

# Enhanced Active Queue Management for Multi-hop Networks

Dr. Naveen Chilamkurti

Department of Computer Science and Engineering,  
Latrobe University,  
Melbourne, Australia.

Email: n.chilamkurti@latrobe.edu.au

Sriram Arul Prakasam

Department of Computer Science and Engineering,  
Latrobe University,  
Melbourne, Australia.

Email: saprakasam@students.latrobe.edu.au

**Abstract**—Wireless networks play a very important role in today’s modern world, convincingly surpassing the wired infrastructure in terms of popularity. Hence, it is important to ensure that services which access wired networks should also be accessible using a wireless network without any performance degradation. One of the most common variants in wireless communications is the Wireless Mesh Network (WMNs). WMNs exploit multi-hop wireless communications between wireless access points. Hence, the effective bandwidth decreases as the number of hops increases in a WMN, thus increasing latency and resulting in reduced performance. This may be due to spatial contention, multipath fading, interference or inefficient queuing mechanisms etc. Here we take queuing mechanisms into consideration and study the QMMN algorithm (Queue Management for Multi-hop Networks) which tends to improve throughput, fairness and reduce global synchronization problems. Based on our study, we implement a modified version of the QMMN algorithm, otherwise called the Enhanced QMMN (EQMMN) algorithm. EQMMN can be considered an effective algorithm which solves the problem of fairness between flows (either responsive or unresponsive) and eventually improves TCP throughput at wireless access points. Our experimental results prove that EQMMN algorithms have better performance characteristics such as throughput (TCP) and fairness index compared to QMMN algorithms..

## I. INTRODUCTION

An increasing demand in the internet community has driven researchers, both in industry and academia, to develop new innovative wireless technological ideas that are relatively cheaper, reliable and flexible. As the end-users continue to increase, so it is necessary to develop applications that satisfy their demands. High bandwidth internet connectivity is required by these applications as they have different demands from the underlying network protocol suite.

In recent years, wireless Local Area Networks (WLANs) have been one of the most successful technologies to provide wireless connectivity for end-users so that requested services can be easily accessed as and when required [1]. This trend has revolutionized the growth of WLANs over the last decade. As the number of wireless devices and users increase, many challenges are being faced.

It would be reasonable for a non-IT professional accessing these services to ask: “Why is my internet connection so slow?” and “What could I do to make my internet connection faster?” etc. Researchers have proposed many solutions to these kinds of problems. In order to answer these questions, we

introduce you to an interesting wireless technology, Wireless Mesh Networks (WMNs).

### A. Wireless Mesh Networks

WMN is an upcoming technology that supplements the wired infrastructure where the wireless backbone provides internet connectivity to mobile nodes or users in offices or residential areas. WMNs are self-organizing, self configuring, self healing and exhibit ease in rapid network deployment as these principles are based on multi-hop ad hoc networks, thereby helping to easily and economically deploy broadband wireless internet connectivity.

One such application in which WMNs are being used as a backbone is the Wi-Fi network. There is an increased popularity in accessing the internet using Wi-Fi hotspots at enterprises, universities, airports, hotels etc. These Wi-Fi hotspots constitute Wireless Access Points (APs) which, in turn, are connected to the wired backbone network.

Traditionally, each AP in a Wi-Fi network is connected to the wired network but nowadays in WMNs, only a few subsets of APs are required to be connected to the wired network. The APs that are connected to the wired network can be called Internet Gateways (IGW) or Mesh Points (MPs). The APs that are wirelessly connected can be termed Mesh Routers (MRs) which are connected to MPs using the multi-hop communication topology. In a similar fashion to wired networks, MRs forward traffic to other MRs, hence they act as intermediate routers. In WMN, MRs forwards each other’s traffic to establish connectivity.

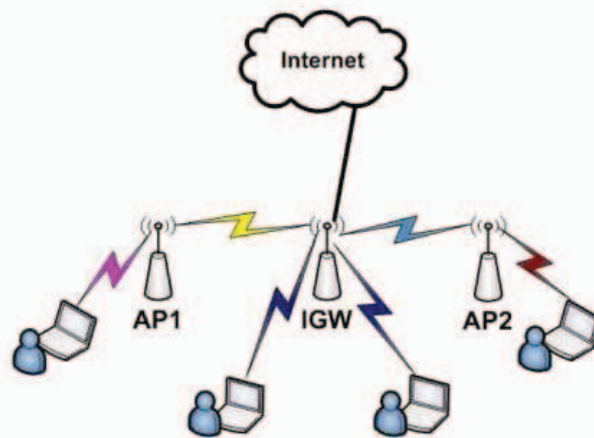


Figure 1: An Example of Wireless Mesh Networks

Figure 1, shows an example of the Wireless Mesh Networking scenario [1].

### B. Problems Faced By Wireless Mesh Networks

Multi-hop wireless communications face several problems such as high interference, increased collisions due to hidden/exposed terminal problems and a high level of congestion. All these factors could result in a low end-to-end throughput and increased latency which does not suit the mesh network applications. Due to these problems, increased interest was focused on issues relating to PHY and MAC standards.

In general, the current queuing mechanisms do not consider the number of hops a packet has traversed when inserted into a link layer queue. This has clearly resulted in starvation and severe unfairness of flows scaling multiple hops. Therefore, the goal of providing equal service to all flows and avoiding the problem of spatial bias has elevated the need to the development of the QMMN algorithm. QMMN considers the longer hop flows and boosts up its performance considerably in comparison with the Drop-Tail algorithm. But even though QMMN improves the performance of longer hop flows, it (TCP flows) considerably suffers at the hands of unresponsive flows (UDP). Therefore, it is really important to devise a strategy that improves TCP's performance in conjunction with unresponsive flows such as UDP [1].

### C. Unfairness Problems in Multi-hop Networks

In this section [2], a brief discussion on wireless mesh networks is presented and also an illustration is given to prove the existing unfairness problem in multi-hop WMNs by using NS2 simulations.

The IEEE 802.11 set up a new task group (802.11s) due to the increase in commercial interests in mesh networks. The main goal for this task group is to extend the existing 802.11 protocol to support the mesh network topology. Several proposals are being taken into consideration out of which one amended protocol allows multiple APs to communicate with each other wirelessly.

Consider the simple mesh networking scenario as shown in figure 2 [2] (four APs). The four APs form a wireless backbone by communicating with each other wirelessly using the legacy IEEE 802.11 based interfaces. AP0 acts as an Internet Gateway (IGW) and hence provides internet connectivity to the other access points. Each AP communicates with their STAs (mobile nodes) using a different 802.11 interface i.e. a different channel. Therefore, no interference can be detected between the communication between two mesh points and the communication between the mobile nodes and its mesh point.

Let us assume that the STAs employ IEEE 802.11 DCF which operates at 2 Mbps with RTS-CTS handshake enabled. A two-ray ground radio propagation model was used with a transmission range of 250m and a carrier sensing range of 550m. In order to avoid unexpected failures, static routing is employed. To avoid complexity, we consider that STAs generate only UDP flows. These UDP flows are aggregated at the corresponding APs and forward them to the IGW (AP 0). A worst case scenario can be simulated using the aggregate load to backlog each AP in terms of packet drop ratio. We also

assume that the UDP flows have a constant packet size of 1024 bytes for all UDP flows.

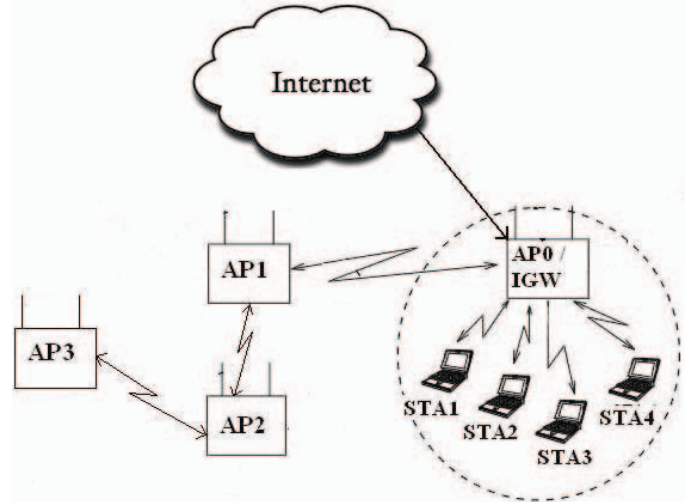


Figure 2: Simple Wireless Mesh Networks Scenario

Figure 3 shows the throughput of the aggregate flows (measured at AP 0) originating from each AP [2]. It shows that there is huge unfairness in the throughput attained by different flows. Hence, the throughput is drastically reduced to negligible with the increase in the hop count (AP 2 and AP 3). Flows from AP 3 need to travel 3 hops to reach IGW (AP 0), only attaining a throughput of 9 kbps of the total throughput whereas the AP 1 flows relatively have a higher throughput of 566 kbps.

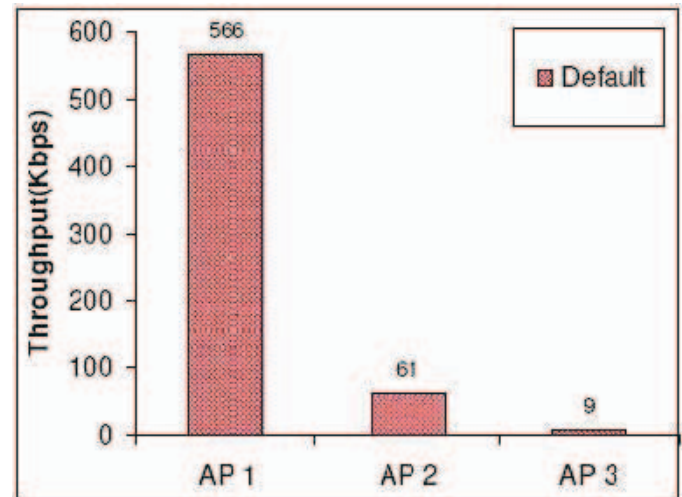


Figure 3: Aggregate Throughput of flows from each AP

Similar results were identified by Gambiroza et al. and the reasons for such unfairness were due to the Hidden and Exposed terminal problems [3]. The link layer (MAC layer) buffer or queue management scheme also has a vital role in this unfairness. In order to illustrate this scenario, a number of UDP packets are transmitted by each AP as shown in figure 4. As shown in figure 4, AP 3 transmits 5753 packets out of which only 482 packets are forwarded to AP 2. The remaining packets are dropped at the MAC layer (link layer) due to the lack of

buffer space. This is may be due to the fact that the local traffic arrives at a more frequent rate and occupies the major share of the buffer. AP 3 packets may often be dropped at AP 2 or AP 1 due to buffer overflow. At the end, only around 110 packets (9 kbps) make it to AP 0. Therefore, this kind of unfair sharing of buffer space adds to the other problems such as hidden and exposed terminal problems, global synchronization problems, multi-path fading and other kinds of interference.

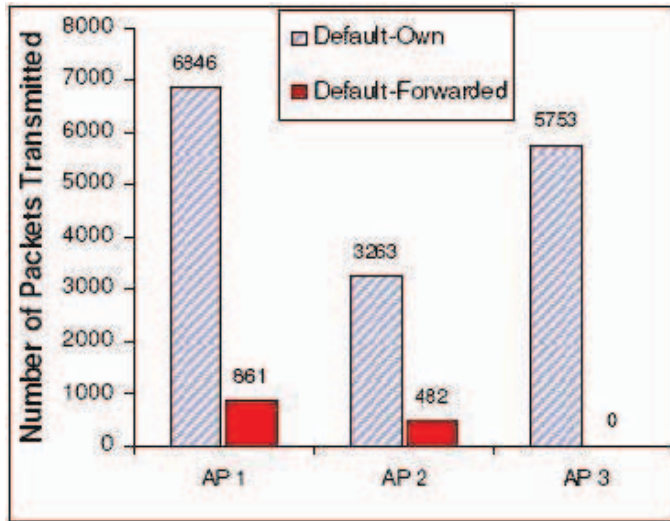


Figure 4: UDP Packets transmitted by each APs MAC layer.

Another interesting and important aspect is that the small number of packets transmitted by AP 2 affects the throughput both at AP 2 and AP 3. This unfair performance is due to the following two reasons. Firstly, whenever AP 1 sends RTS to AP 0, it immediately responds with CTS because AP 0 does not have any traffic to transmit. In contrast, AP 2 loses contention because it has two contenders (AP 1 and AP 3). Secondly, the most important cause of unfair performance is due to the Extended Inter Frame Spacing (EIFS) imposed by 802.11 DCF MAC [4]. Whenever a mobile node (STA) cannot decode a transmission, it should defer its transmission by EIFS time period. But the EIFS deference is cancelled whenever the STA receives a transmission that it can decode. This kind of protection was applicable in WLANs operating in infrastructure mode. However, the results shown here prove to be very unfair in multi-hop networks. The example shown above also illustrates the following: whenever AP 1 sends a UDP packet and the corresponding ACK received from AP 0 cannot be decoded by AP 2 because AP 0 is not within the transmitting range but within the sensing range of AP 2, this makes AP 2 defer for EIFS time period. AP 1 and AP 3 capture the channel with higher probability before AP 2 can again start contending for the channel. Hence, AP 2 has fewer transmission slots when compared to AP 3 or AP 1 which ultimately results in packet loss even though AP 3 transmits packets to AP 2 because of buffer overflow at AP 2.

## II. RELATED WORK

### A. QMMN Algorithm

Experiments in section I.C. indicate that significant throughput degradation occurs due to unfair sharing of the

buffer at significant nodes. QMMN (Queue Management in Multi-hop Networks) is a novel link layer queue management scheme used to solve this problem [2]. The ultimate goal for QMMN is to guarantee a fair buffer share at each AP (or mesh point), for all the flows traversing, irrespective of the hop count.

In the QMMN algorithm, the arriving packet at any AP is either added to the buffer or dropped depending upon the allocated buffer space for the respective source. The packets are buffered if the source has fewer packets buffered than the maximum allocated buffer; otherwise the packets are dropped if there is no residual space available. Therefore, the aggressive nodes cannot suppress the other nodes by merely increasing the traffic rate because QMMN limits the allocated buffer share for each and every source.

The following section describes the data structures and variables used at each AP.

### B. Data Structures

The information on all the active nodes that route packets through mesh points or APs maintains a fairness table [2]. The fields in the table are explained below:

*Source Address:* Through a mesh point, the source node routes one or more flows.

*Max\_Share:* The maximum buffer share allocated for each source node. The total buffer space divided by the number of sources in the fairness table gives us the maximum share space for each node.

*Fair\_Share:* The share that is required by the source node at a mesh point is the Fair\_Share. This share is set to maximum when a new entry is created in the fairness table. The Fair\_Share value is later adjusted according to the estimated arrival rate of traffic from the source node.

*Occupied\_Share:* The number of packets currently buffered for the corresponding source node.

*Arrival\_Rate:* The average arrival rate of packets from its source node.

Table 1: Snapshot of fairness table at AP1

Source Address	Max Share	Fair Share	Occupied Share	Arrival Rate
1	25	25	24	0.015
3	25	17	24	0.06

### C. The QMMN Algorithm

Whenever a packet arrives, the AP (or mesh point) checks whether the fairness table has an entry for the corresponding source [2]. A new entry is created if there is no entry for the source. At this instance, the max\_share is calculated as the total buffer space is divided by the number of sources in the fairness table. Whenever a new entry is created, the fair\_share is initialized to the max\_share and the fair\_share for other source entries is recalculated.



The initialization of the fair\_share of a source to its max\_share is vital because not all the sources might occupy the max\_share of the buffer. This may be due to the different rates at which the node might generate traffic or the inter-arrival times of packets may be low. Hence, by equally reserving the buffer space for the active source, nodes might underutilize the available resources. This could eventually decrease the overall throughput of the network. In order to address this problem, the average arrival rate of the traffic is computed and based on this value, a fair\_share is allocated for each source node at each mesh point. The average arrival rate and the service time are computed as follows:

$$\text{arrv\_rate} = \alpha * \text{arrv\_rate} + (1 - \alpha) * \text{curr\_arrv} \rightarrow \text{eq1}$$

$$\text{serv\_time} = \alpha * \text{serv\_time} + (1 - \alpha) * \text{serv\_time} \rightarrow \text{eq2}$$

Now we calculate the new\_fair\_share for a given source.

$$\text{new\_fair\_share} = \alpha * \text{old\_fair\_share} + (1 - \alpha) * (\text{serv\_time} \div \text{arrv\_rate})$$

We then computed the fair\_share as follows:

$$\text{fair\_share} = \min(\text{max\_share}, \text{new\_fair\_share})$$

The value of  $\alpha$  is set to 0.3 after extensive simulations. This provided better estimates for the actual arrival rates and service time.

After updating the fair\_share value for all sources, the unused buffer space is calculated, otherwise called the residual buffer space. The difference between the max\_share and the fair\_share is known as the residual share. The overall residual buffer for the whole queue can be calculated as follows:

$$\text{residual\_share} = \sum (\text{max\_share}_i - \text{fair\_share}_i) \quad \forall i \in \text{sources}$$

If the source has fewer packets than the fair\_share, the incoming packet will be deterministically enqueued. Packets are accepted even if the source node occupied\_share is greater than the fair\_share but less than the allocated residual buffer for that source entry. This ensures that no single source monopolizes the residual share. Also, whenever the link layer dequeues any packet from the buffer, the occupied\_share for the corresponding source is updated.

In a mesh point, more packets can be buffered than its fair\_share only when the residual\_share is available. If there is any traffic burst, the residual\_share can be temporarily utilized, thus avoiding any under-utilization of the buffer or the residual\_share is introduced to handle fast variations in the traffic load of sources.

The QMMN algorithm's main ideology is to restrict the aggressive nodes by dropping packets in order to protect the flows that traversed multiple hops. Let us assume that each mesh point generates the same traffic load. Therefore, the packets from source nodes that are a few hops away tend to have a relatively low arrival rate due to contention delays at multiple nodes. These nodes rarely exceed their fair\_share, whereas the sources nearby the mesh point frequently pump in packets and eventually consume their source's share. Allowing these packets would consume the other's buffer, hence to avoid

such an event, it is reasonable to drop packets from the near by sources. In this way, QMMN ensures fair allocation of buffer to source nodes.

#### D. Fairness Index

Fairness can be considered as one of the most important criterion implemented in all resource allocation schemes. For instance, when we take a resource from a particular user and allocate it to another user, the newer allocation should be fairer if the receiver has considerably less resources than the giver and vice-versa. A quantified measure of the fairness index is applicable to any resource sharing or allocation problems. The fairness index lies between 0 and 1 which aids intuitive understanding of the fairness index [5]. The following formula represents the fairness index:

$$\text{Fairness Index} = (\sum x_i)^2 / (n * (\sum x_i^2))$$

where  $x_i$  is the Normalized Throughput

$$x_i = \text{Measured Throughput (T}_i) / \text{Fair Throughput (O}_i)$$

### III. PROPOSED ENHANCED QMMN ALGORITHM

#### A. Overview

The performance analysis results of the QMMN algorithm [2] show considerable improvement in fairness values when compared with the Drop-Tail algorithm. These results concentrate mainly on UDP traffic in order to depict its throughput. However, when a TCP flow comes into the picture, the throughput for that TCP flow deteriorates considerably during the deployment of the QMMN algorithm. This claim has been validated in the simulation results shown in section IV.

**used\_residue = occupied\_share - max\_share;**  
**If (used\_residue < (residual\_share/num\_sources))**

**Add the packet p to the queue**  
**occupied\_share++;**  
**residual\_share--;**

**Else**

**Drop the packet p**

**End If**

Figure 5: Residual part for QMMN Algorithm.

It is well known that the QMMN algorithm tends to allocate equal or fair share whenever a new source starts enqueueing packets in the buffer. As time goes on, the QMMN algorithm tends to reduce the fairness rate and eventually consumes the buffer space from its residual share. The code snippet shown in Figure 5 depicts the residual part of the QMMN algorithm.

The packet for a source tends to be dropped if the used\_residue parameter increases beyond the (residual\_share/num\_sources) parameter. Hence, when an unresponsive UDP flow which has reduced the fairness value occupies most of the residual space, it will eventually curb the throughput of its peer TCP flows. In the EQMMN algorithm, we have isolated this problem by distinguishing the TCP and UDP flows. Here, we use the residual\_share parameter to allocate buffer space for UDP sources, and for the TCP sources, we use the used\_residue parameter to allocate buffer space (as stated above). By using this methodology, we can justify an

improved throughput for TCP packets when compared with UDP packets. We have also computed the fairness index for different scenarios. EQMMN outperforms the QMMN algorithm with respect to the throughput of TCP and also with respect to the fairness index as discussed in section IV.

### B. The Enhanced QMMN Algorithm

The pseudo-code shown below depicts the EQMMN algorithm. Whenever a packet arrives, the access point or the APs check whether the corresponding source is present in its fairness table. A new source entry is created if there are no entries in the fairness table. At this instance, the source will acquire the *max\_share* i.e. the total buffer space divided by the number of sources present in its fairness table. We then allocate the *fair\_share* to each source in the fairness table; we set the fair share value to its *max\_share* value initially and recalculate the *fair\_share* values for other sources present in the fairness table.

```

When a packet p arrives:
If (p->source is not in the fairness_table)
    Create an entry for source.
    Recalculate the max_share and fair_share for each node
    Update the residual_share accordingly.
    If a free space is found in the buffer enqueue the packet p
    and update the occupied_share.
Else If (p->source is in the fairness table)
    Update the average inter-arrival rate for the source.
    Recalculate the fair_share and update the residual_share.
    If (occupied_share < fair_share)
        Add the packet p to the queue.
        Update the occupied_share.
    Else
        If (packetType(p) != "tcp")
            used_residue = occupied_share - max_share;
            If (residual_share > 0 &&
                used_residue < (residual_share/num_sources))
                Add the packet p to the queue
                occupied_share++;
                residual_share--;
            Else
                Drop the packet p.
        End If
    End If
Else
    If (residual_share >= packetSize(p))
        Add the packet p to the queue.
        occupied_share++;
        residual_share--;
    Else
        Drop the packet p
    End If
End If
End If
End If
End If

```

Figure 6: Enhanced QMMN Algorithm

In order to avoid an under-utilization of the buffer, we calculate the *inter-arrival rate* and the *service time* for each

source and calculate its fairness values based on the equations as stated in section II.C. (equations 1 and 2). An under-utilization of the buffer would generally lead to reduced throughput for the entire network. Therefore, in order to counteract this issue, we calculate the *fair\_share* (equation 3) for each source during the arrival of a new packet, based on equations 1 and 2.

Upon calculation of the *fair\_share*, we enqueue the packets and then calculate the *occupied\_share* and the *residual\_share*. Here, the *residual\_share* for each source can be termed as the difference between its *max\_share* and the *fair\_share*. If the *occupied\_share* goes beyond the *fair\_share* value, we start using the residual space to enqueue the packets in order to avoid under-utilization of the entire network.

Several experimental simulations suggest that the fairness values for each source tend to become zero exponentially upon increase in time. Therefore, this would result in the utilization of the residual buffer. The graph shown in figure 7 suggests the exponential decline of the fairness value for individual sources.

In the QMMN algorithm, a source (*fair\_share* = 0) with an unresponsive flow would try to occupy the *residual\_share* at a faster pace when compared with a source with a TCP flow. The calculation of *used\_residue* compared with the (*residual\_share/num\_sources*) parameter would give only minimal room for other responsive flows with *fair\_share* zero. This eventually leads to the decreased throughput of the responsive flows as suggested in the simulation.

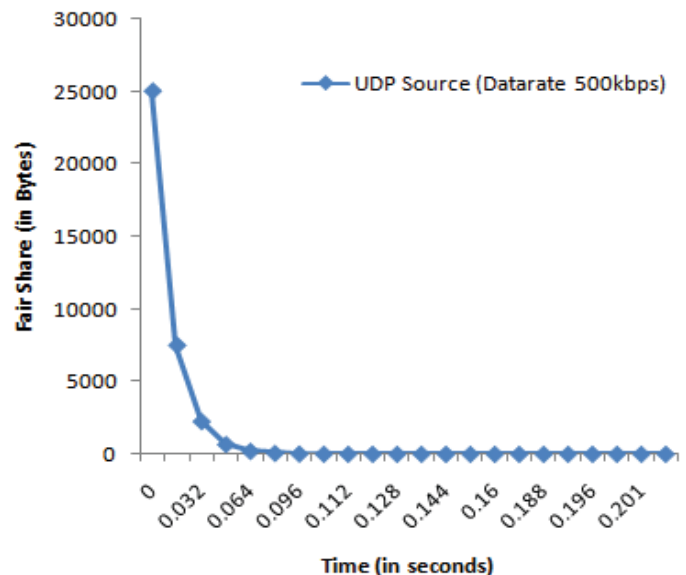


Figure 7: Exponential Decline of the Fairness Value for an UDP source

We isolate this problem by introducing an enhanced version of the QMMN algorithm called Enhanced QMMN algorithm (or EQMMN). Here, we distinguish between flows i.e. we check whether the incoming packet belongs to a TCP or a UDP flow. If the incoming packet belongs to a TCP flow, we check whether the *residual\_share* is greater or equal to the incoming packet size and then enqueue the packet in the buffer, whereas the UDP flows would follow the same concept of *used\_residue* and the (*residual\_share/num\_sources*) parameter. This

algorithm avoids unnecessary packet drops for responsive flows and will eventually punish unresponsive flows. This approach, in comparison with the QMMN algorithm, shows a considerable improvement in the fairness index of flows (both responsive and unresponsive) as depicted in the simulation.

#### IV. EXPERIMENTAL SETUP AND RESULTS

NS-2 version 2.33 has been used to simulate the QMMN and EQMMN algorithms [6] [7] [8]. In this paper, we simulate three different scenarios for both QMMN and EQMMN algorithms and record its performance characteristics. For scenarios 1, 2 and 3, we use 13 nodes to simulate the algorithms i.e. 7 nodes act as traffic sources, 4 nodes as an intermediate node or queue and the final 2 nodes as the traffic sink or destination. In scenario 1, we use sources (TCP and UDP) with equal data rates; similarly in scenario 2 and 3, we consider sources but with different rates. The scenarios depicted below have been able to verify both the algorithms performance characteristics and also the fairness index.

##### A. QMMN & EQMMN Traffic Setup

The figure below depicts the network topology for the scenarios mentioned:

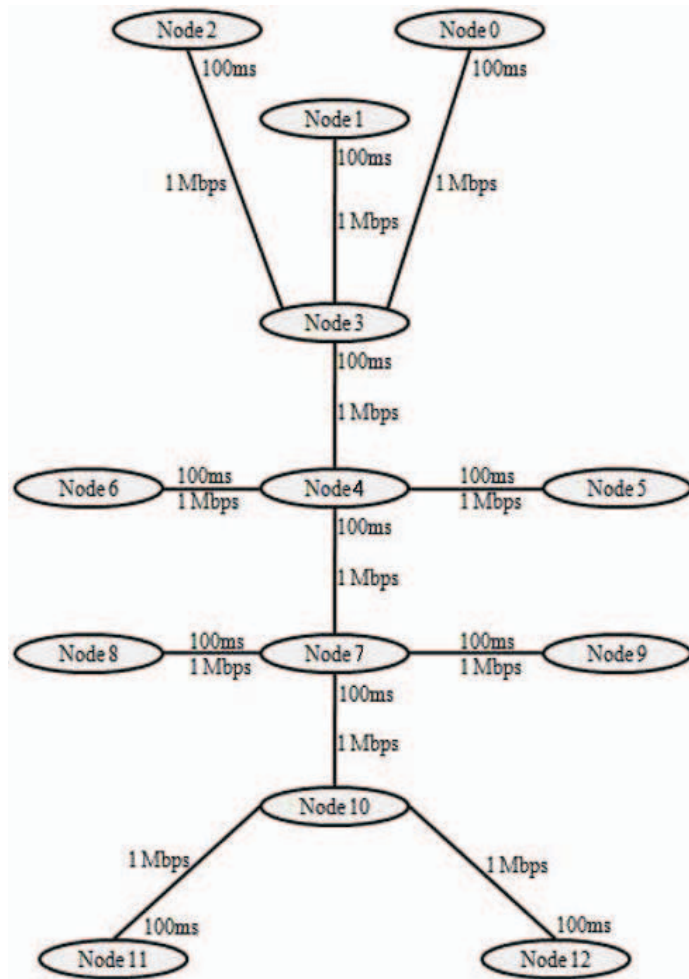


Figure 8: Network Topology

The following parameters shown below depict the configuration setup for all 3 scenarios.

Table 2: Simulation Parameters

Parameters	Simulation 1	Simulation 2	Simulation 3
Source	UDP (CBR) – N0, N6, N8 TCP (FTP) – N1,N2,N5,N9	UDP (CBR) – N0, N6, N8 TCP (FTP) – N1,N2,N5,N9	UDP (CBR) – N0, N6, N8 TCP (FTP) – N1,N2,N5,N9
Start Time	0 seconds	0 seconds	0 seconds
End Time	50 seconds	50 seconds	50 seconds
Q-Limit (qlim_)	100 Packets	100 packets	100 packets
Link Type	Duplex (1Mbps)	Duplex (1Mbps)	Duplex (1Mbps)
Data Rate	500 kbps – N0,N1,N2,N5, N6, N8, N9	800 kbps – N0,N6,N8 300 kbps – N1,N2,N5,N9	300 kbps – N0,N6,N8 400 kbps – N1,N2,N5,N9
Sink	N11 (TCP), N12 (UDP)	N11 (TCP), N12 (UDP)	N11 (TCP), N12 (UDP)
Idle Time	0	0	0
Mean Packet Size (mean_pktsize_)	1000 bytes	1000 bytes	1000 bytes
Propagation Delay	100 ms	100 ms	100 ms

##### B. Performance Analysis

In this section, we compare the EQMMN algorithm with the QMMN algorithm, using simulations in ns2 (version 2.33). We have taken three scenarios into consideration and evaluate with both the versions of the QMMN algorithm by using TCP and UDP traffic for a simulation time of 50 seconds with a mean packet size of 1000 bytes.

##### 1) Performance of TCP and UDP flows at Constant Data Rate

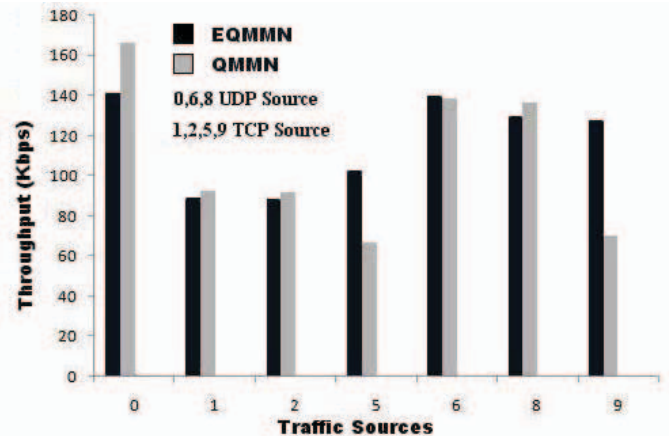


Figure 9: Throughput of flows at Constant Data Rate.

We take the traffic parameters for scenario 1 and perform a simulation with 4 TCP flows and with 3 unresponsive UDP flows having a data rate of 500 kbps. The graph shown above represent throughput for each flow using the QMMN algorithm.



The introduction of an unresponsive UDP flow, the throughput of TCP flows, which uses the QMMN algorithm, has a degraded performance characteristic as the number of hops increases. The throughput of TCP flows aggregates to about 80 kbps whereas the throughput for UDP flows aggregates to about 147 kbps. This result is considerably better when compared to other queuing algorithms such as Drop-Tail, RED etc but it does not ensure complete fairness of flows.

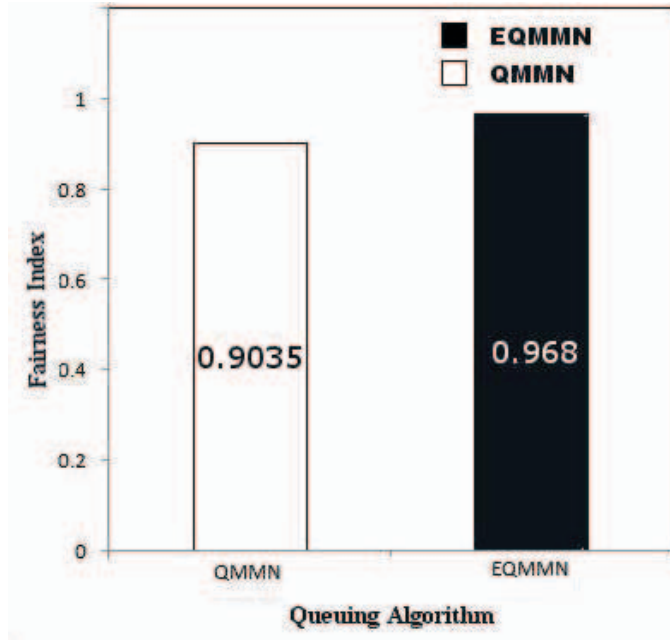


Figure 10: Fairness of flows at constant data rate.

Upon the introduction of the EQMMN algorithm, the unresponsive flows have been punished and the concept of fairness has been entirely granted to TCP flows. The aggregate throughput for TCP flows increases to about 102 kbps from 80 kbps upon the introduction of EQMMN and the aggregate UDP flows have dropped to about 136 kbps which is a remarkable improvement in the performance characteristic of TCP flows.

The fairness index for the QMMN algorithm falls to 0.903 due to the unfair buffer sharing between the TCP and UDP flows, whereas the fairness index of the EQMMN algorithm tends to rise to 0.968 with an unequal number of sources at a constant data rate.

### 2) Performance of TCP and UDP flows at Variable Data Rates

In this scenario, we consider four TCP and three UDP flows with variable data rates and record its performance characteristics based on the simulation.

We introduce three UDP traffic flows at a data rate of 800 kbps and four TCP flows at a data rate of 300 kbps. The simulation time of all these flows extend from 0 to 50 seconds. Upon introduction of unresponsive UDP flows with a higher data rate, the TCP flows suffer and eventually the aggregate throughput drops to around 72 kbps which is 10 kbps less than that of the previous scenario. Upon increase in the data rate of UDP to 800 kbps, the unresponsive flows' throughput

aggregates around 136 kbps. The figure shown below clearly depicts this scenario using the QMMN algorithm.

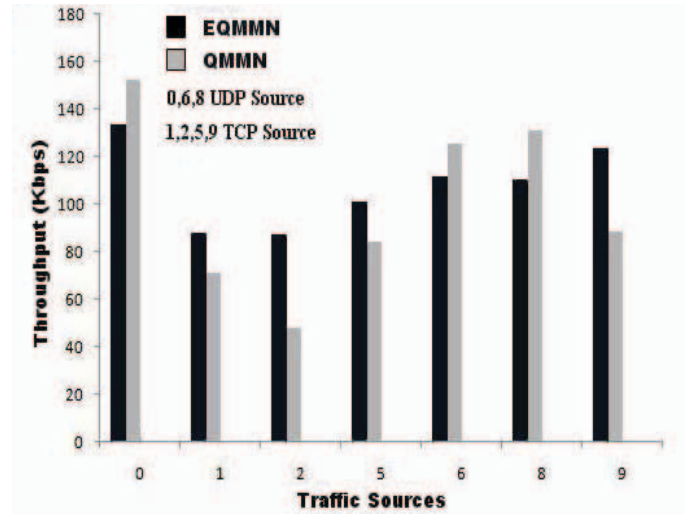


Figure 11: Throughput of flows at variable data rates.

The figure below also shows that the Enhanced QMMN algorithm holds on to the concept of fairness. Even with different flows at different rates, EQMMN was able to deliver an aggregate throughput of 118 kbps for unresponsive UDP flows and 100 kbps for TCP flows. Due to this, the fairness index for EQMMN remains almost the same as the previous scenario (0.978) whereas the fairness index of the QMMN algorithm reduces considerably to around 0.893. This performance result is due to the unfair sharing of buffer space with different categories of flows.

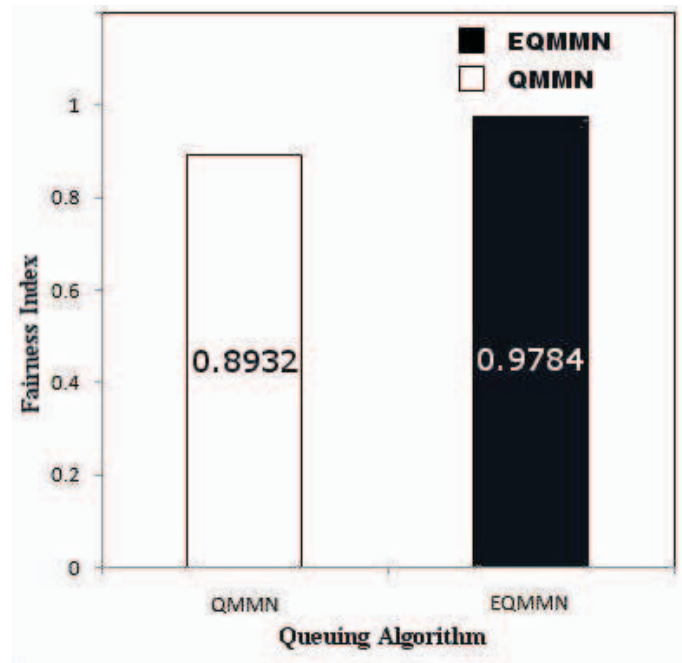


Figure 12: Fairness of flows at Variable data rates.

### 3) Performance of TCP and UDP flows with Smaller Data Rates (Variable)

In this scenario, we consider three unresponsive UDP flows and four TCP flows. The UDP flows have a data rate of around 300 kbps whereas the TCP flows have a data rate of about 400 kbps. Initially, by using the QMMN algorithm, the throughput of TCP drops considerably to about 48 kbps. After 15 seconds of the simulation, the throughput of the TCP flow using QMMN increases to more than 80 kbps and eventually averages around 77 kbps as shown in the figure below.

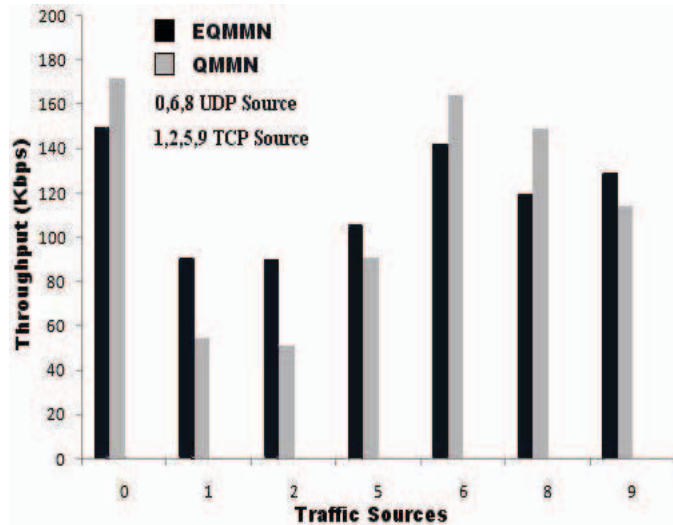


Figure 13: Throughput of flows with Smaller Data Rates (Variable)

The flows using the EQMMN algorithm provide a fair throughput for TCP flow, even in the initial part of the simulation. Even though the aggregate throughput for the TCP flow using EQMMN is a little less when compared to QMMN (for scenario 3), the concept of fairness can be identified in figure 14. Here, the throughput of the TCP flow remains almost constant at around 104 kbps. Unlike QMMN, the fairness index first tends to decrease then it increases drastically. The fairness index for EQMMN remains dominant but the fairness index of QMMN drops to around 0.856.

### C. Summary

Based on these experimental simulations, we are able to verify that the Enhanced QMMN algorithm has a very good fairness index when compared to other queuing algorithms such as QMMN and Drop-Tail. Also, the throughput characteristics for TCP flows using the EQMMN algorithm have seen better improvements, which have eventually lead to the overall improvement in TCPs' performance characteristics.

Even though there have been improvements in the fairness index values by using QMMN (compared with Drop-Tail), EQMMN outperforms QMMN's fairness index by around 5% for both scenarios 1 and 2. But for scenario 3, the fairness index for EQMMN outperformed QMMN by around 10% which is a remarkable performance improvement.

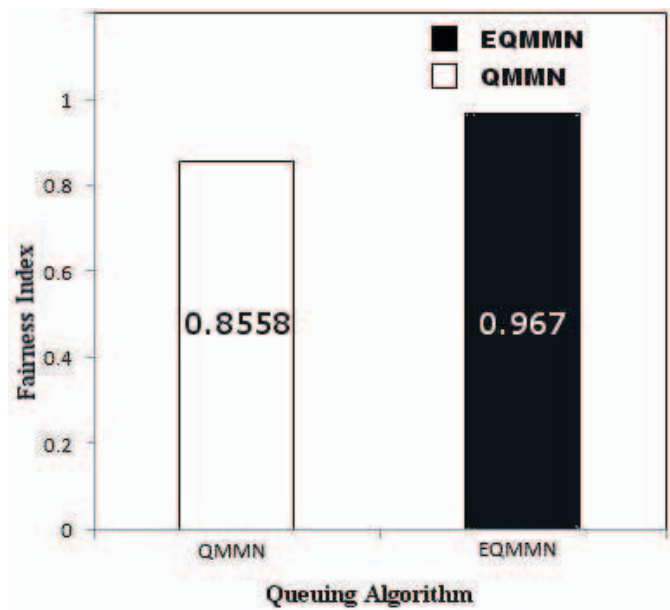


Figure 14: Fairness of flows with Smaller Data Rates (Variable)

## V. FUTURE WORK

Currently, we have focused on two transport layer flows, TCP and UDP. Here, we consider UDP as an unresponsive flow and hence EQMMN restricts its flow by explicitly checking whether it's a TCP or UDP flow. Since there are many other transport layer protocols, such as SCTP (Stream Control Transmission Protocol), DCCP (Datagram Congestion Control Protocol), RSVP (Resource Reservation Protocol), ECN (Explicit Congestion Notification Protocol) etc, by default EQMMN considers all transport layer protocols except TCP as unresponsive flows and hence punishes these flows. In our future implementations, depending on the type of applications used, we will be able to modify the EQMMN algorithm by considering different kinds of transport layer flows and hence improve fairness.

## VI. CONCLUSIONS

The Enhanced QMMN algorithm has been designed to enhance performance characteristics for responsive flows. The EQMMN algorithm implemented in this thesis was designed in order to improve the fairness index and also increase the throughput of TCP flows whenever an unresponsive flow comes into the picture. The tests conducted by simulating the scenarios show that the given solution is able to improve fairness and also increase the throughput of the responsive TCP flow considerably when compared with the QMMN algorithm. The current paper also provides a platform for the future development of the proposed solution by extending the EQMMN algorithm to support other transport layer protocols and eventually improve its performance characteristics.

## REFERENCES

- [1] Nadiraju N.S., "A Cross-layered approach for Achieving Fairness in Multi-hop Wireless Mesh Networks", PhD dissertation, University of Cincinnati, Cincinnati, Ohio, USA. May, 2007.



- [2] Nadiraju N.S., Nadiraju D.S., Cavalanti D., Agarwal D.P., "A Novel Queue Management Mechanism for IEEE 802.11s based Mesh Networks", In Performance, Computing and Communications Conference, 2006, IPCCC 2006, 25th IEEE International, pp-168, Apr. 2006.
- [3] Gambioza V., Sadeghi B., and Knightly E. W., "End to End Performance and Fairness in Multi-hop Wireless Backhaul Networks", In MobiCom '04: Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, pp-287-301, 2004.
- [4] IEEE Std. 802-11. "IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification," ISO/IEC 8802-11:1999 (E), Aug., 1999.
- [5] Jain R., Chiu D., and Hawe W., "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems", DEC Research Report TR-301, Sep. 1984.
- [6] "UCB/LBNL/VINT Network Simulator (NS- v2.33)", Available: <http://www.isi.edu/nsnam/ns/index.html> [Accessed: Mar. 10, 2009]
- [7] "The Network Simulator ns-2: Documentation", Available: <http://www.isi.edu/nsnam/ns/ns-documentation.html> [Accessed: Mar. 11, 2009]
- [8] Issariyakul T, Hossain E, "Introduction to Network Simulator NS-2", NY: Springer, 2009.