# INTELLIGENT IEEE 802.11B WIRELESS NETWORKS MAC LAYER DIAGNOSTIC CONTROLLER IN MOBILE AD HOC NETWORK

***Wooi King Soo[1], Keat Keong Phang[2], Teck Chaw Ling[3], Tan Fong Ang[4]***
[1] Faculty of Engineering,
Multimedia University, Malaysia. Email: wksoo@mmu.edu.my
[2,3,4] Faculty of Computer Science and Information Technology,
University of Malaya, Malaysia.
Email: {kkphang, tchaw, angtf}@um.edu.my

***ABSTRACT***

*In a mobile ad hoc network (MANET), when a node fails during the RTS-CTS handshake process, the MAC protocol is able to identify whether the failure was caused by link failure or due to congestion by inspecting the recorded radio signal strengths. If the cause is due to actual link failure, then MAC abandons any further attempt to retry communicating with the node. If the failure is due to congestion, the conventional approach in attempt to reconnect the link is wasteful of both time and bandwidth. This will in turn cause packet loss and hence degradation in the TCP performance. In this paper, an intelligent link diagnostic controller for IEEE 802.11b wireless network is proposed. The proposed mechanism incorporates fuzzy controller in the MAC protocol of a node to diagnose the link based on the distances and relative velocities before the attempt to reconnect is made. The incorporated intelligence allows a node to adjust the number of retry in proportional to the probability of getting the node connected. Simulation results show that the proposed fuzzy approach increases the amount of TCP packets transferred whilst reducing the number of TCP packets dropped and maintaining the level of overhead traffic. This mechanism improves the TCP performance by7% on average.*

***Keywords: IEEE 802.11b, TCP, Fuzzy Logic Controller, Power Management***

## 1.0 INTRODUCTION

A MANET is a collection of autonomous mobile wireless nodes which are free to move about arbitrarily. It may operate in isolation, or may have gateways to and interface with a fixed network. The nodes are capable of single-hop or multi-hop connectivity. One important characteristic of MANET is that the network topology may change randomly and rapidly at unpredictable times as the nodes move or adjust their transmission and reception parameters [1]. The network is important in situations where temporary networks are needed but fixed infrastructure does not exist. Some of these applications such as military or serving disaster relief efforts can be time-critical.

A MANET is a potential solution whenever a temporary network is needed and no fixed infrastructure exists. Moreover many applications in this environment are time-critical. It is observed that TCP performs poorly in MANET as demonstrated in [2–6, 7, 8]. Here, a node drops a packet either due to congestion in the shared medium or when it is unable to forward the packet to the next hop. A node cannot reach the next hop node because the next hop node has moved out of transmission range or there are simply too many nodes trying to access the channel at the same time [6]. In a MANET, due to the mobility, bandwidth and energy limitations of nodes, link disconnection and packet loss occur frequently. A high level of packet losses and consequently a high number of TCP re-transmission time-outs degraded the overall network performance. It should be noted that in MANET, mobility alone can cause a degradation of performance of TCP in ad hoc networks even when the load is light. The objective in this paper is to address the issue of degradation in TCP performance due to mobility.

Towards this objective, we propose mechanisms to reduce the number of packet losses and to reduce control/management packet overhead. We are making use of the signal strength measurements at the physical layer to predict possible link failure to a neighbour that is about to move out of range as proposed by [9]. Thus, if the measurements indicate that the link is likely to break, fewer attempts should be tried to restore the link and a search for new route can be initiated more quickly. This reduces packet loss. On the other hand, if the link is very likely to remain within range, the number of retries to restore the link should be increased as to avoid rerouting because rerouting in MANET consumes a lot of network computation power and resources. This reduces control/management packets overhead. Section 2 describes related works. Section 3 and 4 presents the proposed mechanism and its simulation results. Finally, Section 5 concludes the paper with future work.

## 2.0 LITERATURE REVIEW

Various approaches have been suggested to improve TCP performance of MANET at the routing layer [7,8,10,11,12], at the transport layer [2-5, 16, 17], at the MAC layer [9], and at the Physical layer [18].

### 2.1 Routing Layer

[10] proposes the COPAS protocol whereby node-disjoint paths are used. TCP-DATA packets are sent in the forward direction and the TCP-ACK packets in the reverse direction to reduce interference between these packets. [8], on the other hand, uses the same route for both TCP-DATA and TCPACK packets so as to reduce the total number of links that may stall the connection. This mechanism is also capable of predicting the occurrence of a link failure by observing the trend of the signal strengths of packet receptions from neighbours. [7] splits long TCP sessions into multiple segments so that if a link failure occurs in one of these segments, data flow can be sustained on other segments. [11] uses the shortest path as the primary path and the shortest delay path as a backup path to improve TCP performance. They have also demonstrated that using multiple paths concurrently does not improve TCP performance. [12] proposes pre-emptive routing scheme whereby link failures are predicted before they actually break.

[13, 14] improve the performance of WLANs by enhancing the quality of routes between nodes in an ad hoc network in their implementation of Zone Routing Protocol. Routes within the specified intra-zone of a node are proactively maintained to ensure that quick and accurate information is always available. Routes outside the intra-zone are only maintained reactively on per requirement basis only. This thus reduces the need for periodic routing updates between nodes which reduce precious available bandwidth. [15] modifies the DSR ad hoc routing protocol to carry a *ELFN* (explicit link failure notification) message in the original "route failure" message of the protocol. This message is similar to an "unreachable destination" message in ICMP. Once the TCP source receives this message from a node, it proceeds to disable all its retransmission timers and enters a standby mode to prevent any further attempts to transmit to the node involved. Periodically, a route request packet is sent to probe the network to see if a route to the node is re-established. If so, the node leaves the standby mode, restores retransmission timers, and resumes transmission.

### 2.2 Transport Layer

[2-5] use explicit link failure notifications to freeze TCP state upon the occurrence of a route failure. TCP transmission is resumed when a new route is established. This is done through explicit route establishment notifications. [16] uses the TCP layer instead to improve WLAN performance and makes use of feedback information provided by the network layer. *Timeout, connection* and *disconnection* event messages from the network layer are used to control the flow of TCP traffic to ensure the usual TCP congestion control procedures are not invoked prematurely. In [17], the TCP layer is once again used to improve performance in the implementation of *TCP-DOORS*. The sequence numbers of TCP packets are monitored to determine network conditions. If the sequence is out of order, then it halts any congestion control methods before resetting states (e.g. congestion window, retransmission timer timeout, etc.) to values prior to the occurrence of congestion. The rationale behind either action is that the congestion that occurred has since resolved and therefore ACKs can be delivered but only in the wrong order. The congestion is assumed to be result of a route change and since ACKs have resumed but only in the wrong order; therefore TCP should not have to invoke slow start or linear window recovery.

### 2.3 MAC Layer

[9] takes an alternative approach to improving ad hoc WLAN performance by altering the MAC layer instead with their implementation of *Persistent MAC and Proactive Link Management*. To improve TCP performance, Persistent MAC keeps track of neighbouring nodes with active routes based on the signal strengths of transmissions. This information tells a node about neighbours that are still within transmission range such that it can then attempt to identify false link failures caused by congestion and try to re-establish connections with nodes using seven additional RTS-CTS handshakes on top of the seven original retries. Proactive link management increases route availability by periodically checking each route for failures. If a route is deemed to be failing or has already failed, then route discovery measures are taken to obtain a new route to the affected destination even though the route has not yet been requested for a transmission.

### 2.4 Physical Layer

[18] studies the interaction between the physical layer and higher level of the wireless networks. They illustrate these effects by studying the impact of physical layer parameters and path loss on throughput in a multi-hop scenario.

Note that the above mentioned researches attempt to improve the TCP performance by manipulating the routing, transport, MAC, and PHY layers. For the most part, the strategies are typically designed for ad hoc networks because they lack the AP component found in infrastructure networks to help distinguish between losses caused by errors or congestion. In [9], when a node fails during the RTS-CTS handshake, the MAC layer at the node could then figure out as to whether the failure was caused by an actual link failure (i.e. nodes moving out of range) or due to congestion using the recorded signal strengths. If the cause is due to actual link failure, then MAC abandons any further attempt to retry communicating with the node. The link layer could then be informed; depending on routing protocol used, the route is either removed from the routing table and/or a new route request is invoked. If the communicating nodes are deemed to be still within range, the failure is assumed to be caused by congestion. In this case, the MAC layer would then continue its attempts to retransmit the RTS in hopes that a handshake connection can be made to allow transmissions to proceed. The default number of times to attempt the handshake is seven times, but retrying an additional seven times such as in [9] is wasteful of both time and bandwidth.

## 3.0 PROPOSED FUZZY LOGIC CONTROL

Our proposed scheme takes a similar approach to improve ad hoc network performance by adapting existing MAC and PHY layers acquired the related works in Section 2.0. In order to improve TCP performance, the proposed method makes use of information obtained from the PHY layer – the most important of which is the signal strength of transmissions since it is inversely proportional to the distance between the nodes. The information collected is stored at every node in a neighbour information table. The entries of the table are, however, only made for immediate neighbouring nodes and not for separate routes or all available next hop destinations to reduce the amount of table entries required. Additionally, only successful transmissions are kept as a record in a neighbours table at each node. This would reduce memory and system demands at the node and simplify processing of the information table in a node. The information stored in the neighbour table is recalled and used to determine the distance between the nodes using the free-space propagation model [19, 20] as follows.

$$d = \sqrt{\frac{P_t G_t G_r \lambda^2}{(4\pi)^2 P_r L}}$$

where $P_t$ is the default transmission power, $P_r$ the received signal strength, $G_t$ and $G_r$ are the antenna gains of the transmitter and receiver; $\lambda$ is the wavelength, and $L$ is the system loss experienced.

Note that this model assumes the ideal propagation condition, i.e., there is only one clear line-of-sight path between the transmitter and receiver. Once the distance between two communicating nodes are available, it is then possible to determine if the nodes are in and will remain in range by calculating the velocity the nodes are moving apart or closer together. Using the difference in distance and recorded times, the velocity of nodes can also be determined to be either getting closer or moving farther apart. This information can then be used as inputs to the fuzzy logic control to continue retry attempts intelligently and adapt to varying network conditions.

Fuzzy logic is a problem-solving methodology often used to approximate reasoning in situations where information supplied is incomplete, imprecise, ambiguous, or unreliable [21, 22]. The fuzzy approach allows for simpler and faster design and development of systems controllers - instead of having to remodel, redesign, and rewrite the control algorithm as in the conventional approach. Often the only change required to fuzzy logic controls are to the definitions of the fuzzy relations/rules and fuzzy sets. Although many different forms of fuzzy logic inference are available, the Sugeno or simplified fuzzy logic inference method [23, 24] is used since it is fast and avoids complex calculations and integrations, making it uniquely suitable to the task. The proposed inference process uses four fuzzy relations to fuzzify the two inputs - the distance between the nodes and velocity of movement. The relations can be approximated into sentences as follows:

If *nodes* are *near* AND nodes are *separating* THEN retry times is *medium*
If *nodes* are *near* AND nodes are *approaching* THEN retry times is *medium-high*
If *nodes* are *average* AND nodes are *separating* THEN retry times is *low*
If *nodes* are *average* AND nodes are *approaching* THEN retry times is *high*

A *high* number of retries is set at seven times and *low* at a single retry. *Medium* and *medium-high* is set at three and five additional retries respectively. Therefore, the number of additional retries is limited to the range of zero to seven times making maximum possible total of 14 retries each time (inclusive of the initial default seven) before the handshake attempt is actually terminated. To represent the fuzzy sets, arrays are used where the array elements represented points of the fuzzy relation graph. The distance array has seven elements of which the first one identifies whether it is a trapezoid or triangular relation to allow for easier changing of fuzzy relation types.

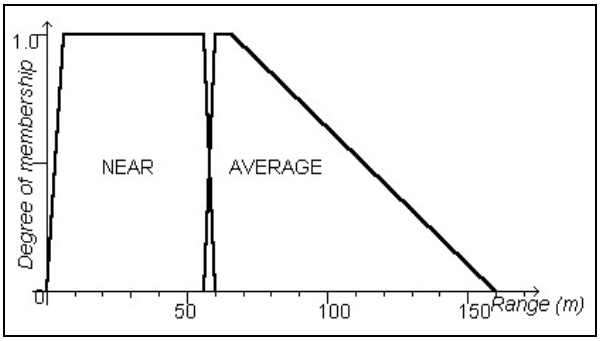

Fig. 1: Fuzzy membership graph for distance

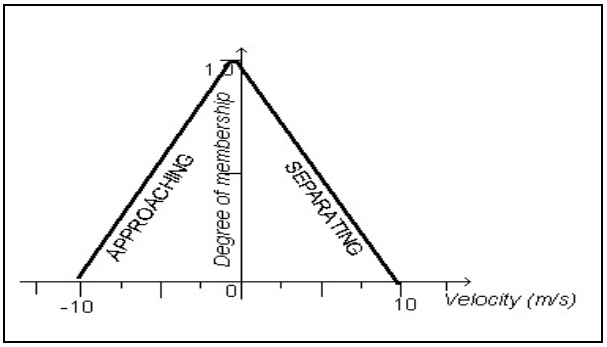

Fig. 2: Fuzzy membership graph for velocity

Since the fuzzy relations above use the AND operator on the consequents, the MIN operation is used on the results of the input fuzzification; so the smaller value between the estimated distance and velocity is chosen during the inferencing stage. The weights used in the Sugeno defuzzification are relative to the output values required (i.e. medium, high, low) and are multiplied with the results of the inferencing stage. The final result is cast to an integer and returned [23, 24].

The pseudocode for the entire process of evaluating the condition of a neighbouring node is as follows:

```
evalRetry(packet P)
  Determine destination node/next hop from P header information
  If node/next hop to P does not exist then
    return 0
  Else
    If entry in table expired then
      Return 0
    Else
  Calculate current distance between nodes
  Calculate movement velocity between nodes
  Use fuzzy logic inference to evaluate conditions
       If nodes are near AND nodes are separating THEN
         retrytimes is medium
       Endif
       If nodes are near AND nodes are approaching THEN
         retrytimes is medium-high
       Endif
       If nodes are average AND nodes are separating THEN
```

```
        retrytimes is low
      Endif
      If nodes are average AND nodes are approaching THEN
        retrytimes is high
      Endif
      Return retrytimes
  Endif
Endif
```

## 4.0 SIMULATION ENVIRONMENT AND RESULTS

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 meters following the *random waypoint* mobility model (Fig. 3). 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 meters simulation area (Fig. 4). 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.

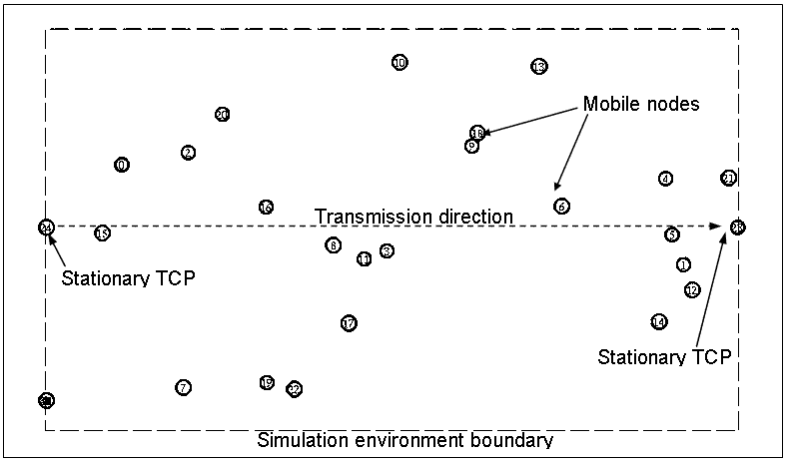


Fig. 3: Simulation setup 1

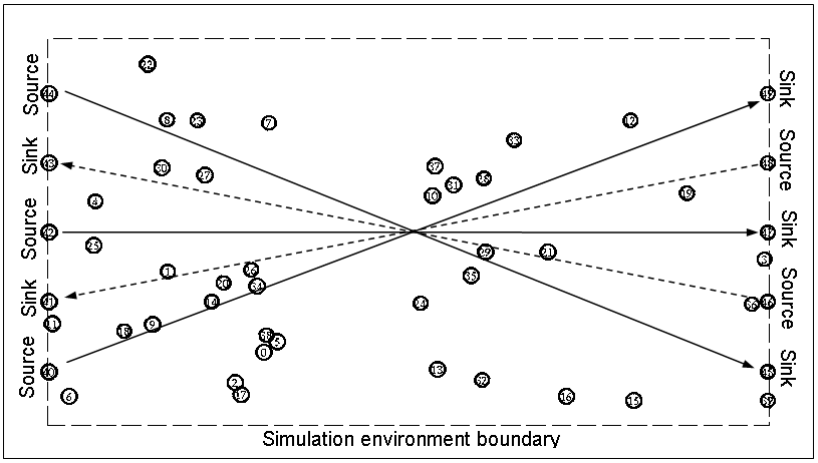


Fig. 4: Simulation setup 2

In each setup, six different movement scenario sets are generated using the *setdest* and *calcdest* utilities in *ns* [25] 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. Since the generation of the node movement is entirely at random and unpredictable, 50 scenarios were generated for each 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 an 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 [26, 27, 28]. 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 such as Tahoe, Reno and New Reno suffer from an instability problem and continuously increase the contention window until it experiences a packet drop [29]. Although ideal in wired networks, wireless networks experiencing congestion will result in link failures more often than buffer queue overflows, resulting in repeated route changes and 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 [19a].

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 averages. Fig. 5a 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 used, the bars of the graph indicates that with the implementation of fuzzy logic, both the packets sent and received experienced an increase in amount. 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. 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.

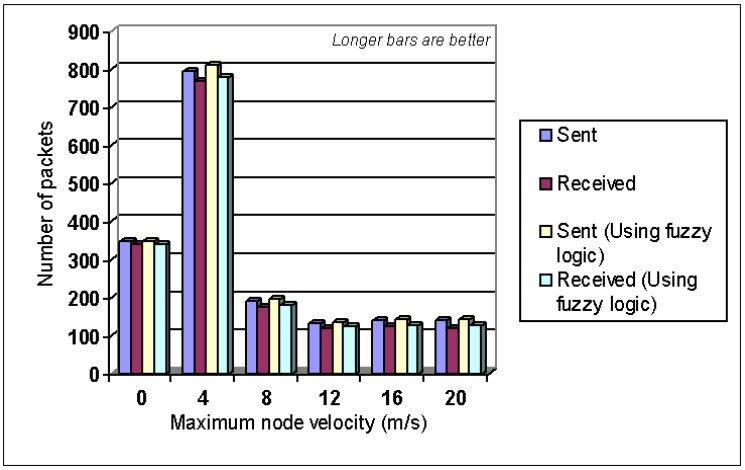


Fig. 5a: Amount of packets sent and received

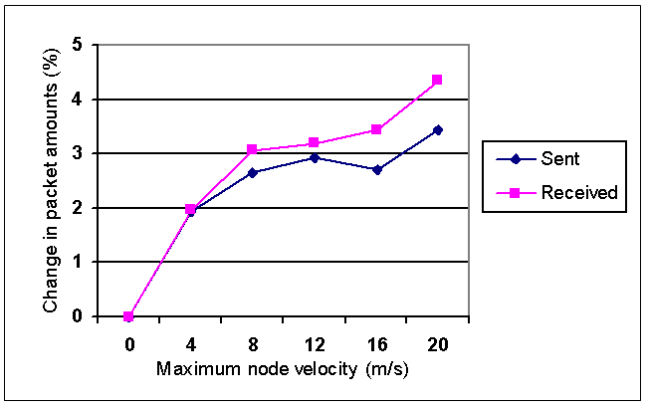


Fig. 5b: Percentage of change

Fig. 5b 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, 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.

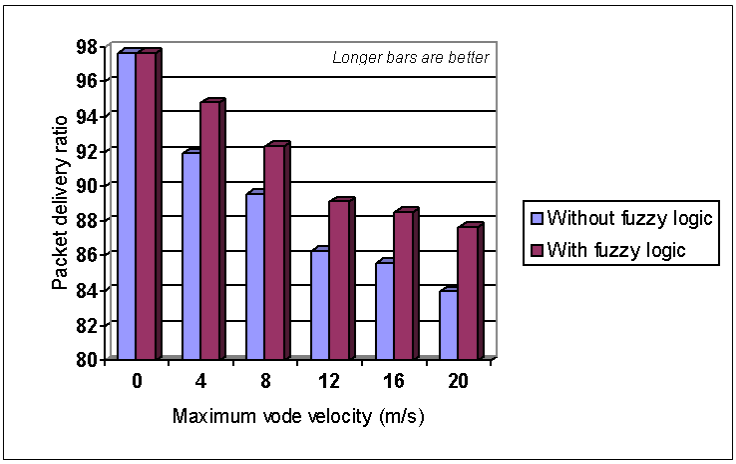


Fig. 6a: Packet delivery ratio

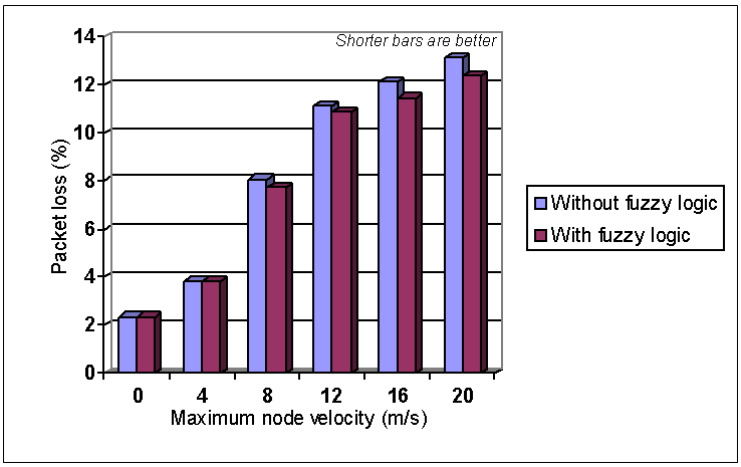


Fig. 6b: Packet loss percentage

Fig. 6a depicts the normalised packet delivery ratio for this simulation. In every scenario, regardless of the value of the node 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 ratios 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.

Fig. 6b 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 are 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 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.

In a sparse network with light traffic loads, the number of routing packets transmitted peaks at roughly three packets to every data packet sent (Fig. 7). 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.

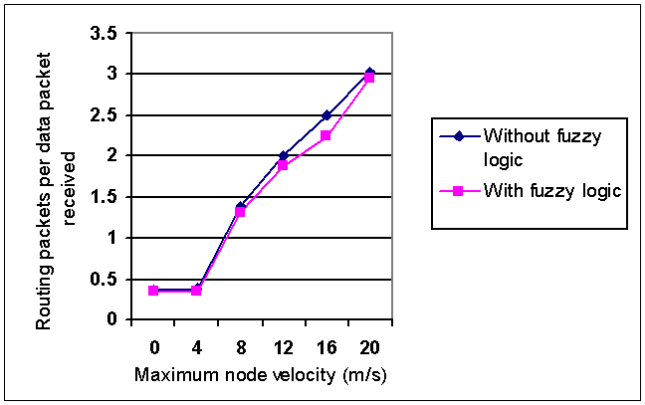


Fig. 7: Routing overhead

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 (Fig. 8) shows that in simulations using the fuzzy logic control, the amount of packets sent and received is higher. 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.

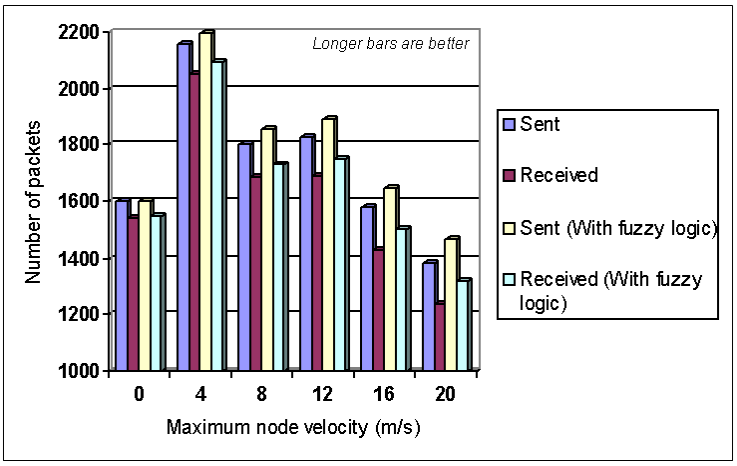


Fig. 8: Amount of packets sent and received

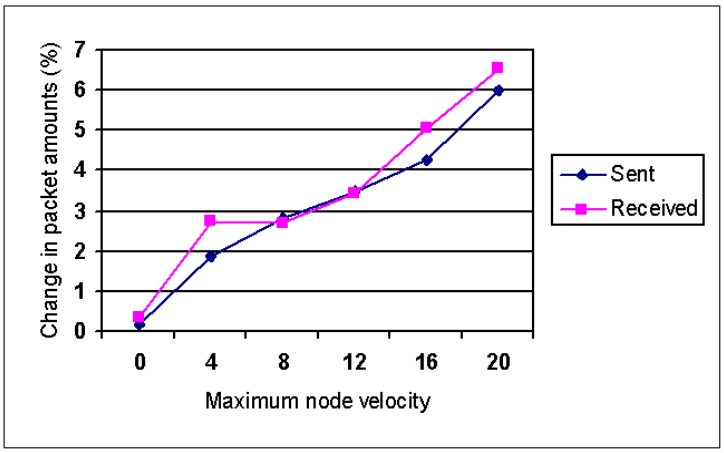


Fig. 9: Percentage of change

Fig. 11 shows that 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 be still valid and a false link failure has been reported instead. Retransmissions are then continued and thus leading to a higher packet amount. The increase in transmissions of up to 6.6% indicates 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 was made, thereby confirming that congestion does result in false link failures.

Fig. 10 depicts the packet delivery ratios of the second setup. The non-fuzzy implementation displays a sharper drop in packet delivery rate when data rate increases. Under normal circumstances, the delivery ratio drops drastically from 96.4% to 84.6% as mobility increases. However, in the fuzzy counterpart, the drop in the delivery ratio is much reduced. In fact, at the highest mobility rate, fuzzy approach is still able to maintain the packet delivery ratio above 90.0%.

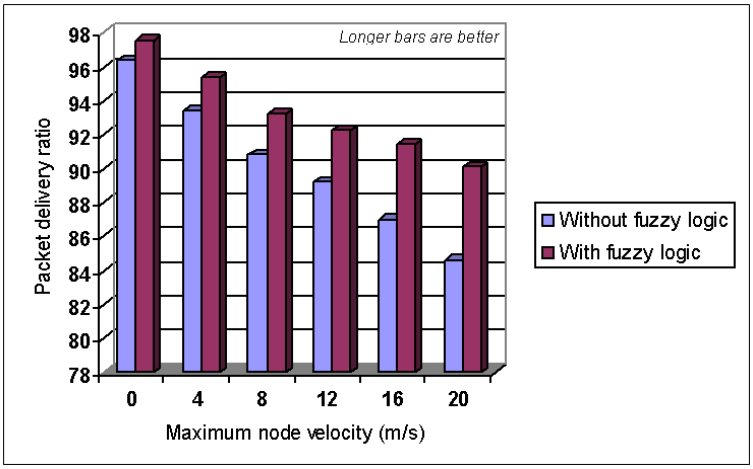


Fig. 10: Packet delivery ratio

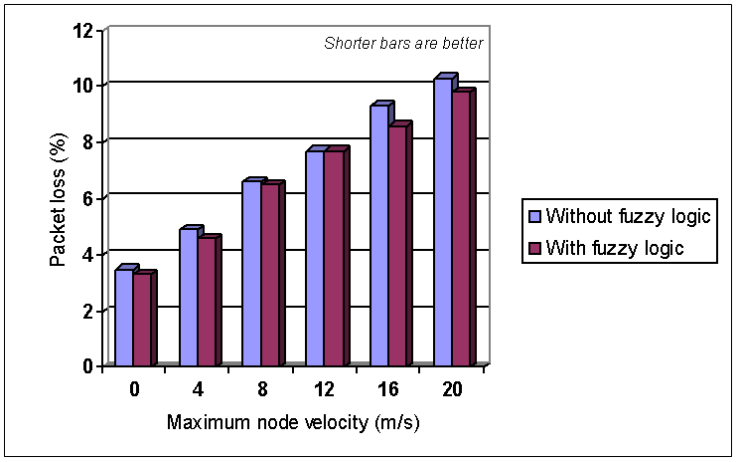


Fig. 11: Packet loss percentage

Fig. 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 highest at 10.3% (without fuzzy control). This figure is lower when compared to the results of packet loss in the first setup (Fig 6b) which had a higher loss of 12.9%. 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, relative to the non-fuzzy counterpart, the packet loss decreases instead. 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 differs 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.

Fig. 12 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 [30, 31]. 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.

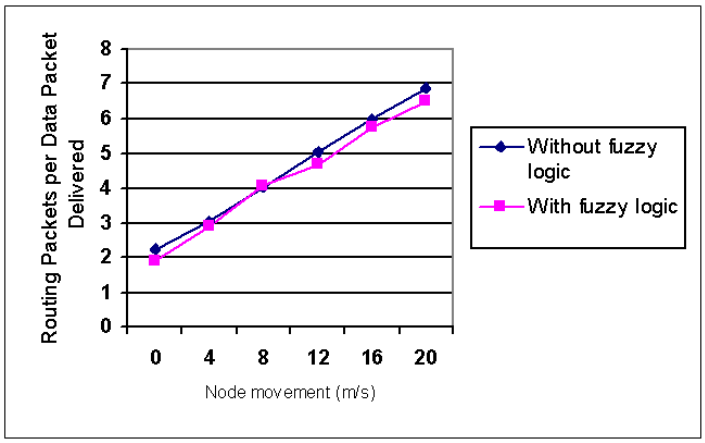


Fig. 12: Routing overhead

## 5.0 CONCLUSIONS

In this paper, we investigated the link diagnostic problem in IEEE 802.11 network. Then, we proposed a new link diagnosis mechanism using fuzzy control to reduce the packet loss due to mobility. The proposed fuzzy approach allows the system to continue retry attempts intelligently. Simulation results showed that the proposed fuzzy approach increases the amount of packets transferred up to 7% whilst maintaining the level of overhead traffic. The number of packets dropped during delivery is also reduced by around 4%, thus giving a higher delivery ratio of packets. Possible future research includes using a more complex but precise method of logic inferencing (e.g. Mamdami or Mizumoto) for finer control of outputs and easier fine-tuning. In addition, the logic control can be adapted to newer implementations of WLAN standards such as 802.11a and g.

## REFERENCES

[1] S. Corson et al, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", *RFC: 2501*, IETF, 1999

[2] K. Chandran, S. Raghunathan, S. Venkatesan, R. Prakash, "A Feedback Based Scheme for Improving TCPX Performance in Ad-Hoc Wireless Networks", in *Proceedings of ICDCS*, 1998.

[3] D. Kim, C.-K. Toh, Y. Choi, "TCP-BuS: Improving TCP Performance in Wireless Ad Hoc Networks", *Journal of Communications and Networks* Vol.3 (2) (2001) 59–71.

[4] G. Holland, N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks", in *Proceedings of ACM MOBICOM*,1999.

[5] J. Liu, S. Singh, "ATCP: TCP for Mobile Ad-Hoc Networks", *IEEE Journal on Selected Areas in Communications* Vol.19 (7)(2001) 1300–1315.

[6] S. Xu, T. Saadawi, "Performance Evaluation of TCP Algorithms in Multi-Hop Wireless Packet Networks", *Wireless Communications and Mobile Computing* Vol. 2 (1) (2002) 85–100.

[7] S. Kopparty, S.V. Krishnamurthy, M. Faloutsos, S.K. Tripathi, "Split-TCP for Mobile Ad Hoc Networks", in *Proceedings of IEEE GLOBECOM*, 2002.

[8] V. Anantharaman, R. Sivakumar, "A Microscopic Analysis of TCP Performance over Wireless Ad-Hoc Networks", in *Proceedings of ACM SIGMETRICS*, 2002.

[9] Klemm, F., et al. "Improving TCP Performance in Ad Hoc Networks Using Signal Strength Based Link Management", *Journal of Ad Hoc Networks,* Vol. 3, 2005, Elsevier, pp. 175–191.

[10] C. Cordeiro, S.R. Das, D.P. Agrawal, "COPAS: Dynamic Contention-Balancing To Enhance The Performance Of TCP Over Multi-Hop Wireless Networks", in *Proceedings of the 10th International Conference on Computer Communications and Networks (IC3N)*, 2002.

[11] H. Lim, K. Xu, M. Gerla, "TCP Performance over Multipath Routing In Mobile Ad-Hoc Networks", in *Proceedings of IEEE ICC*, 2003.

[12] T. Goff, D.S. Phatak, N.B. Abu-Ghazaleh, "Analysis Of TCP Performance On Ad Hoc Networks Using Preemptive Maintenance Routing", in *Proceedings of the International Conference on Parallel Processing (ICPP)*, 2001.

[13] Haas, Z.J. & Pearlman, M. "Determining the Optimal Configuration for the Zone Routing Protocol", *IEEE Journal on Selected Areas in Communications*, 17 (8), 1999, pp. 1395 - 1414.

[14] Haas, Z.J. "A New Routing Protocol for the Reconfigurable Wireless Networks", in *Proceedings of IEEE ICUPC 1997*, San Diego, California. 1997, pp. 562-565.

[15] Holland, G. & Vaidya, N. "Analysis of TCP Performance Over Mobile Ad Hoc Networks", *Wireless Networks*, vol 8, 2002, pp.275-288.

[16] Singh, A.K. & Iyer, S. "ATCP: Improving TCP Performance Over Mobile Wireless Environments". *Fourth International Workshop on Mobile and Wireless Communications Network*, 2002.

[17] Wang, F. & Zhang, Y. "Improving TCP Performance Over Mobile Ad-Hoc Networks with Out-Of-Order Detection and Response", in *Proceedings of The Third ACM International Symposium On Mobile Ad Hoc Networking & Computing*, Lausanne, Switzerland. Lausanne, ACM Press, 2002, pp. 217-225.

[18] Tobagi et al. "Interactions between the Physical Layer and Upper Layers in Wireless Networks", *Ad Hoc Networks*, 2007, pp, 1208–1219.

[19] "Wireless and Mobility Extensions to ns". CMU Monarch Project. Carnegie Mellon University, 1999.

[20] Gast, M. 802.11 "Wireless networks: The definitive guide", California, O'Reilly, 2002.

[21] Zadeh, L. G.J. Klir & B. Yuan (Eds.), "Advances in Fuzzy Systems", Singapore, World Scientific, 2001.

[22] Cordon, O. et al. "Generic Fuzzy Systems", World Publishing, 2001.

[23] Asai, K. "Fuzzy systems for information processing", Japan, Ohmsha Ltd, 1995.

[24] Nguyen, H.T. & Walker, E.A. "A First Course in Fuzzy Logic". Boca Raton, Chapman & Hall, 2000.

[25] The ns manual, The VINT Project, 2007.

[26] Gupta, S. "Performance Evaluation Of Ad Hoc Ruting Protocols Using ns2 Simulations". University of Tenessee, Knoxville, Tenessee, 2002.

[27] Bellardo, J. et al. "Comparison of Ad Hoc Network Routing Protocols". University of California, San Diego, 1999.

[28] Kandukuri, S. "Investigating Performance of Routing Protocols for TCP Transmissions in Ad Hoc Networks". Master's thesis, Stanford University, California, 2000.

[29] Tong, L. "Comparison of different versions of TCP". Wisconsin:University of Winsconsin-Madison. http://homepages.cae.wisc.edu/~ltong/body/ece537.htm, 2007.

[30] Broch, J. et al. "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", in *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Dallas, Texas, 1998 pp. 85-97.

[31] AhleHagh, H. et al. "Statistical Characteristics of Wireless Network Traffic and Its Impact on Ad Hoc Network Performance", in *Proceedings of the Advanced Simulation Technologies Conference*, Orlando, USA, 2003. pp. 66-71.

## BIOGRAPHY

Soo Wooi King is a lecturer at Faculty of Engineering, Multimedia University, Malaysia. He obtained his M. Comp. Sc. in 2005 from University of Malaya. His current research interests include Artificial Intelligence and Wireless Networks.

Phang Keat Keong is a senior lecturer at Faculty of Computer Science & Information Technology, University of Malaya, Malaysia. He obtained his PhD in 2004 from University of Malaya. His current research interests include high speed computer network, network QoS, grid computing, traffic conditioning and traffic engineering, and fuzzy logics.

Ling Teck Chaw obtained his M. Comp. Sc. and PhD in 1996 and 2005 from University of Malaya, Malaysia. He is a senior lecturer at Faculty of Computer Science & Information Technology, University of Malaya. His research areas include core network research, inter-domain Quality of Service (QoS), Voice over IP (VoIP), grid computing and network security.

Ang Tan Fong obtained his B. IT and M. Comp. Sc. in 2000 and 2001 from University of Malaya. He is a lecturer at the Faculty of Computer Science & Information Technology, University of Malaya, Malaysia. His research areas include web services, Voice over IP (VoIP) and grid computing.