CHAPTER 6

SIMULATION TESTS AND RESULTS

This particular chapter discusses the simulations used to test the newly implemented models. The chapter begins with a thorough description of the simulation setups, conditions and parameters used in the NS simulator. Following that are the results of the simulations that were performed.

6.1 Simulation Environment

Two simulation environments were setup to evaluate the models – one representing a sparse network with a light traffic load and the other a dense network with a high traffic load. In the first setup, 23 mobile nodes are randomly distributed and move randomly in an area of 500 by 1500 metres following the *random waypoint* mobility model. Two stationary nodes represent the source and sink pair of nodes and are located at opposite far ends of the simulation area to make a total of 25 nodes.

In the second setup, 40 mobile nodes move in a similar 500 by 1500 metre simulation area. An additional 10 nodes are stationary nodes and are located at the two far edges of the simulation area in groups of five each. They represent the TCP sources and sinks for the simulation and are arranged such that the sink for a source is always at the opposite far ends of the area and alternate between each other. In both setups, only the TCP sources can generate data packets whilst the mobile nodes are responsible for setting up routes and forwarding packets to the destination sinks. The labelled figures below show an example of the two simulation scenarios used as depicted by NAM (NS' network animator tool).

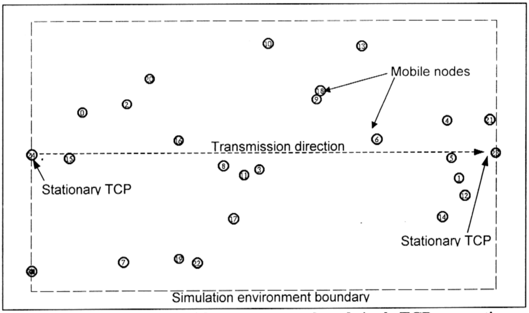

Figure 6.1 – First setup with sparse network and single TCP connection

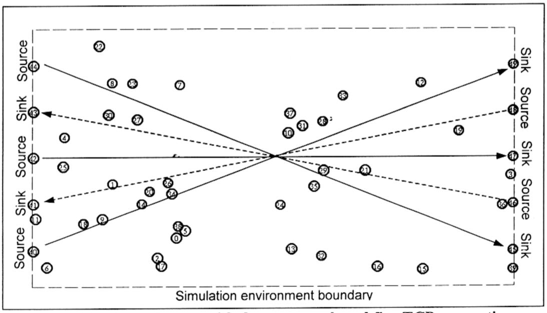

Figure 6.2 – Second setup with dense network and five TCP connections

The movement for the mobile nodes is generated using the *setdest* and *calcdest* utilities provided by NS that have been adapted to suit the 802.11b parameters. In each setup, six different movement scenario sets are generated to represent different movement speeds of the mobile nodes – 0, 4, 8, 12, 16 and 20 metres per second with all the sets having a pause time of zero seconds. Note that this does not mean that all mobile nodes are moving at a fixed pre-set speed during the simulation but move at a random speed anywhere from zero to the maximum set speed and do not stay still at any point in time (except when no movement is used). Since the generation of the node movement is entirely at random and unpredictable, 50 scenarios were generated for <u>each</u> set, thus an average will be taken when the simulations are completed. In total, 300 movement scenarios are generated each (six different speeds with 50 scenarios each), making a total of 600 movement scenarios for the two different environment setups.

Two traffic conditions were used in the simulations – one with a single TCP connection (TCP source and sink pair) and the other with 5 simultaneous TCP connections. The single TCP connection is used to represent a minimal amount of traffic, but since the interference range of a node is approximately 480 metres in a 802.11b network, even this will saturate the entire network quickly making the medium busy most of the time during transmission. The situation with 5 TCP connections represents a heavy amount of traffic and introduces large congestion loads into the scenario as the nodes contend to gain access to the medium to transmit. In all the setups, FTP traffic with 1000 byte sized packets are generated and continuously sent for the entire duration of the simulation which is set at 180 seconds.

The AODV routing protocol is used since it has been shown that it offers better performance in wireless network routing and requires less control packets compared to other protocols (Gupta 2002, Bellardo et al. 1999, Kandukuri 2000). This will encourage testing the performance of the code based on data traffic congestion and not on control traffic. The transport protocol used for traffic is TCP Vegas since it too has been shown that other versions of TCP suffer from an instability problem. Variants of TCP such as Tahoe, Reno and New Reno continuously increase the contention window until it experiences a packet drop (Tong 2003).

Although ideal in wired networks, wireless networks experiencing congestion result in link failures more often than buffer queue overflows resulting in repeated route changes thus making the contention window value invalid. In TCP Vegas, the size of the window is controlled by the round trip time taken for a transmission and does not routinely increase. TCP Vegas also does not take into account packet losses to determine the size of the contention window therefore leading to more stable TCP transfers (Xu & Saadawi 2002).

Since the simulations have to be run both with and without the fuzzy logic control implementation, each movement scenario is therefore simulated twice – once for each condition. With twelve scenario sets, each with 50 scenarios being run twice (once with fuzzy logic and once without), a total number of 1200 simulations is performed to get the result average shown later in this chapter. A summary of the parameters used in the simulation is given in the table on the following page.

Table 6.1 – Summary of simulation parameters

Parameter	Value
Simulation time	180 seconds
Number of mobile nodes	23 and 40
Number of static nodes	2 and 10
Simulation area	500 x 1500 metres
Pause time	Zero (0) seconds
Maximum speed	0, 4, 8, 12, 16 and 20m/s
Mobility model	Random waypoint
Traffic type	FTP agent over TCP with infinite backlog
Packet size	1000 bytes
Number of TCP connections	1 and 5
Routing protocol	AODV
Transmission range of nodes	160 metres

The traffic and movement scenarios are called from a common Tcl script using a batch script. The Tcl simulation script expects 3 arguments to be passed to it when it is run – the traffic scenario filename, movement scenario filename and settings for the fuzzy logic code flag (set to '1' to use fuzzy logic and '0' without). An example command to run a simulation with traffic scenario '*50n5tcp*' and movement scenario '*scen04*' without using fuzzy logic code is:

```
    $ ns run.tcl 50n5tcp scen04 0
```

After completion of the simulation, a trace output file is created with the filename '*scen04_50n5tcp_0.tr*' to differentiate the different simulation runs. The batch script then runs the same scenario but this time using fuzzy logic. Once this run completes it then moves on to the next scenario and repeats the process until all the 50 traffic scenarios in a set have been simulated. The same scripts are used in all the scenario sets and setups (see Appendix B for simulation script).

6.2 Performance Metrics

After the simulations complete, the individual trace output files are parsed using a combination of bash and awk scripts to analyse the performance of the network (see Appendix B for parsing script). The following were used as performance metrics to judge the performance of the implemented code:

Packets sent and received : The number of packets sent and received over the duration of the simulation

Packet delivery ratio : The ratio of the number of received TCP data packets at the sinks to the number of sent TCP data packets from the source.

Packet loss : The number of TCP data packets dropped to the number of TCP data packets being sent by the TCP sources. The difference is given as a percentage.

Routing overhead : The ratio of the number of routing packets sent per TCP data packets successfully received.

Measurements such as throughput and goodput are not used since each randomly generated scenario generates a different amount of traffic and would skew the results if they differ too much. There is also a programming bug present in the NS version used in which the packets transmitted keep growing in size as the headers are added but are never removed therefore NS reports a wrong amount of data transferred. This is especially true in scenarios with maximum node speeds of zero. Since the nodes do not move during the entire simulation, if there are no routes present to the destinations since the beginning then no traffic would be generated at all during the entire simulation run. All the analysis results of the simulations are averaged and discussed in the next section.

6.3 Simulation Results

The results of the simulations are presented as graphs with each value/point in the graph representing an average of the 50 runs with the different movement patterns described earlier. All the graphs have the X-axis representing the node movement in terms of velocity (in metres per second).

6.3.1 Simulation Results of First Setup

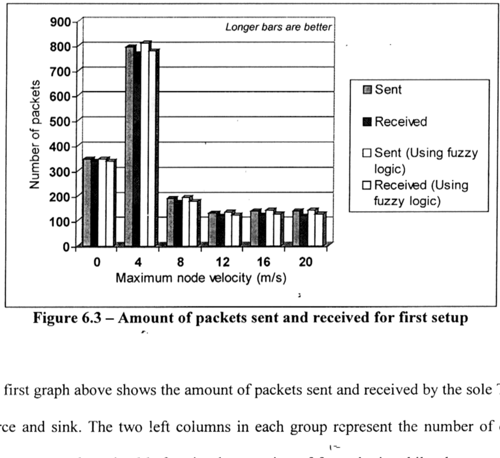

Figure 6.3 – Amount of packets sent and received for first setup

The first graph above shows the amount of packets sent and received by the sole TCP source and sink. The two left columns in each group represent the number of data packets sent and received before implementation of fuzzy logic whilst the two right columns represent those with fuzzy logic running. Although it is not shown clearly due to the scale of axes used, the bars of the graph above indicate that with the implementation of fuzzy logic, both the packets sent and received experienced an increase in amount.

In the figure, the number of packets sent and received peaks when node velocity reaches a maximum 4m/s with an amount nearly four times that of the lowest one. This situation occurred due to the inherent randomness of the simulation scenarios being generated as mentioned earlier in this chapter. In this case, several of the node movement scenarios used in this particular set just so happened to form a stable route from the TCP source to sink throughout the simulation run and therefore allowed higher amounts of data to be sent. Taking into account this random phenomenon, it is therefore more suitable to compare the *change* in the amount of packets rather than the number of packets themselves. This is shown in the following graph.

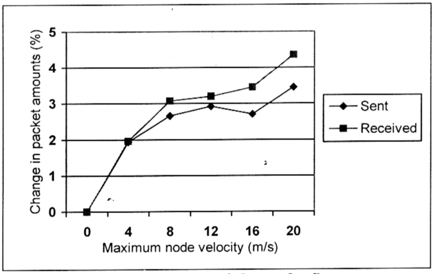

Figure 6.4 – Percentage of change for first setup

This graph, derived from the one in figure 6.3, shows the percentage of change in the number of packets sent and received when fuzzy logic is used. In all scenarios, there is an increase in both packets being sent and received. The higher percentage of packets being received is particularly promising as it indicates that more data is being successfully transmitted across the network even as more data packets are being sent. The exception to this is when node velocity is 0m/s. This is because the nodes are stationary thus making the packet losses due to the lack of any route to the destination and not because of congestion. Since the fuzzy logic code only attempts retransmission if congestion occurs and not because of route failure, there is no change here.

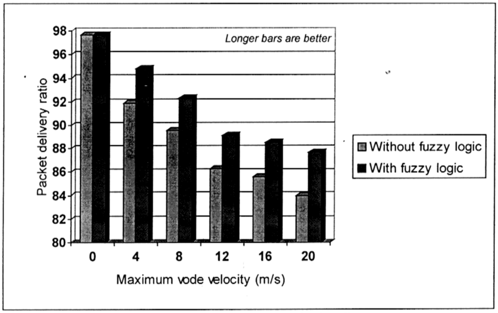

Figure 6.5 – Packet delivery ratio for first setup

The figure above depicts the normalised packet delivery ratio for this simulation. In every scenario, regardless of the rate of velocity, well over 80% of the packets are successfully delivered across the network. Simulations using fuzzy logic show an increase in delivery ratio by as much as 3.7% compared to those running without. Therefore, even at high mobility rates, over 86% of the packets can be delivered when using fuzzy logic control. The delivery ratio when velocity is 0m/s for both simulations with and without fuzzy logic are the same since there is no change in the amount of packets sent or received.

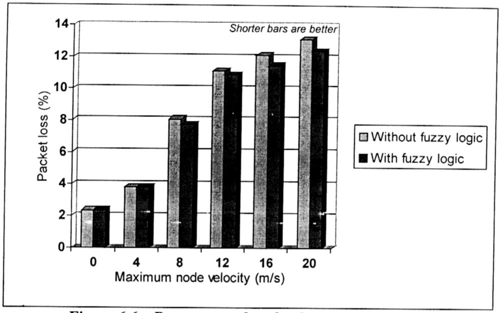

Figure 6.6 – Percentage of packet loss for first setup

Figure 6.6 shows the percentage of packet loss experienced during the run time of the simulations. As mobility increases, packet loss suffers an increase in percentage too as expected. Above 12m/s, when the nodes quite mobile, over 10% of packets sent never reach the destination node. When the nodes are not moving, about 2.4% of the packets are still dropped but in this case, they might be attributed to congestion since the nodes are stationary and routes do not change.

With the addition of the fuzzy logic control, packet loss does reduce but only a nominal amount especially at higher node velocities. When the nodes are stationary, since there is no change in the amount of data packets sent, the percentage of packet loss is the same for both conditions whether with or without fuzzy logic. A decrease of 0.75% may seem trivial, but the result should be taken together with the relatively larger increase in the amount of packets sent and received as shown previously.

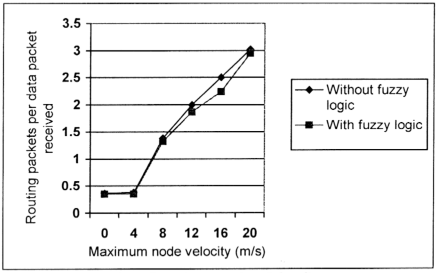

Figure 6.7 – Routing overhead for first setup

In a sparse network with light traffic loads, the number of routing packets transmitted peaks at roughly three packets to every data packet sent. Very few routing packets are sent when the maximum velocity is at 0m/s and 4m/s. A manual parsing of the trace files showed that at 0m/s the nodes were just too far apart from each other to form any routes so after the initial broadcast for routes, the routing protocol simply ignores any further route requests. Surprisingly, the generated scenarios at 4m/s formed several stable routes for simulation run. Since the route was valid throughout the simulation, no additional routing packets were used to patch up broken links.

The advantage of using fuzzy logic in this situation is that even with an increase of up to 4.4% in data packets being sent and received the amount of routing packets required does not increase at all. On the contrary, there is even a *slight* decrease in the amount when fuzzy logic is used. This is especially true with higher and faster node movement where routing overhead is normally expected to increase as routes are often broken and need to be re-established.

6.3.2 Simulation Results of Second Setup

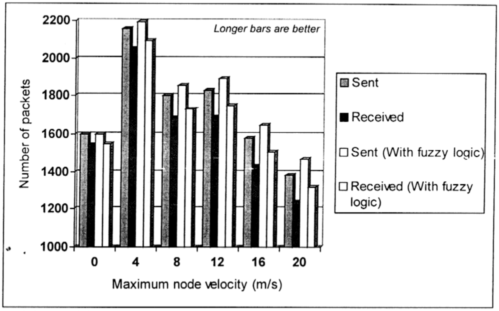

Figure 6.8 - Amount of packets sent and received for second setup

The second setup simulates a dense network and high traffic load. From the results, the first obvious difference is that a much higher amount of data packets are being sent and received at the sources and sinks in the simulations. The bars of the graph in the figure above clearly shows that in simulations using the fuzzy logic control, the amount of packets sent and received is higher than in simulations without.

Similar to the first setup, when all the nodes are stationary, the amount sent and received does not improve by much even with fuzzy logic implemented. This is because there is simply no route available through the network and not because of congestion. In this round of simulations, the number of sent packets increased by two from 1599 to 1601 packets whilst the number of received packets increased by six from 1543 to 1549. This is a small and almost-negligible increase but *is* an increase nonetheless.

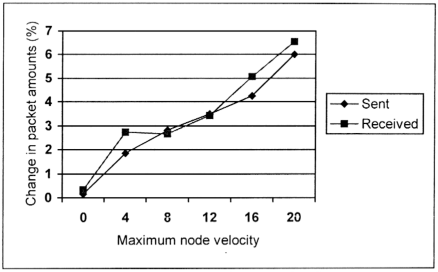

Figure 6.9 – Percentage of change for first setup

From the percentage of change point of view, implementing fuzzy logic manages to increase the amount of packets sent and received in all scenarios. As the node velocity increases, the percentage also increases. This is because in normal situations, mobility causes routes to be broken more often resulting in the need to re-establish routes. With the fuzzy logic control implemented, the routes are assumed to still be valid and a false link failure has been reported instead. Retransmissions are then continued thus resulting in a higher packet amount.

The increase in transmissions of up to 6.6% indicate that more than a few route failures were indeed false link failures. Even when the nodes are stationary an increase of 0.3% in transmissions were made, thereby confirming that congestion does result in false link failures.

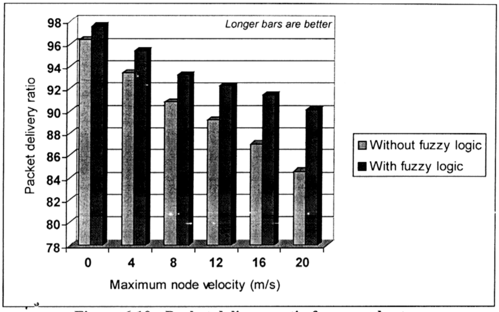

Figure 6.10 - Packet delivery ratio for second setup

Comparing the packet delivery ratios of this second setup, the implementation of fuzzy logic control successfully manages to increase the delivery ratio despite increasing rates of velocity. Under normal circumstances, the delivery ratio drops drastically from 96.4% to 84.6% as mobility increases. With the fuzzy logic control in action, the drop in the delivery ratio is cut so even at high mobility rates the ratio is kept above 90.0% thus ensuring more packets get delivered (the ratio at 20m/s using fuzzy logic is 90.2%).

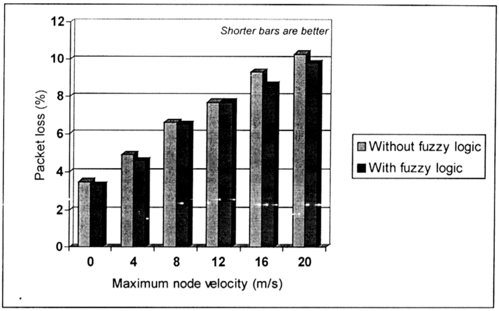

Figure 6.11 – Percentage of packet loss for second setup

Figure 6.11 shows that with five concurrent TCP connections, the percentage of dropped packets increases almost linearly with the rising node velocity. The percentage of packet loss is lower at 10.3% (without fuzzy logic used) when compared the results for packet loss in the first setup which had a higher 12.9% loss. This is probably because the second setup simulated more nodes in the same simulation area thus creating more route possibilities between the source and destination nodes. In the second setup, packet losses can mainly be attributed more towards congestion than route failures.

When the fuzzy logic control is used, the percentage of packet loss never exceeds 10% and in fact, at higher node velocities, the packet loss even decreases. Similar to the first setup, this decrease has to be viewed together with the increase in packets sent and received since the amount between the simulations differ greatly. It should also be pointed out that with fuzzy logic, the drop percentage is never higher than the results of simulations without fuzzy logic.

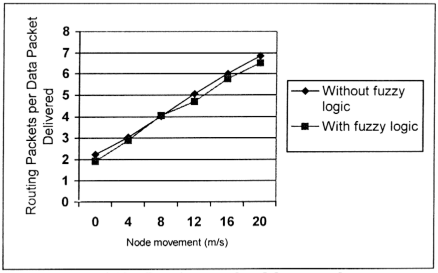

Figure 6.12 – Routing overhead for second setup

The final figure above shows the routing overhead of the simulations in the second setup. This time, a higher number of routing packets are required to establish and maintain routes between the nodes with up to seven routing packets sent for every data packet sent. This result concurs with other research simulations done on routing overhead in WLANs (Broch et al 1998, AhleHagh et al 2003). The overall rise in routing packets is probably due to the congestion which causes the reporting of false link failures.

Similar to the results in the first setup, even with an increase in packet transmissions and packet delivery ratio, the amount of routing packets required remains generally the same when the fuzzy logic control is applied. This means no extra routes were required when in theory, the amount of routing packets should increase by around 5.0% based on the increase in transmitted packets. The fuzzy logic control manages to recover from false link failures and uses existing routes instead of broadcasting unnecessary requests to construct routes that are already present and still valid.