# A Low-Latency TDMA Scheduler for Multi-hop Cluster Based MANETs with Directional Antennas

Michael Iannacone, Yamin Al-Mousa, Nicholas Martin, Nirmala Shenoy, and John Fischer

**Abstract.** For Mobile Ad Hoc Network (MANET) applications which involve large propagation delays and/or directional antennas, a Time Division Multiple Access (TDMA) Medium Access Control (MAC) is a resource- and bandwidth-efficient solution. Meanwhile, clustering is a solution to the scalability and high mobility which is commonly required by MANETs. Here we develop a system which combines a TDMA MAC using directional antennas with the Multi-Meshed Tree (MMT) algorithm, which handles clustering and routing tasks. Some of the benefits of this combination include being able to synchronously schedule all intra-cluster routes as they are formed, being able to optimize the intra-cluster schedules for low latency, and being able to calculate these schedules with knowledge of only the intra-cluster topology, which is already maintained by MMT. We first analytically determine the end-to-end latency under various cases, and then confirm these results for several cases through OPNET simulation. Additionally, we note the high degree of slot re-use which is possible due to the use of directional antennas, which is demonstrated by the simulation results.

**Keywords:** Mobile ad hoc networks (MANTEs), time division multiple access (TDMA), directional antennas, clustering.

## 1 Introduction

Mobile Ad-Hoc Networks (MANETs) are self-organized wireless networks of potentially mobile nodes. This means that the signal characteristics of individual links often change rapidly, resulting in links which are constantly breaking and forming. This in turn results in a constantly changing topology as the network re-organizes around these new conditions.[1] These requirements impose serious challenges for the Medium Access Control (MAC) and routing systems which can be used in MANETs.

### 1.1 Routing

To address these requirements of MANETs, we adopt an algorithm called Multi Meshed Tree (MMT). The clusters MMT forms are multi-hop clusters of homogeneous nodes, which provides highly scalability and provides routing with low overhead.[2] The topology within these clusters is a meshed tree, rooted at the cluster head, such that each client node resides in several branches simultaneously, which

J. Zheng et al. (Eds.): ADHOCNETS 2009, LNICST 28, pp. 896–912, 2010.

provides it with multiple, redundant paths to one or several of the cluster-heads. These paths are proactively established, and represented as Virtual IDs (VIDs) associated with each node. These VIDs contain much important information implicitly, such as the parent node's VID, the cluster-head's Unique ID (UID) and VID, and the number of hops to reach the cluster-head.[3]

Because many nodes will typically be members of multiple clusters, MMT is capable of forming inter-cluster routes by using the border-nodes as gateways between these adjacent, overlapping clusters. An extension to MMT for reactive inter-cluster route discovery, Reactive MMT (RMMT), is discussed in [2]. This paper assumes that the proactive and reactive routes are established using the MMT/RMMT algorithms and focuses on the scheduling within a cluster using the proactive paths setup within the cluster. (Our analysis of end-to-end (ETE) latencies on these inter-cluster routes simply assumes that a valid path exists and that it is known to the network; here we are not explicitly concerned with the inter-cluster route discovery mechanism.)

## 1.2 MAC

MAC protocols generally fall within one of two paradigms: contention-based or reservation-based. Reservation based MAC schemes are constrained by the need for a central scheduler and a high level of synchronization among the nodes, while contention based MACs are distributed but suffer due to collisions that can occur due to the random nature of the access process.

### 1.2.1 Contention Based Protocols

Contention based protocols, before sending any data, must sense the state of the medium to see if the channel is busy or idle by detecting the presence of carrier signals on the media. This can be difficult when considering very large distances or directional antennas, because a node's sensing of the carrier is either delayed significantly because of the propagation delays, or is spatially incomplete due to the directionality of the antennas.[5]

Most contention-based MAC protocols use Request to Sent (RTS) and Clear to Send (CTS) messages to avoid collisions, and this works well for short distances and when using omni-directional antennas. However, when propagation delays are very large, the latency between when a node sends an RTS and receives the corresponding CTS can be significant, i.e. delays on the order of milli-seconds if the distances are in the range of 100s of kilometers. The time which the media will need to be reserved for after the RTS/CTS is complete will also need to be increased significantly to deal with these propagation delays. Additionally, due to the delay in sensing the carrier state, collisions during the RTS/CTS process will become more common. These factors could lead to a higher number of retransmissions, very high latencies, and very low throughput relative to the link capacity.[4]

Worse, when directional antennas are considered, the RTS/CTS process breaks down and no longer solves the hidden terminal problem. This happens because one pair of nodes can exchange RTS/CTS messages, while a 3$^{rd}$ node, which did not necessarily hear this exchange due to the position of its own antenna, may attempt to send its own RTS to the destination node while is it still busy with the first session.[5]

This is shown in Figure 1, where, from left to right, node B send an RTS to A, which node C does not hear because it is busy with another session. Node A sends a CTS to B, which C again does not hear. Later, while node B is sending data to A, node C interrupts this by sending an RTS to A. (This simple example assumes that only the transmitting antennas are directional, but analogous problems arise when both transmitters and receivers are directional.)

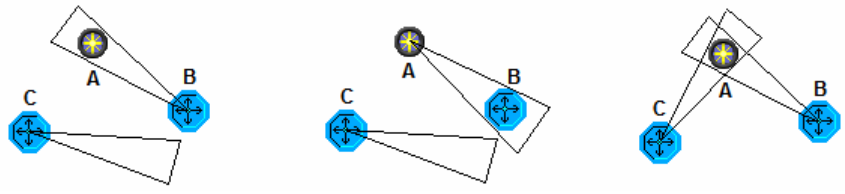


**Fig. 1.** Node C interrupts the message from B to A, because it did not hear the RTS or CTS

### 1.2.2 Reservation Based Protocols

When using directional antennas and very long-range links, a scheduled access system presents an attractive alternative to contention-based MACs. Reserving the links ahead of time allows both the receiving and transmitting antennas to be properly aimed before any data transmission begins, and eliminates the difficulties of carrier sensing and RTS/CTS message exchange. Additionally, when using highly directional antennas, interference is greatly reduced at nodes other than the intended receiver, so multiple non-interfering links can be scheduled within the same vicinity, essentially utilizing spatial-TDMA.[5, 7, 10]

When assigning nodes access for reservation based protocols, the two main paradigms for doing this are synchronous or asynchronous. In the asynchronous assignment channels are allocated based on traffic demands, while in synchronous assignment channels are allocated to all established links that may support communications. Each has strengths and weaknesses under different conditions, which are briefly discussed later, but any detailed comparison of these scheduling paradigms is beyond the scope of this work: here we focus entirely on synchronous scheduling.

## 2 Proposed Solution

Synchronous time slot allocation is normally a challenge to implement over multi hop wireless networks, especially when considering the frequent topology changes common to MANET applications, however we can utilize many properties of MMT to greatly simplify schedule creation. Due to the cluster-based topologies MMT forms, there is no need for any node to know the complete network topology to form a schedule for the cluster: only the intra-cluster topology is needed, and that is only at the cluster head, which already maintains this information for routing purposes. Creating non-conflicting schedules between clusters can be readily achieved by using information from the border nodes, which are members of more than one cluster. The use of this information means that there is no need for each cluster to operate on a

different channel, different CDMA code, or any other channel division method between clusters, which can be an expensive use of bandwidth which is needed by many other inter-cluster TDMA systems.[9]

In this MMT based synchronous TDMA system, the slots are scheduled for any link as soon as it is added to the cluster by the cluster head. In this way, we only schedule slots for the links that will be used by the routing trees (rather than every link that can physically be heard) and the fact that the schedule is only calculated for nodes within the cluster results in simpler computations, and the fact that no additional topology discovery is needed results in reduced message overhead. However, perhaps the largest benefit is that we can schedule these links sequentially along the route, and therefore significantly reduce end-to-end latencies, for both intra- and inter-cluster multi-hop routes.

### 2.1 The Scheduling Algorithm

To better understand the capabilities of the proposed scheduling algorithm, we first consider a random scheduler, where each link on each path is assigned a slot at random. In this simplest case, the average latency between any two nodes along a route would be equation 1a, and the worst case would be equation 1b, where Ft is the frame length, and h is the number of hops on the route.

$$T_t = h*( ½ *Ft) . \quad (1a)$$

$$T_t = h*Ft . \quad (1b)$$

This is because after completing each hop, the packet must wait for the next slot, which will occur at a random time within the frame. Note that while the multi-hop latencies will be much higher with this scheme than with our algorithm, both schedulers will give latencies within some bounded range, assuming that no links in the path are over-subscribed, which would introduce queuing delays.

The proposed algorithm, unlike the random scheduler above, schedules all of its links in order, according to one simple rule: when scheduling a downstream slot for any node ‘A’ to its child node ‘B’, the link ‘A->B’ can only be assigned a slot after ‘A’ has received from its own parent node, if it has one. Thus, a node can only transmit downstream after all of its upstream relatives along that path have already transmitted. This same rule applies in reverse, when scheduling slots for transmitting back upstream. Besides this rule, all slots are simply chosen first-come first-served (FCFS): a link is assigned the first slot which both satisfies this rule and in which both the sending and receiving node are not busy.

Scheduling slots in order according to their routes has been proposed before, but this has always been done with reactive routing protocols, scheduling on a per-session basis, such as [8]. MMT’s unique characteristic of pro-actively maintaining several multi-hop intra-cluster routes for each node makes it possible to schedule its TDMA slots pro-actively. Indeed, if the network is not cluster-based, it is not feasible to follow this rule for every possible route in the network, and if the network is instead based on single-hop clusters then the rule of slot ordering is meaningless for intra-cluster scheduling. However, in a MMT-based network, which maintains multi-hop clusters, it is possible to schedule this way, so that every allowed parent-child link will have a slot assigned to it before traffic is present, and all of these links will be in-order according to the rule above.

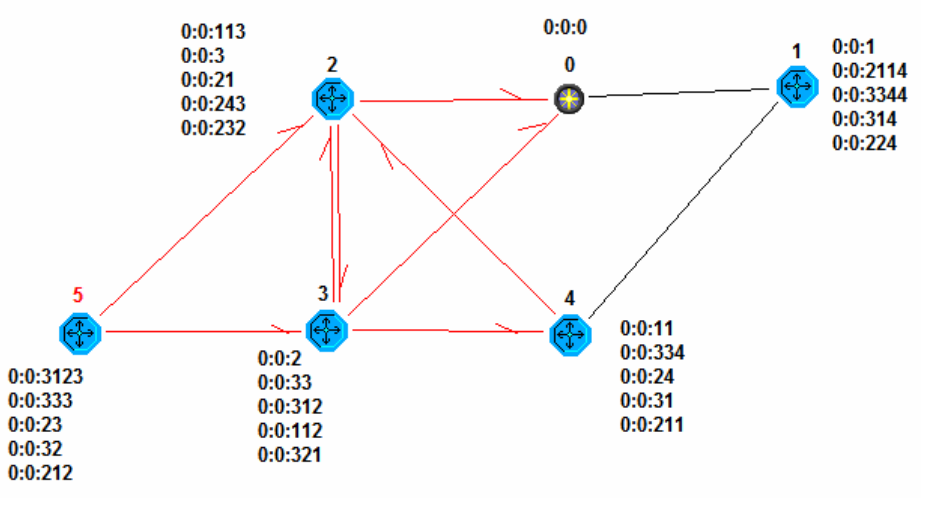


**Fig. 2.** Node 5 has five VIDs identifying five paths to the cluster head, node 0. A failure of one link in these paths would result in node 5 losing one or more paths, but because of the other VIDs node 5 would still have path(s) to node 0.

**Table 1.** Schedule generated by the proposed algorithm for the topology in Figure 2

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... |
| 20 | - | - | - | - | - | - |
| 21 | - | - | - | - | - | - |
| 22 | - | S0:0:211 (4) | - | - | R0:0:2114 (1) | - |
| 23 | - | - | S0:0:11 (4) | - | R0:0:113 (2) | - |
| 24 | - | S0:0:334 (4) | - | - | R0:0:3344 (1) | - |
| 25 | C | C | C | C | C | C |
| 26 | - | S0:0:31 (4) | - | R0:0:3123 (5) | R0:0:314 (1) | S0:0:312 (3) |
| 27 | - | - | - | S0:0:31 (4) | R0:0:312 (3) | - |
| 28 | - | - | S0:0:23 (5) | S0:0:11 (4) | R0:0:112 (3) | R0:0:232 (2) |
| 29 | - | - | S0:0:24 (4) | R0:0:333 (5) | R0:0:243 (2) | S0:0:33 (3) |
| 30 | - | - | R0:0:31 (4) | S0:0:32 (5) | S0:0:3 (2) | R0:0:321 (3) |
| 31 | - | - | R0:0:32 (5) | R0:0:334 (4) | S0:0:33 (3) | S0:0:3 (2) |
| 32 | - | S0:0:24 (4) | R0:0:33 (3) | S0:0:3 (2) | R0:0:244 (1) | - |
| 33 | - | - | R0:0:212 (5) | R0:0:24 (4) | S0:0:2 (3) | S0:0:21 (2) |
| 34 | C | C | C | C | C | C |
| 35 | - | - | R0:0:211 (4) | - | S0:0:21 (2) | - |
| 36 | R0:0:3 (2) | - | S0:0:0 (0) | R0:0:23 (5) | - | S0:0:2 (3) |
| 37 | - | - | S0:0:2 (3) | R0:0:21 (2) | - | - |
| 38 | R0:0:2 (3) | R0:0:11 (4) | - | S0:0:0 (0) | S0:0:1 (1) | - |
| 39 | R0:0:1 (1) | S0:0:0 (0) | - | - | - | - |

There are some specific benefits to scheduling synchronously to all established links because while on-demand or asynchronous scheduling can be more efficient in media utilization and latency (by allowing fewer slots per frame,) this is only true if the traffic patterns meet certain characteristics. If the data sessions are very predictable and regular, and last for relatively longer periods, and also if the links within the route last reasonably long relative to the schedule set-up time, then on-demand scheduling is a good design choice. However, if the data sessions are short, irregular, bursty and/or difficult to predict ahead of time, or if the links tend to break too often, then the overhead and latency involved in establishing such routes can become very high. Under these more difficult but realistic traffic and network conditions, synchronously allocating the slots is a better design choice, especially when the scheduling algorithm allows optimization to reduce latency.

Table 1 shows the schedule that was generated for the topology shown in Figure 2, under one simulation run. This figure only shows the second half of the schedule, slots 20-39, which is the up-stream slots; the first half is a mirror-image of this, with all slots in reverse order and with senders and receivers also reversed. The first row indicates the UID of each node in the cluster. Any slots marked 'C' are control slots, which are used for route advertisements only. The 'S' or 'R' in these entries indicates if it is a send or receive slot for that column's node, the n:n:n number is the VID of the remote-end of that link, with the format [subnet]:[cluster]:[path], as discussed in [2]. The final number, in parenthesis, is the Unique ID of the remote-end of the link. For example, in the column '5', row '29' contains "S0:0:33 (3)," which would be a send slot from node 5 to VID 0:0:33 (which is a VID of node 3,) scheduled for slot 29.

In this table, it can be seen that the restriction discussed above is satisfied: all up-stream links must wait until they have received from their down-stream descendants before they can be scheduled to transmit further up the tree. The highlighted links demonstrate another property of this algorithm: there is no requirement that all 3-hop links be scheduled before all 2-hop links (and so on), simply that the slots be in-order along their own path. This is due to the FCFS component of the algorithm, discussed above, but also has the benefit of reducing the amount of messaging needed to keep all nodes in the same state.

In this example, an average of 1.56 links are scheduled per slot among the non-empty data slots, demonstrating this slot re-use. (Note that this algorithm does not address how to find the minimum frame size, and therefore leaves some slots unused in most cases. All simulations here use a fixed frame size of 40 slots, with 4 being control slots.)

## 3 Expected Latencies

Let $L$ represent the total latency.

Let $Q_t$ represents the time where a packet is queued at the MAC, waiting for the time of the correct slot.

Let $F_t$ represent the length of a frame, and $S_t$ represent the length of a slot (including any guard-time)

Let $S_n$ represent the number of slots in frame.

Let $n$ represent the number of clusters the path travels through.

Let $h$ represent the number of hops in a route, and $h_i$ represents the number of hops within the $i^{th}$ branch.

Let $P_d$ represent the propagation delay.

Let $T_d$ equal the transmission delay, the time needed to transmit the complete packet.

Let $b_{rn}$ equal the number of branches traversed in cluster $n$.

Let $T_t$ represents the transit time, the time that the packet spends actually traversing the path.

We now derive the latencies for several cases below. In all of the cases discussed, it is assumed that none of the links are over-subscribed, and that all packets are completely received before the next slot begins. Under significant traffic loads, the queuing delays for any specific transmit slot can be significant, as discussed in detail in [4], but the application of queuing theory to TDMA systems is already well studied, and the simulations we ran were designed such that the arrival rate was low enough that this queuing time for the transmit slots did not have a significant effect. We first start with a linear topology case and then go on to the more complex topologies.

### 3.1 Linear Topology

Figure 3 shows the case which gives the lowest possible latencies, since all of the hops will be scheduled not only in-order, but also without any other slots scheduled between these slots. This will give equation 2.

$$Tt = St*(h-1) + Pd + Td \tag{2}$$

Ignoring any time waiting for the first needed slot to begin (since that is the Qt term,) the first node will send at t=0, the second node can send at t=St, and so on until the packet travels up the path to the cluster head. Assuming the packet is going upstream, the CHs one-hop child will start to transmit along this final hop at t=St*(h-1), and the CH will receive it at t = St*(h-1) + Pd + Td.

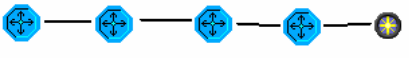

**Fig. 3.** A linear intra-cluster topology, of a one-way path

### 3.2 Intra-cluster, One Branch

Figure 4 presents a more realistic case, where the scheduled slots will still be guaranteed to be in-order by the scheduling algorithm, but the path may have slots for other paths in-between, as shown in the example in table 1. The latency in this case is shown in Equation 3a, with the transit time Tt expanded in 3b.

$$L = T_t + Q_t \tag{3a}$$

$$T_t = ½ *F_t * P_r + P_d + T_d \tag{3b}$$

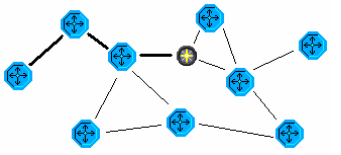

**Fig. 4.** A non-linear intra-cluster topology, traveling a one-way path

Let $P_r$ represent the path ratio, which equals (last used slot on path – first used slot on path) / ( ½ $S_n$). Note that $P_r$ of 1 corresponds to ½ of the frame: this is because any single branch can be traversed with a maximum $T_t$ of ½ $F_t$. In the schedule in Table 1, for example, $P_r$ for the highlighted path would be (39-23)/20 = 0.8, although this can vary significantly. As discussed for Figure 3, the $P_d$ and $T_d$ terms only have any affect on the last hop.

Note that if the path is only one hop, the total latency L will be the same as for the random scheduler, meaning that for one-hop paths, the maximum value of $P_r$ is equal to its minimum.

This expression 3b for $T_t$ can also represent the best case latency for a case such as section 3.1, as shown in equation 4.

$$½ * F_t * P_r = S_t (h-1) \tag{4}$$

$$½ * F_t * ((h-1) / ( ½ * S_n)) = S_t (h-1) \tag{4a}$$

$$½ * F_t * ((h-1) / ( ½ * S_n)) = (F_t / S_n) * (h-1) \tag{4b}$$

Equation 4 presents the initial claim, by setting 3b equal to 2 and subtracting the Pd and Td terms, which would indicate that the general equations 3a and 3b also apply to the linear case in section 3.1. Equation 4a substitutes $P_r = (h-1) / ( ½ * S_n)$, which is true if all the slots are sequential, by the earlier definition of $P_r$. Equation 4b substitutes $S_t = F_t / S_n$, which follows from these terms' definitions, above. After 4b is reduced slightly, it becomes clear that these terms are equal, and equation 4 is confirmed.

### 3.3 Intra-cluster, Two Branches

Figure 5 is an expanded case from section 3.2, where the packet must traverse up one branch and down another (although not necessarily through the CH.) The $T_t$ term must now include both of these paths and any delay between these paths, more than doubling it compared to Figure 4.

$$T_t = T_{t1} + T_{t2} + W_t \tag{5a}$$

$$T_t = ½ * F_t (P_{r1} + P_{r2}) + P_{d1} + P_{d2} + T_{d1} + T_{d2} + W_t \tag{5b}$$

$$P_r = (P_{r1} + (T_{d1} + P_{d1} + W_t)/(½F_t) + P_{r2}) = (\text{“last slot”} - \text{“first slot”})/( ½ S_n) \tag{5c}$$

$$T_t = ½F_t * P_r + T_{d2} + P_{d2} \tag{5d}$$

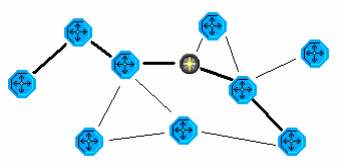

**Fig. 5.** A non-linear intra-cluster topology, traveling a two-way path

Specifically, Equations 5a through 5c show the derivation of the total $T_t$, shown in equation 5d.

Let $W_t$ equal the wait time between branches. This is the time between when a packet is received at the end of one branch, and when it is transmitted at the start of the next branch. Wt will have a value of $S_t - (P_d + T_d) + j*S_t$, where j is the number of unused slots between these branches. Considering this term, it is clear that 5a represents the sum of the Tt of both branches and any time between them, eq. 5b expands this, and eq. 5c uses this expanded expression to show that it is equal to the definition of $P_r$.

Note that in eq. 5d, $P_r$ is in the range: $((h_0\text{-}1)+(h_1\text{-}1)+1)/(\frac{1}{2} S_n) <= P_r <= 2$, since the term $(T_{d1} + P_{d1} + W_t) >= St$

Note that while eq. 5a through 5d assume the path will travel through the CH, the results are true even if the path travels only partially up one branch before switching to the other branch and traveling down (which is possible if these two branches have a point of overlap downstream of the CH.) However in this case $P_{r1}$ and $P_{r2}$ will be shorter, but Wt will be longer, and $P_r$ for the path overall will still be ("last slot" – "first slot") / ($\frac{1}{2} S_n$): which slots are used or unused between the time of the first and last needed slots has no affect on the latency unless congestion is considered. It is still beneficial to switch branches at downstream nodes instead of the CH if possible, for several other reasons such as saving power, reducing traffic load on the CH, etc, but (ignoring queuing delays) none of these will directly affect the end-to-end latency when using this scheduling algorithm.

### 3.4 Neighboring Clusters, One Branch Each

The inter-cluster cases are extensions of the intra-cluster cases. Figure 6 shows a case where a packet must travel down one branch in the source node's cluster, and must travel up another branch in the destination node's cluster. This is similar to the case discussed in 3.3, except in this case the packet must travel downstream to a border node, then back upstream to the destination in the other cluster (rather than up then down, in figure 5.) The equations in eq. 5 still hold, however in practice $W_t$ will tend to be larger, (resulting in a larger $P_r$,) because the frame fills from the outside, leaving the unused space in the middle.

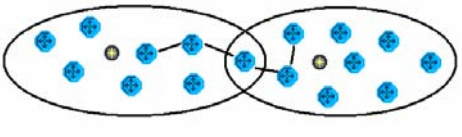

**Fig. 6.** Neighboring clusters, one branch each

### 3.5 Neighboring Clusters, Two Branches Each

Figure 7 presents a more general case based on section 3.4, where this time the path is up to the CH, down to a border node, up to the other CH, and then down to the destination node. The general equations 3a and 3b still hold, however we need to determine $P_r$. Its maximum value will be 4, since 4 branches are traversed in the path, and its minimum value will be eq. 6, since the number of hops in the first and last branches can affect the latency, while the intermediate branches will have no affect. This is because for these branches, as $P_r$ decreases Wt will increase, since it's the first and last slots of the entire path which determine $P_r$ by the more general expression ("last slot" – "first slot") / (½ $S_n$)

$$P_r >= S_t*(h_0-1) + 2 + S_t*(h_3-1) \quad (6)$$

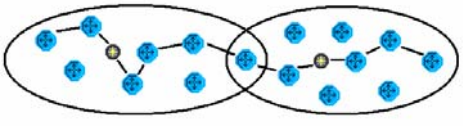

**Fig. 7.** Neighboring clusters, two branches each

### 3.6 Many Clusters, One Branch in Each End Cluster

Figure 8 can be derived simply from the results for sections 3.4 and 3.5: the first and last branches will affect the $P_r$, and the intermediate clusters can have any schedules, since each of these clusters will take one frame to traverse under any valid schedule. Thus, equation 7.

$$P_r >= S_n*(h_0-1) + 2*(n-2) + S_n*(h_n-1) \quad (7)$$

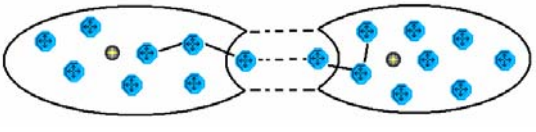

**Fig. 8.** Many clusters, single branch in each end cluster

### 3.7 Many Clusters, Two Branches in Each End Cluster

The results for Figure 9 can again be derived from those of sections 3.5 and 3.6. Simply, since there are two more total branches to traverse compared with Figure 7, $P_r >= (h_0)/ ( ½ S_n) + 2*(n-1) + (h_n-1)/ ( ½ S_n)$.

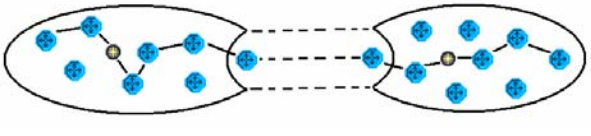

**Fig. 9.** Many clusters, two branches in each end cluster

### 3.8 Summary of Analysis

The general expression for the latency is presented in equation 3a, while eq. 3b expands the $T_t$ term and first includes $P_r$ which is the only term in the latency calculation which depends on the topology or the schedule the algorithm creates. All of the results for $P_r$ are shown in Table 2, while eq. 8a and 8b gives a general expression for the $P_r$ ranges for intra- and inter-cluster cases, respectively. (Note that the asterisk below indicates that $P_r$ max. = $P_r$ min. for one-hop routes, as indicated in the discussion in section 3.2.)

$$(h_0 + h_n - 1)/(\tfrac{1}{2} S_n) <= P_r <= b_{r0} \tag{8a}$$

$$(h_0 + h_n - 1)/(\tfrac{1}{2} S_n) + 2(n-2) + (b_{r0}-1) + (b_{rn}-1) <= P_r <= 2(n-2) + b_{r0} + b_{rn} \tag{8b}$$

In general, these analytical results show that the total latency of any packet along its path is determined by a) the arrival time of the packet, relative to the position in the frame (the Qt term), b) the number of intra-cluster branches the route includes (1 or 2 per cluster, in all cases), c) the number of hops of the first and last branches (which affects the $P_r$ term), d) the number of intermediate slots along the first and last branches, (which also affects the $P_r$ term, and this is primarily determined by the 'width' of the meshed-tree which MMT creates), e) the number of slots per frame, f) the length of each slot, including its guard time, and g) the transmission and propagation delays of the last hop.

**Table 2.** Summary of the $P_r$ values for each path type

| path type | $P_r$ minimum | $P_r$ maximum |
|---|---|---|
| intra-cluster, one branch | $(h-1)/(\tfrac{1}{2} S_n)$ | 1* |
| intra-cluster, two branch | $(h_0)/(\tfrac{1}{2} S_n) + (h_1-1)/(\tfrac{1}{2} S_n)$ | 2 |
| meighboring clusters, one branch | $(h_0)/(\tfrac{1}{2} S_n) + (h_1-1)/(\tfrac{1}{2} S_n)$ | 2 |
| meighboring clusters, two branch | $(h_0)/(\tfrac{1}{2} S_n) + 2 + (h_3-1)/(\tfrac{1}{2} S_n)$ | 4 |
| arbitrary inter-cluster, one branch | $(h_0)/(\tfrac{1}{2} S_n) + 2*(n-2) + (h_n-1)/(\tfrac{1}{2} S_n)$ | n*2 - 2 |
| arbitrary inter-cluster, two branch | $(h_0)/(\tfrac{1}{2} S_n) + 2*(n-1) + (h_n-1)/(\tfrac{1}{2} S_n)$ | n*2 |

## 4 Simulation Results

### 4.1 Static, Linear

This algorithm was simulated with a linear topology, as discussed in 3.1. While this is not a topology that is likely to occur in most real-world applications, it does provide a convenient base-case to confirm the analytical results. Indeed, these end-to-end latency results very closely match the analytical results in Table 2, specifically with $P_r = (h-1)/(\tfrac{1}{2} S_n)$, its minimum value, and with $Q_t = \tfrac{1}{2} f$, due to the random arrival times of packets from the higher layers. These results are from a batch of 7 OPNET simulation runs, each with different random seeds, using Sn = 40, and St = 5ms. With 100km spacing between the nodes, Pd = 0.333ms and Td = 0.03246ms (a 3.2Kbit

packet at 100Mbps), although these values were small enough to not be included in the "expected avg. ETE" values. The percent difference is small enough to be accounted for in the random variation of the $Q_t$ term.

Table 4 shows what the performance of the random scheduler would be, given the same frame length, demonstrating the comparison of optimized ETE / random ETE. This demonstrates how for single hop routes there is no benefit, but even for two hop routes the latency is reduced almost by half. For a 5-hop route, the latency is reduced to only 24% of the random scheduler's result, and for long inter-cluster routes this affect would be even more dramatic.

**Table 3.** Simulation results for the intra-cluster topology, as shown in Figure 3

| number of hops | expected avg. ETE | simulated avg. ETE | % difference |
|---|---|---|---|
| 5 | 0.120 | 0.11740 | 2.17% |
| 4 | 0.115 | 0.11281 | 1.90% |
| 3 | 0.110 | 0.11048 | 0.44% |
| 2 | 0.105 | 0.10382 | 1.12% |
| 1 | 0.100 | 0.09751 | 2.49% |

**Table 4.** Comparison with a random scheduler

| number of hops | expected avg. ETE | random scheduler expected avg. ETE | optimized vs random | avg. ETE with worst case $P_r$ | worst case $P_r$ vs. random |
|---|---|---|---|---|---|
| 5 | 0.120 | 0.500 | 24.00% | 0.200 | 40.00% |
| 4 | 0.115 | 0.400 | 28.75% | 0.200 | 50.00% |
| 3 | 0.110 | 0.300 | 36.67% | 0.200 | 66.67% |
| 2 | 0.105 | 0.200 | 52.50% | 0.200 | 100.00% |
| 1 | 0.100 | 0.100 | 100.00% | 0.100 | 100.00% |

In the real world, the benefit would not be as large, because the topology would typically be non-linear, meaning that the value of $P_r$ would not likely be at its minimum, and because a random scheduler would allow somewhat shorter frames to be used, however even considering these affects the benefit of this scheduler would still be significant. Even if $P_r$ were its worst case, the performance would match that of a random scheduler for one- and two-hop routes, and would still exceed it significantly for 3-hop or longer routes.

### 4.2 Non-linear, Static Case

Figure 11 shows the latency results from one minute of a simulation run of the scenario shown in Figure 10. Here, node 5 is sending data to the cluster head, node 0. Node 5 sends along the shortest routes it has available, in this case the 2-hop routes 0:0:23 and 0:0:42. (In this case it uses 0:0:23, since that VID was obtained first.) It can be seen in figure 9 that the average latency L is about 0.125, this would correspond to a Tt of about 0.025 if it is assumed that Qt is about 0.1.

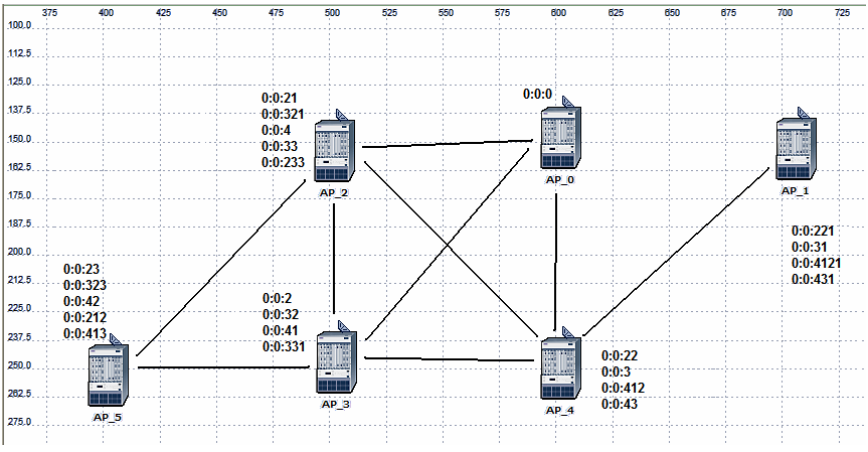


**Fig. 10.** The same node arrangement as in figure 2, but with a different seed and different routes generated

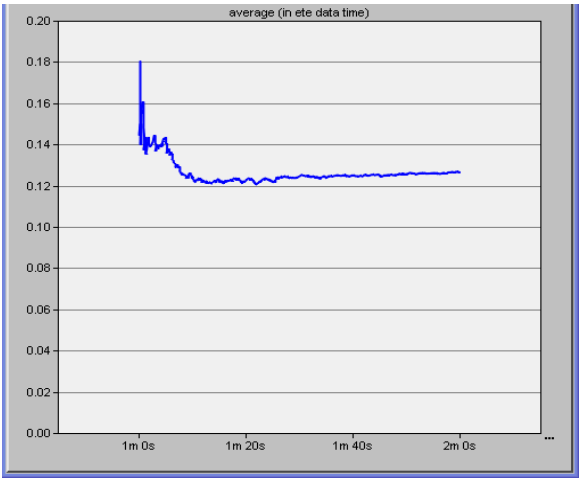


**Fig. 11.** Average latency for data packets

Examining the schedule in Table 5, more precise values for Qt and Tt can be obtained. It can be seen that the $P_r$ value must be (38-35)/20 = 0.15, giving a Tt value of 0.1*0.15(s) + 0.333(ms) + 0.03246(ms) = 0.01536546, if Pd = 0.333ms and Td = 0.03246ms. Given this Tt of about 0.015, Qt must be about 0.11 for this seed value.

Note that an average of 1.46 links are scheduled per slot among the non-empty data slots, which is similar to the result shown in Table 1.

**Table 5.** Schedule from the simulation result, corresponding to figures 10 and 11

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... |
| 20 | - | - | - | - | - | - |
| 21 | - | - | - | - | - | - |
| 22 | - | - | R0:0:331 (3) | S0:0:33 (2) | - | - |
| 23 | - | - | S0:0:3 (4) | - | R0:0:33 (2) | - |
| 24 | - | S0:0:43 (4) | - | - | R0:0:431 (1) | - |
| 25 | C | C | C | C | C | C |
| 26 | - | - | R0:0:43 (4) | - | S0:0:4 (2) | - |
| 27 | - | S0:0:412 (4) | - | R0:0:323 (5) | R0:0:4121 (1) | S0:0:32 (3) |
| 28 | - | - | S0:0:23 (5) | R0:0:412 (4) | S0:0:41 (3) | R0:0:233 (2) |
| 29 | - | - | - | R0:0:413 (5) | - | S0:0:41 (3) |
| 30 | - | - | S0:0:32 (3) | R0:0:321 (2) | - | - |
| 31 | - | - | R0:0:42 (5) | S0:0:3 (4) | R0:0:32 (3) | S0:0:4 (2) |
| 32 | - | S0:0:3 (4) | R0:0:41 (3) | S0:0:4 (2) | R0:0:31 (1) | - |
| 33 | - | - | R0:0:212 (5) | - | - | S0:0:21 (2) |
| 34 | C | C | C | C | C | C |
| 35 | - | S0:0:22 (4) | - | R0:0:23 (5) | R0:0:221 (1) | S0:0:2 (3) |
| 36 | R0:0:4 (2) | - | S0:0:0 (0) | R0:0:22 (4) | S0:0:2 (3) | - |
| 37 | R0:0:3 (4) | - | S0:0:2 (3) | R0:0:21 (2) | S0:0:0 (0) | - |
| 38 | R0:0:2 (3) | - | - | S0:0:0 (0) | - | - |
| 39 | - | - | - | - | - | - |

### 4.3 Mobile Case

Figures 12 and 13 show the mobile topology examined, and the corresponding test results for it. The paths taken by the nodes in Figure 12 are a mix of clockwise (green/light) and counter-clockwise (red/dark), with each with path having a speed of either 800, 1000, or 1200km/h. This scenario of counter rotating orbits causes high dynamic route changes and stresses the routing mechanism. Node 0 is the cluster head, all other nodes are clients and all were sending data packets at random times, averaging 5 packets per node per second. Figure 13 shows a latency L of about 0.102 seconds. If it is assumed that Qt is 0.1 seconds, then Tt would be 0.01 seconds.

As with Table 5, the schedule information in Table 6 allows a more precise analysis of the values of Tt and Qt. All nodes except for node 5 obtain one-hop routes to the CH, meaning that their Tt values are (Pd + Td). If we assume that Pd = 0.333ms and Td = 0.03246ms, as was the case for Figure 10's topology, Tt for these four nodes would be 0.3654 ms, and most of these nodes would have Pd, and therefore Tt, lower than this due to most of these distances being less than the 100km distance from Figure 10. This leaves node 5 with a two-hop route, with $P_r = 2/40$, for a total Tt of 0.8654ms, using the Pd and Td values from previously. This gives the average Tt for all of these nodes as 1.3654ms, giving Qt of 0.1006 seconds.

Again, also note that an average of 1.63 links are scheduled per slot among the non-empty data slots, again showing the high amount of slot re-use.

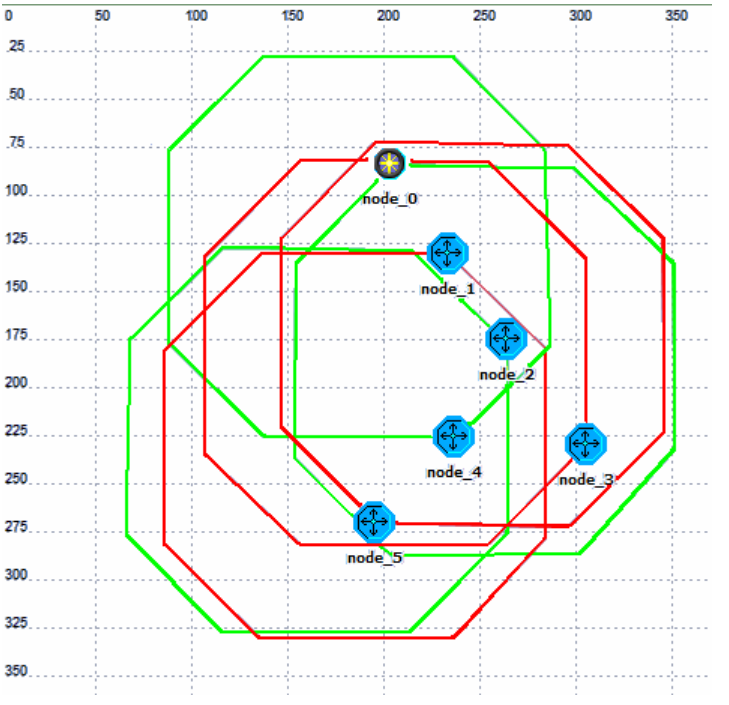


**Fig. 12.** Scenario used for the mobile test

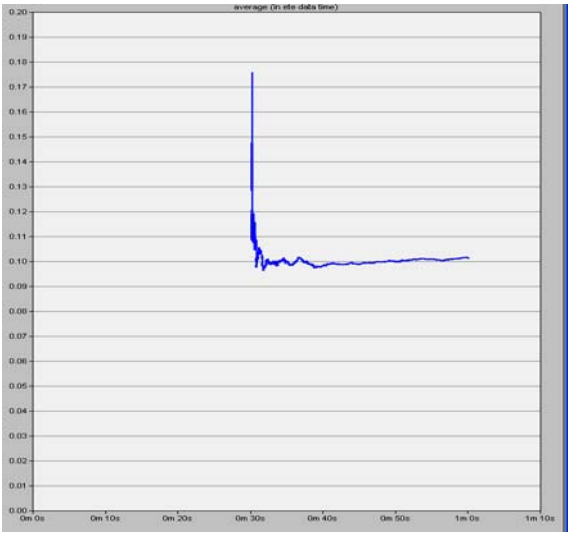


**Fig. 13.** Average latency for data packets

**Table 6.** Schedule for the simulation result corresponding to figures 12 and 13

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... |
| 20 | - | - | - | - | - | - |
| 21 | - | - | - | - | - | - |
| 22 | - | - | - | - | - | - |
| 23 | - | - | - | - | - | - |
| 24 | - | - | - | - | - | - |
| 25 | C | C | C | C | C | C |
| 26 | - | - | - | - | - | - |
| 27 | - | - | - | - | - | - |
| 28 | - | R0:0:334 (4) | - | - | S0:0:33 (1) | - |
| 29 | - | - | - | S0:0:4 (4) | R0:0:42 (3) | - |
| 30 | - | S0:0:4 (4) | - | - | R0:0:41 (1) | - |
| 31 | - | S0:0:32 (5) | S0:0:2 (3) | R0:0:24 (2) | - | R0:0:321 (1) |
| 32 | - | R0:0:14 (4) | R0:0:112 (5) | - | S0:0:1 (1) | S0:0:11 (2) |
| 33 | - | S0:0:3 (2) | R0:0:33 (1) | S0:0:32 (5) | - | R0:0:322 (3) |
| 34 | C | C | C | C | C | C |
| 35 | R0:0:4 (4) | R0:0:13 (3) | R0:0:32 (5) | S0:0:1 (1) | S0:0:0 (0) | S0:0:3 (2) |
| 36 | - | - | R0:0:31 (4) | - | S0:0:3 (2) | - |
| 37 | R0:0:3 (2) | R0:0:12 (5) | S0:0:0 (0) | - | - | S0:0:1 (1) |
| 38 | R0:0:2 (3) | R0:0:11 (2) | S0:0:1 (1) | S0:0:0 (0) | - | - |
| 39 | R0:0:1 (1) | S0:0:0 (0) | - | - | - | - |

## 5 Conclusion

The analysis provides a bounded range of latencies for all possible topologies when using the proposed system. This latency range is a function of several factors which were described, but primarily depend on the number of clusters the route passes through, the number of branches traversed in the first and last clusters (1 or 2) and the number of hops along the first and last branches traversed, as well as the arrival time of the packet within the frame relative to the first node's transmit slot and the structure of the frame (number of slots, slot duration, etc.) This analysis is confirmed through simulation.

As the analysis and simulation results demonstrate, the latencies over multi-hop routes are significantly lower than a non-optimized, random scheduler. For the best case schedule, the proposed scheduler outperforms a non-optimized scheduler for all routes longer than one hop, and for the worse case schedule it outperforms the non-optimized scheduler for all routes longer than two hops, assuming the frame structure is consistent across all cases. All available results and analysis indicate the benefit of this system over using a non-optimized scheduler for this application, and the benefits of a TDMA MAC, electrically steerable directional antennas, and the MMT algorithm over their possible alternatives is also discussed.

## References

[1] Yu, J., Chong, P.: A Survey of Clustering Schemes for Mobile Ad Hoc Networks. IEEE Communications Surveys 7(1), 32–48 (2005)
[2] Pudlewski, S., Shenoy, N., Al-Mousa, Y., Yin, P., Fischer, J.: A hybrid Multi Meshed Tree routing protocol for wireless ad hoc networks. In: 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, pp. 635–640 (2008)
[3] Shenoy, N., Yin, P.: Multi-meshed tree routing for internet MANETs. In: 2nd International Symposium on Wireless Communication Systems, pp. 145–149 (2005)
[4] Rom, R., Sidi, M.: Multiple access protocols - Performance and Analysis, 2nd edn. Springer, New York (1990)
[5] Subramanian, A., Das, S.: Addressing deafness and hidden terminal problem in directional antenna based wireless multi-hop networks. In: 2nd International Conference on Communication Systems Software and Middleware, COMSWARE 2007, pp. 1–6 (2007)
[6] Abbasi, A.A., Younis: A Survey on Clustering Algorithms for Wireless Sensor Networks. Computer Communications 30(14-15), 2826–2841 (2007)
[7] Nelson, R., Kleinrock, L.: Spatial TDMA: A Collision-Free Multihop Channel Access Protocol. IEEE Transactions on Communications 33(9), 934–944 (1985)
[8] Vergados, D.D., Vergados, D.J., Douligeris, C., Tombros, S.L.: QoS-aware TDMA for end-to-end traffic scheduling in ad hoc networks. IEEE Wireless Communications 13(5), 68–74 (2006)
[9] Wu, T., Biswas, S.: A Self-Reorganizing Slot Allocation protocol for multi-cluster sensor networks. In: Proceedings of the 4th international Symposium on information Processing in Sensor Networks, pp. 309–316 (2005)
[10] Jawhar, I., Wu, J.: Resource Allocation in Wireless Networks Using Directional Antennas. In: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications, pp. 318–327 (2006)