a slight modification was made to the IEEE 802.11 MAC protocol. In the standard MAC protocol, the sender computes the backoff value for transmission of its data, but these authors made the receiver assign a backoff value to the sender in order to reduce the effect of misbehavior. Guang et al. [1] addressed the same problem using a predictable random backoff value to reduce the effectiveness of misbehavior. Other prevention mechanisms are discussed in Chapter 2.

1.4 Contributions of Thesis

This thesis considered a new type of selfish misbehavior that varies the packet size of the data sent by a node in a wireless network. Here, the node trying to misbehave selects a packet size greater than other nodes in order to achieve higher throughput. A variety of scenarios were considered to show the effect of misbehavior achieved by altering the packet size.

Packet size-based misbehavior was modeled as a two-class single-server M/M/1/K queue with finite capacity under the weighted fair queuing (WFQ) discipline [10]. The system was modeled with two classes—one class representing packets from well-behaving nodes and the other class representing packets from misbehaving nodes.

The server used a service rate ($\mu$) for all nodes in the network, and the throughput achieved by each node was determined. It was shown that the misbehaving node (node with higher packet size) achieves higher throughput compared to the well-behaving node. To reduce the effect of misbehavior and to provide fairness, the service rate was varied for both queues. The queue involving packets from the misbehaving node was provided lesser service, and the queue representing a well-behaving node was provided higher service.
The impact of packet size-based misbehavior was investigated using the network simulator (NS-2). Results show that if the packet size was higher, then a node achieved higher throughput. In order to reduce the effect of misbehavior, a receiver-assigned backoff scheme [2] was modified to take into account the packet size of the sender. Service rates from the WFQ system analysis were used to appropriately design the reaction strategy for the packet size-based misbehavior. It was demonstrated through simulations that this strategy reduced the effect of misbehavior and was able to ensure fairness in the network.
CHAPTER 2
RELATED WORK

Researchers have developed many detection and prevention mechanisms to mitigate the effect of misbehavior and improve the network performance. As mentioned previously, misbehavior is broadly classified into two types: selfish and malicious.

- *Selfish misbehavior* occurs when a selfish host misbehaves to attain enhanced performance: better throughput, reduced latency, or energy conservation.
- *Malicious misbehavior* by a node disrupts the normal network behavior using jamming or denial of service. Here, the node does not reap any benefit for itself, but rather interrupts the network operation and degrades the overall network performance.

Several approaches have been proposed to detect and prevent these misbehaviors. Raya, et al. [4], classified different types of MAC layer misbehaviors by alterations as follows:

- Shorter DIFS
- Oversized NAV
- Modified backoff values

In this research, the authors classified each type of misbehavior and proposed a DOMINO (system for detection of greedy behavior in the MAC layer of 802.11 public networks) algorithm to detect misbehavior. DOMINO is software that is installed at the access point and does not need any modification to the IEEE 802.11 MAC protocol to address misbehavior. Traffic from the sending station is captured periodically in regular intervals of time, or monitoring periods. The gathered data are passed through the DOMINO algorithm to detect if the corresponding transmitter has misbehaved. This algorithm is implemented only at the access point and helps to detect the greedy misbehavior.
Kyasanur and Vaidya [2] addressed the type of misbehavior caused by changing the backoff value. They found that a misbehaving node regularly selects a lower backoff value compared to other nodes in the network. This results in gaining easy access to the channel and thereby achieving higher throughput. In order to allay this problem, the receiver assigns a backoff value \([B_{\text{exp}}]\) to the sender, and the sender has to use the assigned backoff value for its subsequent transmission. The receiver monitors the channel and checks to see if the sender is deviating from the assigned backoff value \([B_{\text{exp}}]\), that is,

\[
\text{Actual Backoff Value } [B_{\text{act}}] < \alpha \times \text{Expected Backoff Value } [B_{\text{exp}}] \quad 0 < \alpha \leq 1
\]

where the actual backoff value is the value taken by the sender for transmitting the packet, and the expected backoff value is the value assigned by the receiver.

If the sender has modified its assigned backoff value, then the receiver increases the penalty (i.e., increases the backoff value for subsequent transmission). But if the sender continues to misbehave in the same way, the receiver drops the received packet when the difference in backoff values \([B_{\text{act}} - B_{\text{exp}}]\) reaches a threshold. Hence, there is a disincentive for the sender to misbehave. Therefore, the sender must follow the receiver-assigned backoff value; if it continues to misbehave, then the receiver drops the packet from the sender.

The other issue addressed by Kyasanur and Vaidya [2] is what happens if the receiver misbehaves. The receiver may assign a lower backoff value to one particular node if it prefers to receive data from that particular sender. This type of misbehavior can be approached by using multiple observers to monitor the channel and participate in reducing the effectiveness of the misbehavior.

Guang et al. [1] addressed the same kind of sender misbehavior (choosing a lower backoff value compared to other nodes). In order to access the channel, the selfish node tries to
manipulate its backoff value in different ways, either by generating a small backoff value, e.g., using the range \((0, \text{CW}/2)\) rather than \((0, \text{CW})\), or by generating a small random value regardless of the range.

To address the above misbehavior, a slight modification was made to the BEB algorithm. In this algorithm, the lower bound for the transmission was selected from the current contention window \((0, \text{CW}-1)\), whereas in the predictable random backoff (PRB), the lower bound for the transmission depends on the previously used backoff value. If the node used a small backoff value in its previous transmission, then this time-lower bound would be increased from zero to a higher value. For example, instead of \((0, \text{CW} -1)\), the contention window would be \((5, \text{CW}-1)\).

PRB operates as follows:

- The node having a data packet to transmit randomly chooses a backoff value from the range \((0, \text{CW}-1)\).
- Upon successful transmission, if the selected backoff value is less than the threshold value, then the sender has misbehaved and its lower-contention bound value for the next transmission has been increased (e.g., instead of \((0, \text{CW}-1)\), it will be \((5, \text{CW}-1)\))
- If there is collision, the contention window is doubled.

As a result of this slight change in the algorithm, if the selfish node follows a PRB, the negative impact it has on the network performance will be mitigated, regardless of the attack strategies. And if the selfish node does not follow a PRB, since the backoff is predictable, the receiver can easily detect the misbehavior of the sender and perform immediate action, such as dropping the packet from the misbehaving sender.

Giri and Jaggi [5] categorized various kinds of misbehaviors that result from altering the backoff value: \(\alpha\)-misbehavior, deterministic backoff misbehavior, \(\beta\)-misbehavior, fixed
maximum contention window misbehavior, and fixed contention window misbehavior. They categorized the effectiveness of misbehavior by using the following formula:

\[ e = \frac{t_m - t_g}{t_g} \times 100 \]  

(2)

The parameter \( e \) measures the magnitude of misbehavior exhibited by the selfish node, and \( t_g \) denotes the throughput of the node when all nodes are genuine. After introducing one misbehaving node, \( t_m \) denotes the throughput of the misbehaving node when all other nodes are still genuine. In reaction to this misbehavior, all other well-behaving nodes in the network estimate the level of misbehavior of the selfish node and try to replicate the same misbehavior. As a result of this approach, the selfish node achieves less throughput compared to what it would have achieved without misbehaving.

Guang et al. [6] characterized a new type of misbehavior achieved by altering the timeout value. The distributed coordination function defines three types of time intervals: DIFS, SIFS, and EIFS. Before the transmission of any frame, a node must observe a quiet medium for one of the above window periods. DIFS is used for nodes initiating a new frame. According to Figure 1, the short interframe space is used as part of a pre-existing frame exchange before transmitting the data and control packets such as CTS and ACK. A node can misbehave by altering any of the interval values. In this research, the authors classified the misbehavior by altering the SIFS timeout value.

The malicious receiver selects a larger SIFS value and deliberately delays the transmission of CTS/ACK until its corresponding timeout has expired at the transmitter. Therefore, the transmitter is forced to timeout after transmitting the RTS and data frame. After consequent unsuccessful transmissions, the transmitter drops the data packet. The malicious nodes behaving in this way disrupt the route discovery process and are forced to choose the non-
optimal routes. The transmitter also conserves battery power by refusing to forward the packets. Since the data flow is forced away, the malicious node can use the channel with less contention. Similarly, the transmitter can maliciously misbehave by choosing a smaller SIFS value. As a result, the transmitter will not receive a CTS within the SIFS timeout, and hence, the receiver keeps on receiving the RTS frame instead of the data frame.

Guang et al. proposed an identification and reaction mechanism against this sort of malicious misbehavior. They formulated a detection mechanism for both misbehaving transmitter and receiver. If the receiver receives a second RTS, it knows that the transmitter would have misbehaved, so it increases the bad credit (i.e., increases the timeout value by a constant). If the transmitter continues to behave in the same way, the receiver counts the number of RTSs received, and if this number exceeds the specified threshold, then the receiver notes that the transmitter has misbehaved. A similar type of procedure is followed at the transmitter, if the receiver is misbehaving. Therefore, this approach reduces the malicious misbehavior caused by altering the SIFS parameter.

As part of this research work, a new type of misbehavior caused by varying the packet size was investigated. Typically, all nodes in the network use a fixed packet size, and a node trying to achieve higher throughput increases the packet size rather than modifying the backoff value or timeout parameters. The research in this thesis addressed this type of misbehavior and designed a reaction strategy to reduce its effectiveness. The receiver assigning the backoff value, as discussed by Kyasanur and Vaidya [2], was used to reduce the effect of packet size-based misbehavior.
CHAPTER 3

PACKET SIZE-BASED MISBEHAVIOR

3.1 Introduction

This research considered only one node as the misbehaving node and all other nodes as behaving normally in a given wireless LAN environment. All nodes in the network followed the standard IEEE 802.11 MAC protocol and used appropriate backoff values and timeout values for transmitting packets. A node trying to misbehave did not change any of its MAC parameters. The misbehaving node trying to achieve higher throughput increased its packet size more than that of the standard packet size followed by the other nodes.

This chapter shows the throughput gained by the misbehaving node in different network scenarios and also the effectiveness of misbehavior achieved by the misbehaving node. Section 3.2 explains packet size-based misbehavior, section 3.3 describes various network scenarios, section 3.4 shows the throughput attained by each node and also the effectiveness of misbehavior, and section 3.5 presents a summary.

3.2 Range of Packet Sizes Considered

A scenario where all nodes in the network transmit data packets with fixed packet size while one node varies its packet size was considered. The node that varied its packet size was referred to as the misbehaving node, and other nodes were referred to as the well-behaving nodes. All nodes were configured to transmit data at a constant rate, including the misbehaving node.

The standard IEEE 802.11 MAC frame format for this research, shown in Figure 2, indicates that the maximum length of the data packet was 2,312 bytes [7]. The misbehaving node
was configured to use a packet size varying from 256 bytes to 2,304 bytes to show the effectiveness of misbehavior.

Figure 2. MAC frame format

3.3 Experimental Setup

The NS-2 network simulator was used in this experimentation. The network topology consisted of static wireless nodes in a 250*250 m$^2$ area. All nodes were in the communication range of each other. The number of nodes in the network was set to nine. A user datagram protocol (UDP) was used as a transport layer protocol for accepting data from the application layer. The constant bit rate (CBR) traffic generator was used at the application layer. The overall channel capacity was 2 Mbps, and the nodes sent traffic at the rate of 0.5 Mbps. The simulation time was set to 120 secs. Out of nine nodes, eight sent traffic at the rate of 0.5 Mbps to one node. Among the eight nodes, one node misbehaved by altering its packet size. The following scenarios showing the effect of misbehavior were considered:

1. The eight well-behaving nodes’ packet size was set to 512 bytes, while the misbehaving node’s packet size was varied from 256 to 2,304 bytes.
2. The eight well-behaving nodes’ packet size was set to 1,024 bytes, while the misbehaving node’s packet size was varied from 256 to 2,304 bytes.
3. The eight well-behaving nodes’ packet size was set to 2,304 bytes, while the misbehaving node’s packet size was varied from 256 to 2,304 bytes.
3.4 Effectiveness of Misbehavior

The effectiveness of misbehavior was calculated using the following equation:

\[ e = \left( \frac{t_m - t_g}{t_g} \right) \]  

(3)

where \( t_m \) is the throughput of the misbehaving node if it misbehaves, and \( t_g \) is the throughput of the node if it does not misbehave. Therefore, parameter \( e \) gave the effectiveness of misbehavior obtained in different scenarios.

Figure 3 represents the throughput achieved by both the well-behaving nodes and the misbehaving node when the well-behaving nodes’ packet size was fixed at 512 bytes. “Other” in the figure represents the average throughput of the well-behaved nodes, “misb” represents the throughput of the misbehaving node, and “total” represents the overall throughput in the network. Figure 3 clearly shows that the throughput of the misbehaving node gradually increased as the misbehaving node’s packet size increased beyond 512 bytes. The effectiveness of misbehavior achieved is shown in Figure 4.

![Figure 3. Well-behaving nodes’ packet size at 512 bytes](image1)

![Figure 4. Effectiveness achieved by misbehaving node when fixed-packet size is 512 bytes](image2)

Similarly, Figure 5 shows the throughput of both the well-behaving nodes and the misbehaving node when the fixed packet size is set to 1,024 bytes. As can be seen, as the
misbehaving node’s packet size increased higher than the fixed packet size, its throughput and corresponding effectiveness of misbehavior also increased, as shown in Figure 6.

Figure 5. Well-behaving nodes’ packet size set to 1,024 bytes

Figure 6. Effectiveness achieved by misbehaving node when fixed packet size is 1,024 bytes

Figure 7 shows that the packet size of the well-behaving nodes was fixed to 2,048 bytes, and the misbehaving node’s packet size was varied. Here the throughput of the misbehaving node remained less than the well-behaving nodes’ throughput until its packet size was less than the fixed packet size (2,048 bytes). Figure 8 shows the effectiveness of the misbehaving node.

Figure 7. Well-behaving nodes’ packet size set to 2,048 bytes

Figure 8. Effectiveness achieved by misbehaving node when fixed-packet size is 2,048 bytes
3.5 Summary

From Figures 3 to 8, it can be seen that whenever the misbehaving node increased its packet size higher than the fixed-packet size used by other nodes, it achieved higher throughput. Therefore, in order to obtain an unfair and greater throughput share, the node increased its packet size to be greater than the packet size of the well-behaving nodes. This chapter showed the effectiveness of such misbehavior attained in various network scenarios.
CHAPTER 4
MODELING AND ANALYSIS

4.1 Introduction

This chapter discusses the modeling of the packet size-based misbehavior as a two-class M/M/1/K queue with finite capacity under the weighted fair queuing discipline [10]. The Poisson process was used to model two classes of arrival streams. The service time had exponential distribution. The Markov chain based on analytical model was developed to obtain general equations to measure the throughput of the network.

In order to determine the effect of misbehavior, the server provided each node with an equal service rate. The throughput obtained by the nodes was plotted. The model was extended to reduce the effect of misbehavior by providing a lesser service rate for the misbehaving node (variable service rates for the nodes).

The chapter is organized as follows. Section 2 provides an overview of the M/M/1/K queue and WFQ discipline. Section 3 shows how the packet size-based misbehavior was modeled as an M/M/1/K queuing system, and Section 4 provides solutions to the steady-state probability equations. Section 5 discusses how the model was designed for a three-node scenario and then extended to a nine-node scenario. Section 6 shows the model that provided variable service rates for the queues to reduce the effect of misbehavior, and section 7 concludes the chapter by explaining how the variation in service rate brought fairness to the network by reducing the effect of misbehavior.

4.2 Wireless LAN Model

The following terms are defined for understanding the wireless LAN model.

- Node: Each sender and receiver in the wireless LAN environment.
• **Class**: Set of nodes possessing common characteristics (e.g., nodes sending packets of a fixed-packet size are grouped into one class, and nodes altering the packet size are grouped into another class).

• **Queue**: Each class in the M/M/1/K system model.

• **Weighted Fair Queuing System**: A data packet scheduling technique where each class is given different scheduling priorities to statistically multiplex the data flow. It is used for controlling the quality of service (QOS). In this research, the WFQ system meant each class (otherwise referred to as queue) was served with different service rates in order to bring fairness among the nodes in the network. Weights such as $w_1$ and $w_2$ were used to compute how much fraction of the service rate ($\mu$) should be assigned to each class (queue).

The WFQ system, as shown in Figure 9, assumed two classes of jobs (one class represented packets from well-behaving nodes and the other class represented packets from misbehaving nodes). The jobs of class 1 and class 2 arrived according to the Poisson process with rates $\lambda_i$, $i = 1, 2$ and required exponential service times. Each class was assigned a virtual queue, and the arriving jobs entered the virtual queue related to their classes and were served in a first-in-first-out order (classes were differentiated based on packet sizes). The total buffer capacity was limited to $K$, so if there were $m$ packets in queue 1, then the remaining $K-m$ packets could be present in queue 2. If the buffer reached its limit, then the arriving packets were dropped. In this research, the total service rate ($A$) was set to 1.6 Mbps, and it was shared among all nodes in the network. The maximum observed saturated throughput in wireless LAN networks is 1.6 Mbps, with a channel capacity of 2 Mbps (for any number of nodes). In this model, each node was provided an equal service rate $\mu$. For $n$ nodes, queue 1 represented the
well-behaving nodes, and queue 2 represented the one misbehaving node. If packets were present in both queues, then a \((n-1)\mu\) service rate was provided to queue 1, and \(\mu\) was the service rate provided to queue 2, where \(\mu\) is defined later in section 4.3.1.

4.3 M/M/1/K Queuing System

A queue based on weighted fair queuing is the service policy in a multiclass system. The 802.11 wireless LAN network was modeled as a WFQ system with a two-class M/M/1/K queue.

The simple M/M/1/K queue may be viewed as a queue comprising packets from a node and a server that acts as a channel. At a given instance in time, each packet either waited for service or was served by the server. Each packet in the M/M/1/K system underwent the following process:

- The packet arrived at the system (the arrival of each packet is separated by random time intervals called *inter-arrival times*).
- The packet entered the queue.
- If the server was idle, then the packet was served; if not, then the packet waited in the queue until the server was free to serve.
- Each packet was served for a random interval of time called service time.
- The packet departed after it had been served completely by the server.
In a simple M/M/1/K queuing system, the arrival and departure processes follow a Poisson distribution. In other words, the arrival of a job and the departure of a job is Markovian in nature since both events occur randomly and are independent of past arrivals or departures. The arrival rate and service rates are denoted in packets/sec. In this M/M/1/K system, the queue capacity was limited to 50 packets.

The steady-state probability equations for the M/M/1/K queuing system were formulated and solved using MATLAB. The required data, such as arrival rates, service rates, and queue length, were entered manually. The steady-state probability values were determined and used to calculate throughput of the nodes.

4.3.1 Parameters

The arrival rate used for simulations in MATLAB was calculated as

\[
\text{Arrival Rate} (\lambda_i) = \frac{\text{Arrival Rate (bits/sec)} \times n_i \text{ (packets/sec)}}{\text{Packet Size (bits)}}
\]  \hspace{1cm} (3)

where \( n_i \) is the number of nodes represented by each queue \( i \). \( i \in \{1,2\} \)

According to Bianchi [8], the maximum attainable service rate is 80% of the channel capacity. The channel capacity used in this research was 2 Mbps, so the maximum attainable service rate (\( A \)) was set to 1.6 Mbps. The service rate (\( A \)) of 1.6 Mbps was shared among all nodes in the network. The service rate was calculated as follows:

If packets are only in queue 1, then the service rate for queue 1, denoted by \( \mu_{q1} \), is given by

\[
\text{Service Rate} (\mu_{q1}) = \frac{A}{\text{packet size (bits) from queue 1}} \text{ (packets/sec)}
\]  \hspace{1cm} (4)

If packets are only in queue 2, then the service rate for queue 2, denoted by \( \mu_{q2} \), is given by

\[
\text{Service Rate} (\mu_{q2}) = \frac{A}{\text{packet size (bits) from queue 2}} \text{ (packets/sec)}
\]  \hspace{1cm} (5)

If packets are in both queues, then the service rate for each node is given by
Service Rate ($\mu$) = \( \frac{A}{(n-1)\text{packet size(bits) from queue 1} + \text{packet size(bits) from queue 2}} \text{ (pkts/sec)} \) (6)

The service rate for queue 1 was $(n-1)\mu$, and the service rate for queue 2 was $\mu$.

Let $n$ denote the number of nodes in the network. For example, $n = 4$ (three well-behaved nodes sending data with packet sizes of 100 bits, and one misbehaving node sending data with a packet size of 200 bits. Let $A = 1,000$ bits/sec. Then,

\[
\mu_{q1} = \frac{A}{100} = 10 \text{ pkts/sec}
\]

\[
\mu_{q2} = \frac{A}{200} = 5 \text{ pkts/sec}
\]

\[
\mu = \frac{A}{3 \times 100 + 200} = 2 \text{ pkts/sec}
\]

Since, the attempt here was to model the network where each node would receive equal service in terms of packets/sec, irrespective of packet size used, each node in this example received a service rate of $\mu = 2$ pkts/sec. Note that $n \mu \neq A$, since $A$ is expressed in bits/sec, and $\mu$ is expressed in pkts/sec. If the packet size used by each node was the same, then $n \mu \times$ (packet size) = $A$.

4.3.2 State Transition Diagram

Figure 10 shows the state transition diagram for this WFQ system, where each state denotes the number of customers in the system. The generalized two-dimensional Markov chain is described as two states ($x_1$, $x_2$), where $x_1$ represents the packets from queue 1, and $x_2$ represents the packets from queue 2. The transitions from one state to another are given by the arrival rates $\lambda_1$, $\lambda_2$ and the service rates $\mu_{q1}$, $\mu_{q2}$, and $\mu$. 
Figure 10. State transition diagram

The state transition process can be described as follows:

1. If packets are only in queue 1 (implies no packets from misbehaving node), then the service rate would be $\mu_{q1}$ for queue 1. This means that the server serves queue 1 completely and does not provide any service to queue 2 since there is no packet in it.

2. If packets are only in queue 2 (only packets from misbehaving node in the network), then the service rate would be $\mu_{q2}$ for queue 2. This means that the server serves queue 2 completely and does not provide any service to queue 1 since there is no packet in it.

3. If packets are in both queues (packets from both well-behaving and misbehaving nodes are present in the network), then the service rate would be $\mu$ for each node in the network. If there are $n$ nodes in the network, one node represents the misbehaving node, and the rest of the nodes are classified as well-behaving nodes. Therefore, queue 1 representing packets from $(n-1)$ well-behaved nodes, is offered a service of $(n-1) \mu$, and the queue 2 representing the packet from the single misbehaving node is offered a service of $\mu$. The arrival rates, departure rates, and service rates are in packets/sec.
4.3.3 Steady-State Equations

The steady-state probability equations for each state shown in Figure 10 are given below:

**Case 1:** \( i \neq 0, j = 0, i < K \)
\[
\pi(i,j) = \lambda_1 \pi(i-1,j) + \mu q_1 \pi(i+1,j) + \mu \pi(i,j+1)/(\lambda_1 + \lambda_2 + \mu q_1) \tag{7}
\]

**Case 2:** \( i = K, j = 0 \)
\[
\pi(i,j) = \lambda_1 \pi(i-1,j) / \mu q_1 \tag{8}
\]

**Case 3:** \( i = 0, j \neq 0, j < K \)
\[
\pi(i,j) = \lambda_2 \pi(i,j-1) + (n-1)\mu \pi(i+1,j) + \mu q_2 \pi(i,j+1)/(\lambda_1 + \lambda_2 + \mu q_2) \tag{9}
\]

**Case 4:** \( i = 0; j = K \)
\[
\pi(i,j) = \lambda_2 \pi(i,j-1) / \mu q_2 \tag{10}
\]

**Case 5:** \( i \neq 0, j \neq 0, (i+j) < K \)
\[
\pi(i,j) = \lambda_1 \pi(i-1,j) + \lambda_2 \pi(i,j-1) + (n-1)\mu \pi(i+1,j) + \mu \pi(i,j+1)/(\lambda_1 + \lambda_2 + n\mu) \tag{11}
\]

**Case 6:** \( i \neq 0, j \neq 0, (i+j) = K \)
\[
\pi(i,j) = \lambda_1 \pi(i-1,j) + \lambda_2 \pi(i,j-1)/n\mu \tag{12}
\]

**Case 7:** \( \Sigma_{i,j} \pi(i,j) = 1 \)
\[
\tag{13}
\]

By solving equations (7) to (13), \( \pi(0,0) \) can be determined. By solving these steady-state probability equations in MATLAB, the throughput of nodes can be calculated as follows:

Let B denote the set of states of the form \((0,j)\), where queue 1 has no customers, i.e,
\[
B = \{(0,0),(0,1),(0,2),\ldots,(0,K)\} \tag{14}
\]

Let C correspond to the probability that the server is idle for queue 1, i.e.,
\[
C = \Sigma_{i \in B} \pi_i \tag{15}
\]

Let D denote the set of states of the form \((i,0)\), where queue 2 has no customers, i.e,
\[
D = \{(1,0),(2,0),\ldots,(K,0)\} \tag{16}
\]
Let $E$ correspond to the probability that the server is idle for queue 2 if there are packets in queue 1, i.e.,

$$E = \sum_{i \in D} \pi_i$$ (17)

$$\text{Thr}_1 = [E \cdot \mu_{q1} + (1-C-E) \cdot \mu \cdot (n-1)] \cdot \text{pkt size used by queue 1 nodes}$$ (18)

$\text{Thr}_1$ is the throughput of queue 1 in bits/sec. Throughput of each node in queue 1 = $\text{Thr}_1/(n-1)$.

Similarly, the throughput of queue 2 is calculated as follows:

Let $W$ denote set of states of the form $(j,0)$ where queue 2 has no customers, i.e,

$$W = \{(0,0),(1,0),(2,0),\ldots,(K,0)\}$$ (19)

Let $X$ corresponds to server idle probability for queue 2, i.e.,

$$X = \sum_{i \in W} \pi_i$$ (20)

Let $Y$ denote set of states of the form $(0,j)$ where queue 1 has no customers, i.e,

$$Y = \{(0,1),(0,2),\ldots,(0,K)\}$$ (21)

Let $Z$ correspond to the probability that the server is idle for queue 1 if there are packets in queue 2, i.e.,

$$Z = \sum_{i \in Y} \pi_i$$ (22)

$$\text{Thr}_2 = [Z \cdot \mu_{q2} + (1-X-Z) \cdot \mu] \cdot \text{pkt size used by queue 2 nodes}$$ (23)

Thus, throughput of the misbehaving node equals $\text{Thr}_2$.

### 4.4 Effectiveness of Packet Size-Based Misbehavior

This chapter explains that three nodes were used to explain the performance of the M/M/1/K queue under the WFQ discipline with two queues, and then the same setup was extended to a nine-node scenario.

#### 4.4.1 Three-Node Scenario

Of the three nodes, two acted as senders and one acted as a receiver. Of the two sender nodes, one acted as a misbehaving node. Each sender represented one queue. Queue 1
represented packets from the well-behaving node, and queue 2 represented packets from the misbehaving node. Both nodes sent traffic at the rate of 2 Mbps, and the service rate was 2 Mbps. The arrival and service rates of the nodes for the different set of packet sizes were computed as follows:

**Case 1: Well-Behaving Node’s Packet Size Set to 512 Bytes**

The arrival rate for queue 1 is given as

\[
\lambda_1 = \frac{2 \times 10^6}{512*8} = 488.25 \text{ packets/sec}
\]

The service rate for queue 1, if packets are present only in queue 1, is given as

\[
\mu_{q1} = \frac{1.6 \times 10^6}{512*8} = 390.6 \text{ packets/sec}
\]

Queue 2 (misbehaving-node traffic) deals with packets of variable sizes (256–2,304 bytes). Table 1 shows the rate for queue 2 \((\lambda_2)\), service rate \((\mu_{q2})\) if packets are present only in queue 2, and service rate \((\mu)\) for each node if the packets are present in both queues.

**TABLE 1**

<table>
<thead>
<tr>
<th>Sample No.</th>
<th>(P_{\text{var}}) (bytes)</th>
<th>(\lambda_2) (packets/sec)</th>
<th>(\mu_{q2}) (packets/sec)</th>
<th>(\mu) (packets/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
<td>976</td>
<td>781.25</td>
<td>260</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
<td>488</td>
<td>390.625</td>
<td>195</td>
</tr>
<tr>
<td>3</td>
<td>768</td>
<td>325</td>
<td>260.4</td>
<td>156</td>
</tr>
<tr>
<td>4</td>
<td>1,024</td>
<td>244</td>
<td>195.31</td>
<td>130</td>
</tr>
<tr>
<td>5</td>
<td>1,280</td>
<td>195</td>
<td>156.25</td>
<td>111.6</td>
</tr>
<tr>
<td>6</td>
<td>1,536</td>
<td>162</td>
<td>130.20</td>
<td>97.6</td>
</tr>
<tr>
<td>7</td>
<td>1,792</td>
<td>139</td>
<td>111.60</td>
<td>86.8</td>
</tr>
<tr>
<td>8</td>
<td>2,048</td>
<td>122</td>
<td>97.65</td>
<td>78.1</td>
</tr>
<tr>
<td>9</td>
<td>2,304</td>
<td>108</td>
<td>86.80</td>
<td>71</td>
</tr>
</tbody>
</table>

Figure 11 shows each node’s throughput and total throughput, when the well-behaving node’s packet size was set to 512 bytes. It is clear that as the misbehaving node’s packet size
increased beyond 512 bytes, its throughput increased and that of the well-behaving node decreased.

![Graph](image)

Figure 11. Three nodes with well-behaving node’s packet size set to 512 bytes

**Case 2: Well-Behaving Node’s Packet Size Set to 1,024 Bytes**

The arrival rate for queue 1 is given as

$$\lambda_1 = \frac{2 \times 10^6}{1,024 \times 8} = 244.10 \text{ packets/sec}$$

The service rate for queue 1, if packets are present only in queue 1, is given as

$$\mu_{q1} = \frac{1.6 \times 10^6}{1,024 \times 8} = 195.3 \text{ packets/sec}$$

Queue 2 (misbehaving-node traffic) deals with packets of variable sizes (256–2,304 bytes). The corresponding arrival rate for queue 2, service rate ($\mu_{q2}$) if packets are present only in queue 2, and service rate ($\mu$) for each node if packets are present in both the queues are shown in Table 2.
TABLE 2
THREE NODES WITH WELL-BEHAVING NODE’S PACKET SIZE SET TO 1,024 BYTES

<table>
<thead>
<tr>
<th>Sample No.</th>
<th>$P_{var}$ (bytes)</th>
<th>$\lambda$ (packets/sec)</th>
<th>$\mu_q$ (packets/sec)</th>
<th>$\mu$ (packets/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
<td>976</td>
<td>781.25</td>
<td>156</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
<td>488</td>
<td>390.625</td>
<td>130</td>
</tr>
<tr>
<td>3</td>
<td>768</td>
<td>325</td>
<td>260.4</td>
<td>111.6</td>
</tr>
<tr>
<td>4</td>
<td>1,024</td>
<td>244</td>
<td>195.31</td>
<td>97.6</td>
</tr>
<tr>
<td>5</td>
<td>1,280</td>
<td>195</td>
<td>156.25</td>
<td>86.8</td>
</tr>
<tr>
<td>6</td>
<td>1,536</td>
<td>162</td>
<td>130.20</td>
<td>78.1</td>
</tr>
<tr>
<td>7</td>
<td>1,792</td>
<td>139</td>
<td>111.60</td>
<td>71.0</td>
</tr>
<tr>
<td>8</td>
<td>2,048</td>
<td>122</td>
<td>97.65</td>
<td>65.1</td>
</tr>
<tr>
<td>9</td>
<td>2,304</td>
<td>108</td>
<td>86.80</td>
<td>60.0</td>
</tr>
</tbody>
</table>

Figure 12 shows the throughput of the nodes when the well-behaving nodes packet size was set to 1,024 bytes.

Figure 12. Three nodes with well-behaving node’s packet size set to 1,024 bytes
**Case 3: Well-Behaving Node’s Packet Size Set to 2,048 Bytes**

The arrival rate for queue 1 is given as

\[
\lambda_1 = \frac{2 \times 10^6}{2,048} = 122.07 \text{ packets/sec}
\]

The service rate for queue 1, if packets are present only in queue 1, is given as

\[
\mu_{q1} = \frac{1.6 \times 10^6}{2,048} = 97.6 \text{ packets/sec}
\]

Queue 2 (misbehaving-node traffic) deals with packets of variable sizes (256–2304 bytes). The corresponding arrival rate for queue 2, service rate (\(\mu_{q2}\)) if packets are present only in queue 2, and service rate (\(\mu\)) for each node if packets are present in both queues are shown in Table 3.

**TABLE 3**

<table>
<thead>
<tr>
<th>Sample No.</th>
<th>(P_{\text{var}}) (bytes)</th>
<th>(\lambda_2) (packets/sec)</th>
<th>(\mu_{q2}) (packets/sec)</th>
<th>(\mu) (packets/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
<td>976</td>
<td>781.25</td>
<td>86.8</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
<td>488</td>
<td>390.625</td>
<td>78.1</td>
</tr>
<tr>
<td>3</td>
<td>768</td>
<td>325</td>
<td>260.4</td>
<td>71</td>
</tr>
<tr>
<td>4</td>
<td>1,024</td>
<td>244</td>
<td>195.31</td>
<td>65.1</td>
</tr>
<tr>
<td>5</td>
<td>1280</td>
<td>195</td>
<td>156.25</td>
<td>60.0</td>
</tr>
<tr>
<td>6</td>
<td>1536</td>
<td>162</td>
<td>130.20</td>
<td>55.8</td>
</tr>
<tr>
<td>7</td>
<td>1792</td>
<td>139</td>
<td>111.60</td>
<td>52.08</td>
</tr>
<tr>
<td>8</td>
<td>2,048</td>
<td>122</td>
<td>97.65</td>
<td>48.8</td>
</tr>
<tr>
<td>9</td>
<td>2304</td>
<td>108</td>
<td>86.80</td>
<td>46</td>
</tr>
</tbody>
</table>

Figure 13 shows the throughput of the nodes when the well-behaving node’s packet size was set to 2,048 bytes.
Figure 13. Three nodes with well-behaving node’s packet size set to 2,048 bytes

4.4.2 Nine-Node Scenario

The weighted fair queuing system was extended to a nine-node wireless LAN environment. In the scenario involving three nodes, each queue represented one sender, whereas in this scenario involving nine nodes, queue 1 represented seven well-behaving senders conveying traffic at the rate of 0.5 Mbps, and queue 2 represented one misbehaving sender conveying traffic at the rate of 0.5 Mbps. The service rate was 2 Mbps. The arrival rate and service rates for each case was calculated and throughput was plotted.

Case 1: Well-Behaving Nodes’ Packet Size Set to 512 bytes:

The arrival rate for queue 1 is given as

\[
\lambda_1 = \frac{0.5 \times 7 \times 10^6}{512 \times 8} = 855 \text{ packets/sec}
\]

The service rate for queue 1, if packets are present only in queue 1, is given as

\[
\mu_{q1} = \frac{1.6 \times 10^6}{512 \times 8} = 390.625 \text{ packets/sec}
\]
Queue 2 (misbehaving-node traffic) deals with packets of variable sizes (256–2,304 bytes). Table 4 shows the corresponding arrival rate for queue 2, service rate ($\mu_{q2}$) if packets are present only in queue 2, and service rate ($\mu$) for each node if packets are present in both queues.

### TABLE 4

**NINE NODES WITH WELL-BEHAVING NODES’ PACKET SIZE SET TO 512 BYTES**

<table>
<thead>
<tr>
<th>Sample No.</th>
<th>$P_{\text{var}}$ (bytes)</th>
<th>$\lambda_2$ (packets/sec)</th>
<th>$\mu_{q2}$ (packets/sec)</th>
<th>$\mu$ (packets/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
<td>244</td>
<td>781.25</td>
<td>52.1</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
<td>122</td>
<td>390.625</td>
<td>48.8</td>
</tr>
<tr>
<td>3</td>
<td>768</td>
<td>81.3</td>
<td>260.41</td>
<td>46</td>
</tr>
<tr>
<td>4</td>
<td>1,024</td>
<td>61</td>
<td>195.31</td>
<td>43.4</td>
</tr>
<tr>
<td>5</td>
<td>1,280</td>
<td>49</td>
<td>156.25</td>
<td>41.1</td>
</tr>
<tr>
<td>6</td>
<td>1,536</td>
<td>40</td>
<td>130.21</td>
<td>39</td>
</tr>
<tr>
<td>7</td>
<td>1,792</td>
<td>35</td>
<td>111.60</td>
<td>37.2</td>
</tr>
<tr>
<td>8</td>
<td>2,048</td>
<td>30.5</td>
<td>97.656</td>
<td>35.5</td>
</tr>
<tr>
<td>9</td>
<td>2,304</td>
<td>27</td>
<td>86.80</td>
<td>34</td>
</tr>
</tbody>
</table>

The calculated service rates and arrival rates were used to solve the steady-state probability equations in MATLAB. Throughput of the well-behaving nodes, misbehaving node, and the total network were determined and plotted in Figure 14. In Figure 14, “Other (well)” represents the average throughput of well-behaved nodes, “misb” is throughput of the misbehaving node, and “total” is total throughput of all n nodes in the network.
Figure 14. Nine nodes with well-behaving nodes’ packet size set to 512 bytes

Case 2: Well-Behaving Nodes’ Packet Size Set to 1,024 Bytes

The arrival rate for queue 1 is given as

\[ \lambda_1 = \frac{0.5 \times 10^6}{1,024 \times 8} = 427 \text{ packets/sec} \]

The service rate for queue 1, if it receives packets only from queue 1, is given as

\[ \mu_{q1} = \frac{1.6 \times 10^6}{1,024 \times 8} = 195.3 \text{ packets/sec} \]

Queue 2 (misbehaving-node traffic) deals with packets of variable sizes (256–2,304 bytes). The corresponding arrival rate for queue 2, service rate (\( \mu_{q2} \)) if packets are present only in queue 2, and service rate (\( \mu \)) for each node if packets are present in both queues are shown in Table 5.
### TABLE 5

**NINE NODES WITH WELL-BEHAVING NODES’ PACKET SIZE SET TO 1,024 BYTES**

<table>
<thead>
<tr>
<th>Sample No.</th>
<th>$P_{var}$ (bytes)</th>
<th>$\lambda_2$ (packets/sec)</th>
<th>$\mu_{q2}$ (packets/sec)</th>
<th>$\mu$ (packets/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
<td>244</td>
<td>781.25</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
<td>122</td>
<td>390.625</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>768</td>
<td>81.3</td>
<td>260.41</td>
<td>25.2</td>
</tr>
<tr>
<td>4</td>
<td>1,024</td>
<td>61</td>
<td>195.31</td>
<td>24.4</td>
</tr>
<tr>
<td>5</td>
<td>1,280</td>
<td>49</td>
<td>156.25</td>
<td>23.6</td>
</tr>
<tr>
<td>6</td>
<td>1,536</td>
<td>40</td>
<td>130.21</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td>1,792</td>
<td>35</td>
<td>111.60</td>
<td>22.3</td>
</tr>
<tr>
<td>8</td>
<td>2,048</td>
<td>30.5</td>
<td>97.656</td>
<td>21.7</td>
</tr>
<tr>
<td>9</td>
<td>2,304</td>
<td>27</td>
<td>86.80</td>
<td>21.1</td>
</tr>
</tbody>
</table>

Figure 15 shows the throughput of well-behaving and misbehaving nodes, and total throughput of the network.

Figure 15. Nine nodes with well-behaving nodes’ packet size set to 1,024 bytes

**Case 3: Well-Behaving Nodes’ Packet Size Set to 2,048 Bytes**

The arrival rate for queue 1 is given as

$$\text{Arrival Rate} (\lambda_1) = \frac{0.5 \times 7 \times 10^6}{2,048 + 8} = 213.5 \text{ packets/sec}$$
The service rate for queue 1 if it receives packets only from queue 1 is given as

\[
\text{Service Rate (}\mu_{q1}\text{)} = \frac{1.6 \times 10^6}{2048} = 97.65 \text{ packets/sec}
\]

Queue 2 (misbehaving-node traffic) deals with packets of variable sizes (256–2,304 bytes). The corresponding arrival rate for queue 2, service rate (\(\mu_{q2}\)) if packets are present only in queue 2, and service rate (\(\mu\)) for each node if packets are present in both queues are shown in Table 6.

**TABLE 6**

NINE NODES WITH WELL-BEHAVING NODES’ PACKET SIZE SET TO 2,048 BYTES

<table>
<thead>
<tr>
<th>Sample No.</th>
<th>(P_{var}) (bytes)</th>
<th>(\lambda_2) (packets/sec)</th>
<th>(\mu_{q2}) (packets/sec)</th>
<th>(\mu) (packets/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
<td>244</td>
<td>781.25</td>
<td>13.7</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
<td>122</td>
<td>390.625</td>
<td>13.46</td>
</tr>
<tr>
<td>3</td>
<td>768</td>
<td>81.3</td>
<td>260.41</td>
<td>13.24</td>
</tr>
<tr>
<td>4</td>
<td>1,024</td>
<td>61</td>
<td>195.31</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>1,280</td>
<td>49</td>
<td>156.25</td>
<td>12.8</td>
</tr>
<tr>
<td>6</td>
<td>1,536</td>
<td>40</td>
<td>130.21</td>
<td>12.6</td>
</tr>
<tr>
<td>7</td>
<td>1,792</td>
<td>35</td>
<td>111.60</td>
<td>12.4</td>
</tr>
<tr>
<td>8</td>
<td>2,048</td>
<td>30.5</td>
<td>97.656</td>
<td>12.2</td>
</tr>
<tr>
<td>9</td>
<td>2,304</td>
<td>27</td>
<td>86.80</td>
<td>12</td>
</tr>
</tbody>
</table>

Figure 16 shows the throughput of well-behaving nodes, the misbehaving node, and total throughput of the network.

Figure 16. Nine nodes with well-behaving nodes’ packet size set to 2,048 bytes
The throughput results obtained by solving the M/M/1/K queuing system in MATLAB are closer to the practical results shown in Chapter 3. Thus, the WFQ model used to represent the network seems to be accurate enough.

4.5 Model to Reduce Impact of Misbehavior

Section 4.4 showed the effect of misbehavior achieved by altering the packet size. In the previous model, the server served each node with an equal service rate in terms of pkts/sec. In order to reduce the effect of misbehavior, a WFQ system was modeled, where the server served each queue with different service rates based on their packet sizes. Weights $w_1$ and $w_2$ are unknowns, which must be determined in order to reduce the effect of misbehavior. Computation of the weights is shown in section 4.5.1. Therefore, queue 1, representing packets from the well-behaving node was offered a service of $w_1(n-1)\mu$ instead of $(n-1)\mu$, which was the case in section 4.4. Queue 2, representing a single misbehaving node, was offered a service of $w_2\mu$ instead of $\mu$. The M/M/1/K model is represented in Figure 17. In order to reduce the impact of misbehavior, it is desirable that $w_1 > 1$ and $w_2 < 1$.

![Figure 17. Queuing system with varied service rates](image)

4.5.1 Parameters

In order to solve the above steady-state equations, the parameters $\lambda_1$, $\lambda_2$, $\mu_{q1}$, $\mu_{q2}$, $\mu$, $w_1$, and $w_2$ must be determined. Arrival rates $\lambda_1$, $\lambda_2$ were calculated using the formulas shown in section
4.4, as were service rates \( \mu_{q1}, \mu_{q2}, \mu \). The channel capacity used in this research was 2 Mbps. According to Bianchi [8], the maximum attainable service rate is 80% of the channel capacity used. Hence, the total service rate \( A \) would be 1.6 Mbps. Therefore, in this research, the service rate \( A \) of 1.6 Mbps was shared among all nodes in the network. In order to provide varied service rates for well-behaving and misbehaving nodes, the weights \( w_1 \) and \( w_2 \) were computed as follows:

\[
\begin{align*}
\mu_1 &= \mu \cdot \text{Pfixed} \times 8 \\
\mu_2 &= \mu \cdot \text{Pvar} \times 8 \\
(n-1) \frac{\mu_1}{w_1} + \frac{\mu_2}{w_2} &= A
\end{align*}
\]

From section 4.3.1,

\[
\mu = A / \left[ (n-1) \text{Pfixed} + \text{Pvar} \right] \times 8
\]

which implies that \( A = [(n-1) \text{Pfixed} + \text{Pvar}] \times 8 \mu \)

Therefore, \( (n-1)w_1 \mu_1 + w_2 \mu_2 = A \) implies that

\[
(n-1)w_1 \mu_1 + w_2 \mu_2 = [(n-1) \text{Pfixed} + \text{Pvar}] \times 8 \mu
\]

Now, substituting equations (25) and (26) into the above equation yields the following:

\[
(n-1) \frac{\mu_1}{w_1} \cdot \text{Pfixed} \times 8 + \frac{\mu_2}{w_2} \cdot \text{Pvar} \times 8 = [(n-1) \text{Pfixed} + \text{Pvar}] \times 8 \mu
\]

\[
=> (n-1)w_1 \cdot \text{Pfixed} + w_2 \cdot \text{Pvar} = (n-1) \text{Pfixed} + \text{Pvar}
\]

\[
=> (w_1-1)(n-1) \text{Pfixed} = (1 - w_2) \text{Pvar}
\]

\[
=> \left( \frac{\text{Pvar}}{\text{Pfixed}} \right) w_2-1)(n-1) \text{Pfixed} = (1 - w_2) \text{Pvar}
\]

\[
=> (\text{Pvar} \frac{w_2}{\text{Pfixed}} - \text{Pfixed})(n - 1) = (1 - w_2) \text{Pvar}
\]

\[
=> (w_2 \frac{\text{Pfixed}}{\text{Pvar}}) (n-1) = (1 - w_2)
\]

The above equation was used to solve for \( w_2 \).
Then, \( w_1 = w_2^{\text{Pvar}} \frac{\text{Pfixed}}{\text{Pvar}} \)

If \( \text{Pfixed} = \text{Pvar} \), then \( \text{eq}^n \Rightarrow w_2 = 1 \) and, hence, \( w_1 = 1 \) (since \( n > 1 \))

If \( \text{Pfixed} < \text{Pvar} \), then \( \text{eq}^n \Rightarrow \frac{\text{Pfixed}}{\text{Pvar}} < w_2 < 1 \) and \( w_1 > 1 \).

As mentioned in section 4.5, this is needed

where

\( n = \) number of nodes in network

\( (n-1) = \) number of well-behaving nodes in network

\( \text{Pfixed} = \) fixed-packet size used by well-behaving nodes in network

\( \text{Pvar} = \) variable-packet size used by misbehaving nodes in network

\( \mu_1, \mu_2 = \) service rates for queue 1 and queue 2, respectively, in bits/sec

\( \mu = \) service rate for each node if packets are present in both queues and unit is packets/sec

Solving equations (1) and (4) gave the weights \( w_1 \) and \( w_2 \), which were used to compute the service rate for both queues. If packets were present in both queues, then \( w_1 (n-1)\mu \) would be the service rate for queue 1, and \( w_2 \mu \) would be the service rate for queue 2.

4.5.2 State Transition Diagram

Figure 18 depicts the state transition diagram for the model and provides different service rates for each queue. As mentioned in section 4.4, the state of the system \( (x_1, x_2) \) is defined as (packets from queue 1, packets from queue 2), and the transitions are given by the arrival rates \( \lambda_1, \lambda_2 \) and the service rates \( \mu_{q1}, \mu_{q2}, \mu \) and \( w_1 * (n-1)\mu, w_2 * \mu \).
4.5.3 Steady-State equations

The steady-state equations with varied service rates are shown below:

**Case 1:** $i \neq 0, j = 0, i < K$

$$\pi(i,j) = \lambda_1 \pi(i-1,j) + \mu q_1 \pi(i+1,j) + w_2 \mu \pi(i,j + 1)/\lambda_1 + \lambda_2 + \mu q_1$$  \hspace{1cm} (28)

**Case 2:** $i = K, j = 0$

$$\pi(i,j) = \lambda_1 \pi(i-1,j)/\mu q_1$$  \hspace{1cm} (29)

**Case 3:** $i = 0, j \neq 0, j < K$

$$\pi(i,j) = \lambda_2 \pi(i,j-1) + (n-1)w_1 \mu \pi(i+1,j) + \mu q_2 \pi(i,j + 1)/\lambda_1 + \lambda_2 + \mu q_2$$  \hspace{1cm} (30)

**Case 4:** $i = 0, j = K$

$$\pi(i,j) = \lambda_2 \pi(i,j-1)/\mu q_2$$  \hspace{1cm} (31)

**Case 5:** $i \neq 0, j \neq 0, (i+j) < K$

$$\pi(i,j) = \lambda_1 \pi(i-1,j) + \lambda_2 \pi(i,j-1) + (n-1)w_1 \mu \pi(i+1,j) + w_2 \mu \pi(i,j + 1)/\lambda_1 + \lambda_2 + (n-1)w_1 \mu + w_2 \mu$$  \hspace{1cm} (32)
**Case 6:** \( i \neq 0, j \neq 0, (i+j) = K \)

\[
\pi(i,j) = \lambda_1 \pi(i-1,j) + \lambda_2 \pi(i,j-1)/(n-1)w_1 \mu + w_2 \mu
\]  

(33)

**Case 7:** \( \sum_{i,j} \pi(i,j) = 1 \)

(34)

By solving equations (28) to (34), \( \pi(0,0) \) can be determined. By solving these steady-state equations in MATLAB, the throughput of both well-behaving and misbehaving nodes can be calculated as follows:

Let B denote the set of states of the form \((0,j)\), where queue 1 has no customers, i.e,

\[
B = \{(0,0), (0,1), \ldots, (0,K)\}
\]

(35)

Let C correspond to the probability that the server is idle for queue 1, i.e.,

\[
C = \sum_{i \in B} \pi_i
\]

(36)

Let D denote the set of states of the form \((i,0)\), where queue 2 has no customers, i.e.,

\[
D = \{(1,0), (2,0), \ldots, (K,0)\}
\]

(37)

Let E correspond to the probability that the server is idle for queue 2, i.e.,

\[
E = \sum_{i \in D} \pi_i
\]

(38)

\[
\text{Thr}_1 = [E \cdot \mu_{q1} + (1-C-E) \cdot \mu \cdot w_1 \cdot (n-1)] \cdot \text{pkt size used by queue 1 nodes}
\]

(39)

\( \text{Thr}_1 \) is the throughput of queue 1 in bits/sec. The throughput of each node in queue 1 = \( \text{Thr}_1/(n-1) \). Similarly, the throughput of queue 2 is

\[
\text{Thr}_2 = [Z \cdot \mu_{q2} + (1-X-Z) \cdot \mu \cdot w_2] \cdot \text{pkt size used by queue 2 nodes}
\]

(40)

\( X \) and \( Z \) were calculated previously in section 4.4.

**4.6 Impact of Using Varied Service Rates**

This section discusses the impact on throughput achieved by each node when using variable service rates. The service rate was varied to reduce the effect of misbehavior and to
bring fairness into the network. The scenarios illustrated in section 4.4.2 were repeated to show the impact of varied service rates.

**Case 1: Well-Behaving Nodes’ Packet Size Set to 512 Bytes**

The arrival rates, $\lambda_1$ and $\lambda_2$, and service rates, $\mu_{q1}$, $\mu_{q1}$, were the same values used earlier. The only variation here is that the service rate for the nodes when the packets are present in both the queues was different. In the case of both queues having packets, the weights $w_1$ and $w_2$ were determined to compute the service rates.

Using the above formulas, Table 7 shows the computed weights and corresponding service rates. Using the service rates shown in Table 7, the throughput attained by the nodes is plotted in Figure 19. It can be observed that throughput of the misbehaving node was the same as the average throughput achieved by the well-behaving nodes, suggesting that this varied service-rate scheme could serve as an appropriate reaction strategy to reduce the effectiveness of packet size-based misbehavior.

**TABLE 7**

WELL-BEHAVING NODES’ PACKET SIZE SET TO 512 BYTES WITH VARIED SERVICE RATES

<table>
<thead>
<tr>
<th>Sample No.</th>
<th>$P_{var}$ (bytes)</th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_1\mu$ (packets/sec)</th>
<th>$w_2\mu$ (packets/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
<td>0.9372</td>
<td>1.87</td>
<td>48.823</td>
<td>97.656</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
<td>1.00</td>
<td>1.00</td>
<td>48.823</td>
<td>48.82</td>
</tr>
<tr>
<td>3</td>
<td>768</td>
<td>1.061</td>
<td>0.707</td>
<td>48.823</td>
<td>32.55</td>
</tr>
<tr>
<td>4</td>
<td>1,024</td>
<td>1.125</td>
<td>0.562</td>
<td>48.823</td>
<td>24.41</td>
</tr>
<tr>
<td>5</td>
<td>1,280</td>
<td>1.188</td>
<td>0.475</td>
<td>48.823</td>
<td>19.53</td>
</tr>
<tr>
<td>6</td>
<td>1,536</td>
<td>1.25</td>
<td>0.417</td>
<td>48.823</td>
<td>16.27</td>
</tr>
<tr>
<td>7</td>
<td>1,792</td>
<td>1.312</td>
<td>0.375</td>
<td>48.823</td>
<td>13.95</td>
</tr>
<tr>
<td>8</td>
<td>2,048</td>
<td>1.375</td>
<td>0.343</td>
<td>48.823</td>
<td>12.20</td>
</tr>
<tr>
<td>9</td>
<td>2,304</td>
<td>1.436</td>
<td>0.319</td>
<td>48.823</td>
<td>10.85</td>
</tr>
</tbody>
</table>
Figure 19: Well-behaving nodes’ packet size set to 512 bytes with varied service rate

**Case 2: Well-Behaving Nodes’ Packet Size Set to 1,024 Bytes**

Table 8 shows the weights computed for both queues when the well-behaving nodes’ packet size was set to 1,024 bytes. Figure 20 shows the throughput achieved by nodes using the service rates mentioned in Table 8, which is a similar impact to that shown in Figure 19.

**TABLE 8**

WELL-BEHAVING NODES’ PACKET SIZE SET TO 1,024 BYTES WITH VARIED SERVICE RATES

<table>
<thead>
<tr>
<th>Sample No.</th>
<th>$P_{var}$ (bytes)</th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_1\mu$ (packets/sec)</th>
<th>$w_2\mu$ (packets/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
<td>0.904</td>
<td>3.616</td>
<td>24.414</td>
<td>97.656</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
<td>0.939</td>
<td>1.87</td>
<td>24.414</td>
<td>48.828</td>
</tr>
<tr>
<td>3</td>
<td>768</td>
<td>0.9688</td>
<td>1.291</td>
<td>24.414</td>
<td>32.55</td>
</tr>
<tr>
<td>4</td>
<td>1,024</td>
<td>1.00</td>
<td>1.00</td>
<td>24.414</td>
<td>24.4</td>
</tr>
<tr>
<td>5</td>
<td>1280</td>
<td>1.0344</td>
<td>0.827</td>
<td>24.414</td>
<td>19.53</td>
</tr>
<tr>
<td>6</td>
<td>1536</td>
<td>1.061</td>
<td>0.707</td>
<td>24.414</td>
<td>16.27</td>
</tr>
<tr>
<td>7</td>
<td>1792</td>
<td>1.094</td>
<td>0.625</td>
<td>24.414</td>
<td>13.95</td>
</tr>
<tr>
<td>8</td>
<td>2,048</td>
<td>1.125</td>
<td>0.562</td>
<td>24.414</td>
<td>12.20</td>
</tr>
<tr>
<td>9</td>
<td>2304</td>
<td>1.157</td>
<td>0.512</td>
<td>24.414</td>
<td>10.87</td>
</tr>
</tbody>
</table>
Case 3: Well-Behaving Nodes’ Packet Size Set to 2,048 Bytes

Table 9 shows the weights computed for both queues when the well-behaving node’s packet size was set to 2,048 bytes. Figure 21 shows the throughput achieved by nodes when using the service rates mentioned in Table 9. Figures 19, 20, and 21 suggest that the “varied service rate” reaction strategy is applicable to different network scenarios as well.

**TABLE 9**

<table>
<thead>
<tr>
<th>Sample No.</th>
<th>$P_{var}$ (bytes)</th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_1\mu$ (packets/sec)</th>
<th>$w_2\mu$ (packets/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
<td>0.891</td>
<td>7.128</td>
<td>12.207</td>
<td>97.656</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
<td>0.906</td>
<td>3.62</td>
<td>12.207</td>
<td>48.82</td>
</tr>
<tr>
<td>3</td>
<td>768</td>
<td>0.921</td>
<td>2.45</td>
<td>12.207</td>
<td>32.55</td>
</tr>
<tr>
<td>4</td>
<td>1,024</td>
<td>0.939</td>
<td>1.87</td>
<td>12.207</td>
<td>24.41</td>
</tr>
<tr>
<td>5</td>
<td>1280</td>
<td>0.953</td>
<td>1.525</td>
<td>12.207</td>
<td>19.531</td>
</tr>
<tr>
<td>6</td>
<td>1536</td>
<td>0.968</td>
<td>1.291</td>
<td>12.207</td>
<td>16.27</td>
</tr>
<tr>
<td>7</td>
<td>1792</td>
<td>0.984</td>
<td>1.125</td>
<td>12.207</td>
<td>13.95</td>
</tr>
<tr>
<td>8</td>
<td>2,048</td>
<td>1.00</td>
<td>1.00</td>
<td>12.207</td>
<td>12.207</td>
</tr>
<tr>
<td>9</td>
<td>2304</td>
<td>1.017</td>
<td>0.211</td>
<td>12.207</td>
<td>10.850</td>
</tr>
</tbody>
</table>
4.7 Summary

This chapter presented the modeling of a 802.11 wireless LAN comprised of well-behaving and misbehaving nodes as an M/M/1/K queuing system where the server followed a round-robin scheduling. Results showed that by increasing the packet size higher than the fixed packet size, the nodes achieved higher throughput. The theoretical results obtained were close to the practical results simulated in NS-2. This model was extended using varied service rates for each queue, in order to reduce the effectiveness of misbehavior. The calculated weights were displayed in tables, and the throughput achieved by the nodes was plotted in figures. From the plots, it can be seen that by varying the service rate for queues, it is possible to reduce the effect of misbehavior and to bring fairness in the throughput of nodes in the network.
CHAPTER 5
DESIGN AND IMPLEMENTATION OF REACTION SCHEME

5.1 Introduction

Previous chapters in this thesis discussed the effect of misbehavior by varying packet size in a wireless LAN environment. This chapter explains how an algorithm was designed to reduce the impact of misbehavior by altering the packet size. Kyasanur and Vaidya [2] introduced the concept of receiver-assigning backoff to reduce the impact of a node’s selfish misbehavior (by altering its backoff value). The same strategy, i.e., a receiver-assigning backoff value, was used in this research as a reaction mechanism for packet size-based misbehavior. Here the receiver assigned a backoff value based on the packet size received from the particular node. The algorithm was motivated by the analytical results obtained in Chapter 4.

Section 5.2 presents the algorithm for reaction against packet size-based misbehavior. Section 5.3 shows how the MAC protocol was implemented in NS-2. Section 5.4 deals with code modifications in NS-2 in order to implement the reaction algorithm. Section 5.5 shows plotted throughput results, which explain how the reaction scheme reduced the effect of misbehavior, and section 5.6 concludes the chapter.

5.2 Algorithm to Reduce Effect of Misbehavior

Chapters 3 and 4 explained how packet size-based misbehavior affects network performance. Chapter 3 shows the implementation of a nine-node wireless network in NS-2, where each node follows the IEEE 802.11 MAC protocol. In that setup, all seven nodes transmitted packets of a fixed packet size, and one node misbehaved by varying its packet size. The simulation results clearly show that the node with the higher packet size acquired greater throughput.
In order to reduce the impact of misbehavior, the standard IEEE 802.11 MAC protocol was modified. In the IEEE 802.11 MAC protocol, the node transmitting the data packet computes or selects the backoff value by itself. In this research, the protocol was modified so that the destination assigned the backoff value for the sender, rather than the sender itself computing its backoff value. This approach is similar to that used by Kyasanur and Vaidya [2].

The algorithm was formulated under the assumption that the receiver is a genuine node and also knows the fixed packet size used by the well-behaving nodes in the network. The algorithm is defined as follows:

1. After the RTS/CTS exchange mechanism the sender transmits the data packet.
2. Upon receipt of the data packet, the receiver examines the size of the data packet and responds based on the following:
   - If the size of the received data packet is greater than the fixed packet size, then the receiver assigns a larger backoff value to the sender.
   - If the size of the received data packet is less than the fixed packet size, then the receiver assigns a smaller backoff value to the sender.
   - If the size of the received data packet equals the fixed packet size, then the receiver assigns a normal backoff value.

The receiver computes the initial backoff with the function \( \text{rand}(0, \text{CW}) \) and modifies the backoff value based on the packet size as follows:

\[
\begin{align*}
\text{If packet size (} P_{\text{size}} \text{) > fixed packet size (} P_{\text{fixed}} \text{), then} \\
\text{actual backoff value} &= \left( \frac{P_{\text{size}}}{P_{\text{fixed}}} \right) \times 2 \times \text{backoff.} \\
\text{If packet size (} P_{\text{size}} \text{) < fixed packet size (} P_{\text{fixed}} \text{), then} \\
\text{actual backoff value} &= \left( \frac{P_{\text{size}}}{P_{\text{fixed}}} \right) \times \text{backoff.} \\
\text{If packet size (} P_{\text{size}} \text{) = fixed packet size (} P_{\text{fixed}} \text{), then} \\
\text{actual backoff} &= \text{backoff.}
\end{align*}
\]
where $P_{\text{size}}$ denotes the packet size of the data packet received from the sender, and actual backoff value denotes the backoff value assigned by the receiver to the sender based on the packet size of the data received from that sender.

3. A new field is introduced into the ACK frame to carry the backoff value that is to be used by the sender in subsequent transmissions.

4. Upon receipt of the ACK frame, the sender must follow the backoff value assigned by the receiver for its subsequent transmission to that particular receiver.

As a result of this algorithm, the sender that is misbehaving by altering its packet size greater than that of the well-behaving nodes received a higher backoff value and waited for a longer period of time to obtain access to the channel. Therefore, other well-behaving nodes received better access to the channel. This ensured fairness among nodes in the network.

Note that this algorithm was motivated by the analytical results obtained in Chapter 4. It was shown that if $P_{\text{var}} > P_{\text{fixed}}$, then a choice of $w_1, w_2$ such that $w_1 > 1$ and $P_{\text{fixed}}/P_{\text{var}} < w_2 < 1$ is able to achieve fairness in throughput of all nodes in the network. In this algorithm, if $P_{\text{var}} > P_{\text{fixed}}$, then a backoff value proportional to $P_{\text{var}}/P_{\text{fixed}}$ was assigned to the misbehaving node to achieve a similar impact.

5.3 Flowchart for Reaction Mechanism:

Figure 22 represents the flowchart of the algorithm to reduce the effect of misbehavior.
Figure 22. Flowchart for reaction mechanism

5.4 IEEE 802.11 MAC Implementation in NS-2

The IEEE 802.11 MAC protocol implemented in NS-2 follows the distributed coordination function for contention resolution (explained in Chapter 1). Simply, DCF works as follows: If a node has a packet to send, it senses the channel, and if it finds the channel to be idle, then it transmits the RTS and the receiver responds back with a CTS. Other nodes hearing the RTS/CTS will defer their transmission for the time allocated in network allocation vector. After the RTS/CTS exchange, the sender transmits the data, and the receiver sends back the ACK.

The following C++ files were used in NS-2 to implement the IEEE 802.11 MAC protocol:
The code has four different functions:

- Transmitting the packet
- Receiving the packet destined for itself
- Overhearing the packet that is not destined for itself
- Sensing packet collision

The appropriate functions were defined in the code to perform the above-mentioned process. Timers such as defer timer and backoff timer were defined in mac-802_11timers.cc and mac-802_11timers.h.

5.5 Code Changes in NS-2

The original mac-802_11.cc and mac-802_11.h files in NS-2 were modified as follows and named mac-802_11mb.cc and mac-802_11mb.h, respectively:

On the receiver side, to compute the backoff value, the following function was added:

```cpp
u_int16_t BackoffSender::getBackoff(u_int32_t dst)
{
    int backoff = ((Random :: random() % CW) + 1);

    Pfixed = 512;

    int actual backoff value;

    if (Psize > Pfixed)
        actual backoff value = (Psize / Pfixed) * backoff * 2
    else
```
actual backoff value = (P_{size} / P_{fixed}) * backoff

return actual backoff value;

After this, the ACK frame was modified to send the computed backoff value to the sender. For this modification the following piece of code was added:

**Mac802_11mb::sendACK(int dst)**

Packet * p = Packet::alloc();
hdr_cmn* ch = HDR_CMN(p);
struct ack_frame *cf = (struct ack_frame*)p->access(hdr_mac::offset_);

   cf->cf_backoff = backoffSender->getBackoff(dst);

   cf->cf_src = index_;

In order to make the sender follow the backoff value assigned by the receiver, a new set of codes was introduced on the sender side. Function check_pktRTS() was used in mac802_11.cc to transmit the RTS packet. The following piece of code was added to the function to force the sender to use the receiver assigned backoff value:

**Mac802_11mb::check_pktRTS()**

int backoff;

if(backoffReceiver)

   backoff= backoffReceiver->getBackoff((u_int32_t)ETHER_ADDR(mh->dh_da));

else

   backoff = (Random::random() % cw_) + 1;

mhBackoff_.start(backoff, is_idle());

The above code works as follows: If sender is sending the data packet for the first time, then it computes the backoff value by itself; otherwise, it follows the backoff value assigned by
the receiver. The backoff receiver is set true/false during node formation in the tcl script. The above code changes were made to file mac-802_11.cc, and the file was renamed mac-802_11mb.cc.

5.6 Impact of Reaction Mechanism

The impact of the receiver-assigned backoff value to the sender based on the packet size is shown for the same set of scenarios addressed in Chapters 3 and 4.

Case 1: Well-Behaving Nodes’ Packet Size Set to 512 Bytes

Figure 23 shows the normal IEEE 802.11 MAC protocol behavior, and Figure 25 shows the throughput achieved by the nodes after implementing the reaction mechanism. The abbreviation “misb” denotes throughput achieved by the misbehaving node, “other(well)” represents the average throughput achieved by the well-behaving nodes in the network, and “total” denotes the total throughput achieved by the nodes in the network. From Figure 24 it can be seen that throughput of the misbehaving node decreased, such that it was nearly equal to that of the well-behaving nodes. Note that Figures 23 and 24 are the same as Figure 3, 4 (in Chapter 3).

![Figure 23. IEEE 802.11 mechanism with fixed packet size of 512 bytes](image1)

![Figure 24. Reaction mechanism with fixed packet size of 512 bytes](image2)
The effectiveness achieved by the misbehaving node with and without the reaction mechanism is shown in Figures 25 and 26. From Figure 26, it is clear that effectiveness of the misbehaving node is decreased with the reaction mechanism.

![Figure 25. Effectiveness achieved by misbehaving node without reaction mechanism (512 bytes)](image)

![Figure 26. Effectiveness achieved by misbehaving node with reaction mechanism (512 bytes)](image)

**Case 2: Well-Behaving Nodes’ Packet Size Set to 1,024 Bytes**

Figure 27 shows the normal IEEE 802.11 MAC protocol behavior, and Figure 26 shows the throughput achieved by the nodes after implementing the reaction mechanism. Figure 28 shows that throughput of the misbehaving node dropped, even though the node increased its packet size greater than the fixed packet size of 1,024 bytes. Note that Figures 27 and 29 are the same as Figures 5 and 6 in Chapter 3.
The effectiveness achieved by the misbehaving node with and without the reaction mechanism is shown in Figures 29 and 30. From Figure 30, it is clear that the effectiveness of the misbehaving node was decreased with the reaction mechanism.
Case 3: Well-Behaving Nodes’ Packet Size Set to 2,048 Bytes

Figure 31 shows the normal IEEE 802.11 MAC protocol behavior, and Figure 32 shows the throughput achieved by the nodes after implementing the reaction mechanism. Figure 32 shows that throughput of the misbehaving node increased slightly, even though its packet size was less than 2,048 bytes. From Figure 32, it can be inferred that all node throughputs were fairly equal because if $P_{\text{var}} < P_{\text{fixed}}$, then a smaller backoff value was assigned to the sender. Note that Figures 31 and 33 are same as Figure 7 and 8 in Chapter 3.

The effectiveness achieved by the misbehaving node with and without the reaction mechanism is shown in Figures 33 and 34. From Figure 34, it is clear that effectiveness of the misbehaving node is decreased with our reaction mechanism.
5.7 Summary

This chapter explained the design of the new algorithm to reduce the effect of packet size-based misbehavior in wireless networks. In this algorithm, the receiver assigns the backoff value to the sender based on the packet size of the data transmitted by the sender. As a result, the effect of misbehavior was reduced, and all nodes received an approximately equal share of throughput. Even though the sender misbehaved, it achieved the same throughput as the other nodes, which provided a disincentive to the misbehaving user.
CHAPTER 6

CONCLUSION

Recently, wireless networks have experienced considerable development, thereby making security a major concern. As wireless networks become more vulnerable, proper security measures must be taken. A large amount of research is currently taking place in order to provide security in wireless networks. A node in a wireless network misbehaves by dropping the packet destined for other genuine nodes to preserve its energy, or simply floods unnecessary traffic in the channel to restrain other genuine users from accessing the channel. These sorts of misbehaviors have become very common in wireless networks.

Current wireless MAC protocols are designed to provide equal share of throughput for all nodes in the network. However, the presence of a misbehaving node (selfish node deviates its behavior compared to other genuine nodes in the network to get better share of throughput) in the wireless network poses several threats to the fairness aspect of MAC protocols.

This research addressed a new type of selfish misbehavior achieved by altering the packet size of the data sent by the node. In this case, a node misbehaved by altering its packet size without changing any of its MAC parameters. The wireless network scenario was simulated in NS-2, and one node was made to misbehave. Results show that the node with the higher packet size achieved higher throughput.

The packet size-based misbehavior was also formulated mathematically using an M/M/1/K queuing system with two queues, where one queue represented packets from well-behaving nodes, and one queue represented packets from misbehaving nodes. The server served both queues in a round-robin fashion with equal service rates. The same wireless network scenario used in NS-2 was modeled as an M/M/1/K queuing system, and throughput of the nodes
in the network was obtained by solving the steady-state probability equations. The theoretical results obtained were close to practical results and also show that the misbehaving node with a higher packet size attained higher throughput. In order to reduce the effect of misbehavior, the service rates for both the queues were altered. Varied service rates were provided for both queues, and results show that the misbehaving node’s throughput decreased and fairness was achieved among other nodes in the network.

In addition, a slight modification was made to the MAC protocol to reduce the impact of misbehavior. In the standard MAC protocol, a sender chooses a backoff value for transmitting the packet. In this thesis, an algorithm in which the receiver chooses a backoff value for the sender based on the packet size of the data sent by the sender was designed. The receiver was made to assign a higher backoff value for the node that sent the data of a larger packet size. As a result, the misbehaving node waited for a longer time to access the channel. This mechanism helped to increase the chance for other well-behaving nodes to obtain better access to the channel. As a result of this reaction mechanism, fairness was achieved among nodes in the wireless network.

6.1 Future Work

Since this research considered only one misbehaving node, future work would be to consider multiple misbehaving nodes. The same reaction mechanism could be implemented for multiple misbehaving nodes to see how it reduces the effect of misbehavior.

Kyasanur and Vaidya [2] have discussed the type of misbehavior by altering the backoff values used by nodes in the network. The receiver assigning the backoff mechanism was used to reduce the effect of misbehavior. Therefore, packet size-based misbehavior could be combined with selfish misbehavior mentioned by these authors to determine how the combined
misbehavior affects the performance of nodes in the network. The receiver-assigning backoff value could be used as the reaction mechanism against this type of selfish misbehavior, whereby the receiver would assign the backoff value considering both misbehaviors (altering both backoff value and packet size). If a selfish node picked a smaller backoff value and also sent the data of a higher packet size, then the receiver would assign a higher backoff value for that particular node compared to other well-behaving nodes.

The research in this thesis assumed that the receiver was a well-behaved node. But what happens if the receiver is not genuine? This research could be extended to investigate a scenario where multiple observers monitor the channel and develop a reaction mechanism to reduce the effect of misbehavior.
REFERENCES


APPENDICES
l1=855;
muq1=390.625;
l2=61;
muq2=195.31;
mu=37.2;
p1=512;
p2=1,024;
k=3;
w1=1.125;
w2=0.562;
fp1=fopen('Equations.m','w+');
fprintf(fp1,'Ans=solve(');
for i=0:1:k
    for j=0:1:k
        if (i~=0 && i<k && j==0)
            fprintf(fp1,'''
        
        fprintf(fp1,\'p_{%1.0f\_%1.0f}=\(\%1.2f\*p_{%1.0f\_%1.0f}\)\(\%1.2f\*p_{%1.0f\_%1.0f}\)\(\%1.2f\*p_{%1.0f\_%1.0f}\)\/%1.2f\),i,j,l1,i-1,j,(mu1+l1+l2));
            fprintf(fp1,"''

        elseif (i==k && j==0)
            fprintf(fp1,"''
        
        fprintf(fp1,\'p_{%1.0f\_%1.0f}=\(\%1.2f\*p_{%1.0f\_%1.0f}\)\(\%1.2f\*p_{%1.0f\_%1.0f}\)\/%1.2f\),i,j,l1,i-1,j,(mu1));
            fprintf(fp1,"''

        elseif (i==0 && j==0 && j<k)
            fprintf(fp1,"''

        else
            fprintf(fp1,"''

    end
end

fclose(fp1);
APPENDIX A (continued)

fprintf(fp1,'p_{%1.0f\_%1.0f}=%1.2f*p_{%1.0f\_%1.0f}+(%1.2f*p_{%1.0f\_%1.0f})+\ldots+((n-1)*w_{1}\mu_{1})+w_{2}\mu_{2});

elseif (i==0 && j==k)
    fprintf(fp1,"\n\n");
    fprintf(fp1,'p_{%1.0f\_%1.0f}=%1.2f*p_{%1.0f\_%1.0f})/\ldots);\n
elseif (i+j==k)
    fprintf(fp1,"\n\n");
    fprintf(fp1,'p_{%1.0f\_%1.0f}=%1.2f*p_{%1.0f\_%1.0f})/(\ldots));\n
end
end
end

for i=0:1:k
    for j=0:1:k
        if ((i+j)<=k)
            fprintf(fp1,'p_{%1.0f\_%1.0f}+\ldots;\n
        end
    end
end

fprintf(fp1,"\n\n");
APPENDIX A (continued)

fprintf(fp1,'0=1');
fprintf(fp1,'\n');
for i=0:1:k
    for j=0:1:k
        if ((i+j)<=k)
            fprintf(fp1,'p_%1.0f_%1.0f=single(Ans.p_%1.0f_%1.0f);\n',i,j,i,j);
        end
    end
end

fprintf(fp1,'\nC = ');for j=0:1:k-1
i=0;
fprintf(fp1,'p_%1.0f_%1.0f+',i,j);
end
fprintf(fp1,'p_0_%1.0f',k);
end
fprintf(fp1,'\nE = ');for i=1:1:k-1
j=0;
fprintf(fp1,'p_%1.0f_%1.0f+',i,j);
end
fprintf(fp1,'p_%1.0f_0',k);
end

fprintf(fp1,'\nThroughput_queue1= (E*%1.2f + (1-C-E)*%1.2f)*8*10^(-6)',(w1*fmu1*p1),((n-1)*mu1*p1));
fprintf(fp1,'\n\nX = ');for i=0:1:k-1

end
fprintf(fp1,'\nC = ');for j=0:1:k-1
i=0;
fprintf(fp1,'p_%1.0f_%1.0f+',i,j);
end
fprintf(fp1,'p_0_%1.0f',k);
end
fprintf(fp1,'\nE = ');for i=1:1:k-1
j=0;
fprintf(fp1,'p_%1.0f_%1.0f+',i,j);
end
fprintf(fp1,'p_%1.0f_0',k);
end
\nThroughput_queue1= (E*%1.2f + (1-C-E)*%1.2f)*8*10^(-6)',(w1*fmu1*p1),((n-1)*mu1*p1));
fprintf(fp1,'\n\nX = ');for i=0:1:k-1

end
APPENDIX A (continued)

j=0;
    fprintf(fp1,'p_%1.0f_%1.0f+',i,j);
end
fprintf(fp1,'p_%1.0f_0',k);

fprintf(fp1,"n\nY = ");
for j=1:1:k-1
    i=0;
    fprintf(fp1,'p_%1.0f_%1.0f+',i,j);
end
fprintf(fp1,'p_0_%1.0f',k);
fprintf(fp1,'
Throughput_queue2=(Y*%1.2f + (1-X-Y)*%1.2f)*8*10^{-6}',(w2*smu2*p2),
mu2*p2);
fclose(fp1);
Equations();
APPENDIX B

TCL SCRIPT

set opt(chan)       Channel/WirelessChannel
set opt(prop)       Propagation/TwoRayGround
set opt(netif)      Phy/WirelessPhy
set opt(mac)        Mac/802_11
# set opt(mac)        Mac/802_11mb                    #Reaction mechanism
set opt(ifq)        CMUPriQueue
set opt(ll)         LL
set opt(ant)        Antenna/OmniAntenna
set opt(x)          250
set opt(y)          250
set opt(ifqlen)     50000
set opt(seed)       0.0
set opt(nn)         9.0
set opt(adhocRouting)   DSR
set opt(stop)       50.0
set ns_             [new Simulator]
set topo            [new Topography]
set tracefd        [open test3-out.tr w]
set namtrace       [open test3-out.nam w]
$ns_trace-all $tracefd
$ns_namtrace-all-wireless $namtrace $opt(x) $opt(y)
$stopo load_flatgrid $opt(x) $opt(y)
create-god $opt(nn)
$ns_node-config -adhocRouting $opt(adhocRouting) \ 
  -llType $opt(ll) \ 
  -macType $opt(mac) \ 
  -ifqType $opt(ifq) \ 
  -ifqLen $opt(ifqlen) \
-antType $opt(ant) \\
-propType $opt(prop) \\
-phyType $opt(netif) \\
-channelType $opt(chan) \\
-topoInstance $topo \\
-agentTrace ON \\
-macTrace OFF

for {set i 0} {$i < $opt(nn) } {incr i} {
    set node_($i) [$ns_ node]
    #node_($i) setbackoff Receiver 1    #Node follows the backoff value assigned by the receiver
}
$node_ (0) set X_ 1.0
$node_ (0) set Y_ 1.0
$node_ (0) set Z_ 0.0

$node_ (1) set X_ 2.0
$node_ (1) set Y_ 2.0
$node_ (1) set Z_ 0.0

$node_ (2) set X_ 3.0
$node_ (2) set Y_ 3.0
$node_ (2) set Z_ 0.0

$node_ (3) set X_ 4.0
$node_ (3) set Y_ 4.0
$node_ (3) set Z_ 0.0
$node_ (4) set X_ 5.0
$node_(4) set Y_ 5.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 6.0
$node_(5) set Y_ 6.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 7.0
$node_(6) set Y_ 7.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 8.0
$node_(7) set Y_ 8.0
$node_(7) set Z_ 0.0

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(8) $null_(0)

set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 2,048
$cbr_(0) set rate_ 500000
$cbr_(0) set maxpkts_ 10000000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 0.0 "$cbr_(0) start"

set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(1)
APPENDIX B (continued)

set null_(1) [new Agent/Null]
$ns_attach-agent $node_(8) $null_(1)

set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 1,024
$cbr_(1) set rate_ 500000
$cbr_(1) set maxpkts_ 10000000
$cbr_(1) attach-agent $udp_(1)
$ns_connect $udp_(1) $null_(1)
$ns_at 0.0 "$cbr_(1) start"

set udp_(2) [new Agent/UDP]
$ns_attach-agent $node_(2) $udp_(2)
set null_(2) [new Agent/Null]
$ns_attach-agent $node_(8) $null_(2)

set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 1,024
$cbr_(2) set rate_ 500000
$cbr_(2) set maxpkts_ 10000000
$cbr_(2) attach-agent $udp_(2)
$ns_connect $udp_(2) $null_(2)
$ns_at 0.0 "$cbr_(2) start"

set udp_(3) [new Agent/UDP]
$ns_attach-agent $node_(3) $udp_(3)
set null_(3) [new Agent/Null]
$ns_attach-agent $node_(8) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 1,024
$cbr_(3) set rate_ 500000
$cbr_(3) set maxpkts_ 10000000
$cbr_(3) attach-agent $udp_(3)
$ns_connect $udp_(3) $null_(3)
$ns_at 0.0 "$cbr_(3) start"

set udp_(4) [new Agent/UDP]
$ns_attach-agent $node_(4) $udp_(4)
set null_(4) [new Agent/Null]
$ns_attach-agent $node_(8) $null_(4)

set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 1,024
$cbr_(4) set rate_ 500000
$cbr_(4) set maxpkts_ 10000000
$cbr_(4) attach-agent $udp_(4)
$ns_connect $udp_(4) $null_(4)
$ns_at 0.0 "$cbr_(4) start"

set udp_(5) [new Agent/UDP]
$ns_attach-agent $node_(5) $udp_(5)
set null_(5) [new Agent/Null]
$ns_attach-agent $node_(8) $null_(5)

set cbr_(5) [new Application/Traffic/CBR]
$cbr_(5) set packetSize_ 1,024
$cbr_(5) set rate_ 500000
$cbr_(5) set maxpkts_ 10000000
$cbr_(5) attach-agent $udp_(5)
$ns_ connect $udp_(5) $null_(5)
$ns_ at 0.0 "$cbr_(5) start"

set udp_(6) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(6)
set null_(6) [new Agent/Null]
$ns_ attach-agent $node_(8) $null_(6)

set cbr_(6) [new Application/Traffic/CBR]
$cbr_(6) set packetSize_ 1,024
$cbr_(6) set rate_ 5000000
$cbr_(6) set maxpkts_ 10000000
$cbr_(6) attach-agent $udp_(6)
$ns_ connect $udp_(6) $null_(6)
$ns_ at 0.0 "$cbr_(6) start"
set udp_(7) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(7)
set null_(7) [new Agent/Null]
$ns_ attach-agent $node_(8) $null_(7)
set cbr_(7) [new Application/Traffic/CBR]
$cbr_(7) set packetSize_ 1,024
$cbr_(7) set rate_ 5000000
$cbr_(7) set maxpkts_ 10000000
$cbr_(7) attach-agent $udp_(7)
$ns_ connect $udp_(7) $null_(7)
$ns_ at 0.0 "$cbr_(7) start"

#simulation ends
$ns_ at $opt(stop).0002 "puts "NS EXITING...\"" ; $ns_ halt"
puts "Starting Simulation...
$ns_ run