An Overlay MAC Layer for 802.11 Networks

Ananth Rao
UC Berkeley

Ion Stoica
UC Berkeley

Abstract

The widespread availability of 802.11-based hardware has made it the premier choice of both researchers and practitioners for developing new wireless networks and applications. However, the ever increasing set of demands posed by these applications is stretching the 802.11 MAC protocol beyond its intended capabilities. For example, 802.11 provides no control over allocation of resources, and the default allocation policy is ill-suited for heterogeneous environments and multi-hop networks. Fairness problems are further exacerbated in multi-hop networks due to link asymmetry and hidden terminals. In this paper, we take a first step towards addressing these problems *without* replacing the MAC layer by presenting the design and the implementation of an *Overlay MAC Layer* (OML), that works on top of the 802.11 MAC layer. OML uses loosely-synchronized clocks to divide the time in to equal size slots, and employs a distributed algorithm to allocate these slots among competing nodes. We have implemented OML in both a simulator and on a wireless testbed using the Click modular router. Our evaluation shows that OML can not only provide better flexibility but also improve the fairness, throughput and predictability of 802.11 networks.

1 Introduction

In recent years, the popularity of the 802.11 protocol has made it the de facto choice for developing and deploying a multitude of wireless networks and applications. In addition to the traditional model of a last-hop wireless link to an access point, 802.11 networks are used to setup wireless infrastructures in corporate networks, long-haul links using directional antennae in rural areas [31], and more recently multi-hop networks for broadband Internet access, *e.g.,* rooftop or mesh networking [5, 20].

Despite making deployment easier, the 802.11 protocol does pose serious limitations in addressing the different demands of these emerging applications. The 802.11 MAC protocol was carefully engineered for the wireless LAN environment [27] and many of the underlying assumptions may not hold in the new environments. Several problems have been reported in earlier research [23]. We illustrate more problems through a careful performance study of the MAC using statistics collected from the device driver and packet-sniffing tools [1, 2]. In particular, we show that link asymmetry or hidden terminals can either cause (a) poor fairness, sometimes even shutting off all flows through a node or (b) excessive collisions, leading to poor performance. In addition, we consider several scenarios where the default allocation of the medium by 802.11 is far from being optimal.

Two main approaches have been proposed in the literature to address these problems. The first approach is to build workarounds in the routing or transport layers to avoid the cases where the MAC layer performs badly [5, 20, 22, 38]. The second approach focuses on replacing the MAC layer with new protocols [10, 16] and standards such as 802.16 (WiMax) and 802.11e. The first approach is more easily deployable as the functionality of both the routing and transport layers is fully implemented in software, so it is relatively easy to modify. However, as we will show in this paper, the 802.11 MAC layer suffers from certain limitations, such as unfairness due to asymmetric interference, that cannot be fully addressed through changes only to the higher layers. In contrast, the second approach is far more powerful since modifying the MAC layer can directly address all the 802.11 limitations, but it is much harder to implement and experiment with. The MAC layer is implemented partly in hardware, partly in firmware, and partly in the device driver of the Network Interface Card (NIC). Thus, changing the MAC layer can require hardware and firmware changes, a highly expensive proposition.

In this paper, we propose a third approach that combines power of changing the MAC layer and the ease of deployability of modifying only the higher layers. In particular, we propose the design of an Overlay MAC layer (OML) on *top* of the 802.11 MAC layer. OML performs access control and scheduling, and does not require any changes to the 802.11 MAC hardware or the 802.11 standard. Our approach is inspired by the success of the "Overlay" networks, which have been used in the past few years to study new network protocols and implement new functionality without any modifications to the underlying IP layer. In the context of the MAC layer, using an overlay offers the following three advantages. First, it provides an immediately useful piece of software which can be used in 802.11 networks while waiting for newer standards to become more widespread. Second, the additional flexibility of having the MAC layer in software allows better integration with routing and application requirements. Third, it allows research on new protocols to be conducted on any of the numerous, already-deployed 802.11 testbeds. However, these advantages do not come for free. Like many overlay approaches, OML suffers some additional overhead compared to implementing the same changes at the MAC layer. In addition, the design of OML is limited by the interface exposed by the 802.11 MAC layer. For example, OML cannot carrier sense the communication channel since 802.11 network cards do not typically export the channel status to higher layers. Despite such limitations, we believe that the advantages of the our overlay approach make it a valuable alternative to modifying the MAC layer.

To address the previously mentioned limitations of the 802.11 MAC layer, OML uses loosely synchronized clocks to divide the time in equal size slots, and then uses a distributed algorithm to allocate these slots across the competing nodes. In addition to preventing unfavorable interaction between senders at the underlying MAC layer, OML also allows users to implement application-specific resource allocation in the same way overlay networks allow application-specific routing. The slot allocation algorithm of OML, called Weighted Slot Allocation (WSA), implements the weighted fair queueing policy [19], where each of the competing nodes that has traffic to send receives a number of slots proportional to its weight.

The main contribution of this paper is the architecture and implementation of the Overlay MAC Layer. We believe this is the first time such an approach has been proposed to improve the performance and flexibility of the MAC layer. Because of the ease of deployability of the overlay approach, we are able to demonstrate these improvements through a real implementation of OML on a testbed. OML not only offers better flexibility than the 802.11 MAC layer, it also improves throughput and predictability by minimizing losses due to contention.

The rest of the paper is organized as follows. Section 2 presents the related work, and Section 3 uses experimental results to motivate the need for an overlay MAC protocol. Section 4 describes the challenges in designing OML. Section 5 describes the hardware and software of our testbed. Sections 6 and 7 present simulation and experimental results. Finally, Section 8 discusses the open issues and limitations, and Section 9 concludes the paper.

2 Related Work

802.11 MAC Limitations: Many researchers have reported problems with the 802.11 MAC protocol. Heusse *et al.* have described how senders with heterogeneous data rates can affect the system throughput adversely [23]. The Roofnet and Grid projects [4, 5] have reported a variety of problems with 802.11 multi-hop testbeds such as low throughput and unpredictable performance. We add to the set of the problems reported in these studies, by showing that *asymmetric* interference at the MAC layer can cause significant unfairness.

MAC Layer Improvements: A plethora of MAC protocols have been proposed to improve the predictability and the resource management capabilities of wireless networks. Many of these solutions use either sophisticated back-off protocols [12] or slot allocation algorithms [10, 11] to implement more flexible allocation policies such as Weighted Fair Queueing [19]. Other solutions achieve this goal by using a combination of both back-off and slot allocation algorithms [16,30]. All these protocols work at the MAC layer and assume full control over the hardware and the physical layer. In contrast, we assume that OML can use only the limited interface exposed by most 802.11 cards to control packet transmission.

Routing and Transport Layer Improvements: Some recent proposals use mechanisms above the MAC layer to improve the fairness of 802.11 networks. Gamrioza *et al.* [22] propose to explicitly limit the sending rate of the flows nodes, while Yi and Shakkottai [38] propose to implicitly limit the TCP flow rates by delaying the TCK ACKs at intermediate nodes to achieve fair bandwidth allocation. However, as we show in Section 3.2, controlling only the sending rate is not enough to avoid all undesirable interaction between competing flows.

Our work is complementary to several proposals that aim to improve the performance of 802.11 multi-hop networks. For example, Couto *et al.* [18] propose a new routing metric to improve network throughput. OML can be used in conjunction or integrated with such routing protocols to improve the throughput or other network metrics. Extremely Opportunistic Routing [13] (ExOR) proposes a modification to the 802.11 MAC so that any eligible node that receives a packet without errors can send an ACK and forward the packet. Since OML schedules transmissions based on the sender only, and not on the destination of a packet, we believe that OML can be used on top of ExOR to obtain better performance.

Distributed Slot Allocation: The Weighted Slot Allocation (WSA) algorithm that we propose in this paper is based on the Neighborhood-aware Contention Resolution (NCR) protocol, which was previously proposed by Bao and Garcia-Luna-Aceves [10, 11]. WSA extends NCR in two aspects. First, unlike OML, NCR assumes that the interference graph in a multi-hop network consists of isolated, easily identifiable cliques, *i.e.,* if node A interferes with B and B with C, then A interferes with C. Second, while NCR uses *pseudo-identities* to support integer weights, WSA can support arbitrary weights.

Overlay Networks: Our solution is similar in spirit to the overlay network solutions that aim to improve routing resilience and performance in IP networks [8, 9, 15, 36]. Overlay networks try to overcome the barrier of modifying the IP layer by employing a layer on top of the IP to implement the desired routing functionality. Similarly, OML runs on top of the existing 802.11 MAC layer, and its goal is to enhance the MAC functionality without changing the existing MAC protocols.

3 Motivation

In this section, we motivate our approach of building an Overlay MAC Layer for 802.11 networks. Briefly, our motivation stems from the following three factors:

1. Due to *asymmetric interaction* between flows and an *inflexible default allocation policy*, the efficiency and fairness of the 802.11 MAC suffer in a number of scenarios.
2. Solutions that only modify layers above the MAC, though applicable in certain cases, are of limited use in addressing certain undesirable interactions at the MAC layer.
3. The additional flexibility, the low cost and the immediate deployability of an overlay solution makes it an attractive alternative for developing a new MAC layer or modifying an existing one.

Next, we substantiate the first two factors by conducting experiments on a six node wireless testbed using 802.11a radios. In Section 3.1, we illustrate the limitations of 802.11 in the form of asymmetric interference and the inflexible default allocation policy, and in Section 3.2 we argue that these limitations cannot be fully addressed at layers above MAC. For more details on our testbed, please refer to Section 5.

3.1 Limitations of 802.11 MAC Layer

In this section, we illustrate two specific limitations of the 802.11 MAC protocol using simple experiments:

1. *Asymmetric interactions:* Interference between two flows either at the senders or at the receivers can cause one flow to be effectively shut-off.
2. *Sub-optimal default allocation:* As other researchers have pointed out [23], we show that the default allocation of the transmission medium by the MAC layer fails to meet the requirements of some applications.

3.1.1 Effect of Asymmetric Interaction

In this section, we take a closer look at the impact of interference on performance at the MAC layer. Interference has been often cited as being the cause for poor performance in other testbeds [5]. Here, we try to understand and quantify the effect of interference through experiments on a multi-hop testbed. To avoid any complex interaction between the MAC, routing, and transport layers, we restrict our experiments to two simultaneous 1-hop UDP flows. Figure 1 shows the location of machines in our testbed in relation to the floor plan of our office building. We also show the signal strength of each link in either direction as reported by the device driver of the wireless card. The topology of the testbed resembles a chain and we study how simultaneous transmissions along two links in the chain interfere with each other.

1. *Asymmetric sender interaction:* Because of asymmetry in radio propagation, one sender may be able to carrier sense transmissions from the other sender, but not vice-versa. This can lead to starvation of the first sender.
2. *Asymmetric receiver interaction:* Hidden terminal problems can completely shut-off flows in the network, particularly in the presence of other flows which do not experience similar problems.

Asymmetric Carrier Sense: We conduct our first set of experiments using broadcast packets only. This allows us to better understand the interaction between *senders*, as broadcast communication abstracts away the ACKs and packet retransmissions at the MAC layer. We make the two senders continuously send broadcast UDP packets and measure the sending rate of each sender averaged over 1 minute. The sending rate of each node depends only on whether it can carrier sense transmissions from the other node.

We conducted this experiment for each of the 15 pairs of nodes in our six node testbed. As expected, nodes that were far away (*e.g.,* nodes 1 and 5 in Figure 1) did not carrier sense each other and were able to simultaneously send at about 5.1 Mbps. When the nodes are close to each other (*e.g.,* nodes 2 and 3), they carrier sense each others transmissions, and hence share the channel capacity to send at about 2.5 Mbps each. However, in *three* cases we found than one of the nodes was transmitting at more than 4.5 Mbps, while the other was transmitting at less than 800 Kbps. The only possible explanation for this asymmetric performance is that one sender can carrier sense the other, but not vice-versa. This can impact all configurations of flows where both senders are active, irrespective of who the receivers are.

Asymmetric Receiver Interaction: Now, we study the interference at the receiver. To eliminate the effects of sender (carrier sense) interference, we only consider senders that are able to broadcast simultaneously at full rate. In these cases, we conducted experiments with each sender sending unicast UDP packets simultaneously to receivers within their communication range at the maximum rate. We found that depending on the configuration of the flows, either one or both receivers can be affected by interference. For example, when the flows from node 1 to 2 and from node 3 to 4 are simultaneously active, node 2 experiences interference from node 3, whereas the latter flow is not disrupted by transmissions from node 1. We

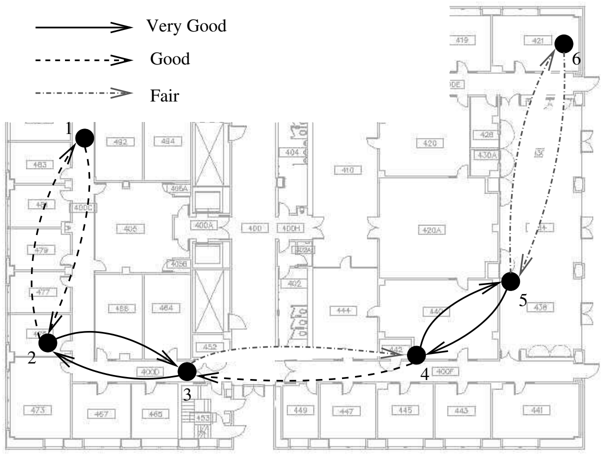


Figure 1: Location of the nodes and the signal strength of each link as reported by the driver in our multi-hop testbed

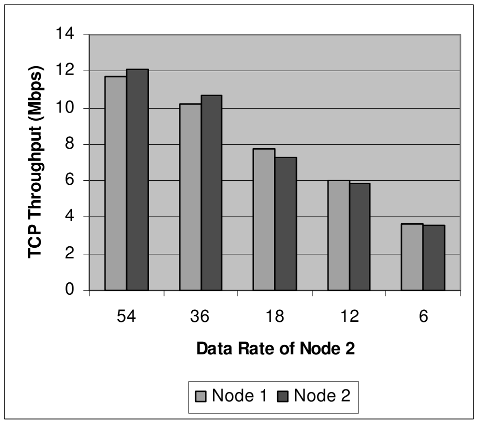


Figure 2: 802.11 throughput in the presence of heterogeneous data rate senders

found two such cases where the sending rate of affected sender drops by more than 60% due to the repeated back-off at the MAC layer triggered by retransmission time-outs. Furthermore, based on the statistics gathered from the device driver, we found more than 85% of the packets sent by this sender were not received at the destination. In the second case, both flows can experience problems due to interference, *e.g.,* when both nodes 1 and 3 are sending to node 2. This case illustrates the classic hidden terminal problem, where back-off at the MAC layer causes the sending rate of both senders to drop, which in turn reduces the loss rate of both flows to 35%. However, in this case, the channel utilization and hence the total system throughput drops by about 55%.

3.1.2 Sub-optimal Default Allocation

In this section, we show two examples where the default bandwidth allocation by the 802.11 MAC is far from ideal. These examples illustrate the need for a flexible allocation policy which can be controlled by applications.

Heterogeneous Transmission Rates: The 802.11 MAC allocates an equal number of *transmission opportunities* to every competing node. However, as shown in previous work [23], this fairness criterion can lead to a low throughput when nodes transmit at widely different rates.

We illustrate this behavior using a simple experiment comprising two heterogeneous senders connected to a single access point. We emulate heterogeneous senders by fixing the data-rate of transmissions from Node 1 to the access point at 54 Mbps and varying the data-rate of Node 2 between 6 Mbps and 54 Mbps. Figure 2 shows the average throughput of two TCP flows originated at the two nodes. This experiment shows that while the behavior is fair as nodes see equal performance irrespective of their sending rate, it hurts the overall system throughput. In particular, as the sending rate of Node 2 decreases from 54 Mbps to 6 Mbps, the total system throughput decreases from 24 Mbps to 7.2 Mbps. In addition, this behavior leads to poor predictability. For example, if Node 1 is the only active one, it will have a throughput of roughly 24 Mbps. However, when Node 2 starts transmitting at 6 Mbps, the throughput seen by Node 1 drops to 3.6 Mbps.

Several Flows Traversing a Node: In a multihop network, the fairness policy implemented by 802.11 can lead to poor fairness. This is because in an ad-hoc network, the fairness policy of 802.11 does not account for the traffic forwarded by a node on the behalf of other nodes. Ideally, we would like to provide higher priority to nodes that relay traffic from a lot of flows.

3.2 Need for a MAC Layer Solution

A natural question that arises is whether the limitations we described so far can be addressed at a layer above the MAC layer, such as the network or transport layer. Several proposals have tried to answer this question affirmatively [22, 38]. In a nutshell, these proposals estimate the capacity and interference patterns of the network, and then use this information to limit the sending rate (usually, employing token-buckets) of each node at the network layer. While these solutions may perform well in many scenarios, our experience suggests that in certain practical situations they fall short. For instance, consider the problem of fair allocation as defined in [22]. In order to achieve fair allocation, the following two requirements should hold:

1. Every node should access the medium for only its

fair share of time.

2. When a node is transmitting data, other nodes should not interfere with it.

By limiting the sending rate of each node according to its fair share, we can address the first requirement. However, this solution would work only if the underlying MAC layer satisfies the second requirement. Unfortunately, as discussed in Section 3.1, the 802.11 MAC fails to satisfy this requirement quite often. As an example, in the chain configuration of our testbed, we found that both the first and the third link of the chain taken in isolation had a loss rate of less than 5% and were able to support a TCP flow at close to the channel capacity of 4.6 Mbps. But when we started simultaneous flows on both links, one flow always received less than 100 Kbps whereas the other flow received in the excess of 4 Mbps. In order to mitigate this problem we tried rate limiting both TCP flows to 2.3 Mbps. In this case we found that the throughput of the first flow only improved to about 580 Kbps even though the other flow was only receiving 2.3 Mbps (as specified by the token bucket).

The inability of 802.11 to effectively address the second requirement suggests that a general solution to fully address all the 802.11 limitations requires changes to the MAC layer.

3.3 Advantages of an Overlay Solution

Given that the 802.11 MAC limitations cannot be fully addressed without changing the MAC layer, we propose the use of an Overlay MAC layer as an alternative to building a new MAC layer. We believe that the overlay approach offers several advantages such as low cost, flexibility and the possibility of integration with higher layers.

First, changing the MAC layer requires the use of expensive proprietary hardware, or waiting for a new standard and hardware to become available. In contrast, OML can be deployed using the existing 802.11 hardware.

Second, the fact that OML is implemented in software makes it easier to modify OML to meet the diverse requirements of the ever increasing spectrum of wireless applications [4, 5, 23, 31]. For example, in our OML implementation, we support service differentiation both at the flow and node granularity. In the case of a rooftop network [17], one could modify OML to take advantage of the relative stationarity of the link quality and interference patterns.

Finally, the software implementation of OML also enables us to have tighter integration between the link, network and transport layers. For example, Jain *et al* [24] show that it is possible to significantly increase the throughput of an ad-hoc network by integrating the MAC and routing layers. Another example is that the transport layer can provide information about the traffic type (*e.g.,* voice, ftp, web), and OML can use this information to compute efficient transmission schedules.

Of course, these advantages do not come for free. Fundamentally, OML incurs a higher overhead and it is more inefficient than a hardware implementation of the same functionality. Furthermore, OML is limited to using only the interface exposed by the 802.11 MAC layer as opposed to all the primitives that the hardware supports. We present some of the limitations of OML in more detail in Section 8.

4 Design

In this Section, we present our solution, an overlay MAC layer (OML), that alleviates the MAC layer issues described in Section 3. We first state our assumptions.

4.1 Assumptions

The primitives available for the design of OML are determined by the interface exposed by the device driver. For the sake of generality, in this paper we make minimal assumptions about this interface. In particular, we assume that

1. The card can send and receive both unicast and broadcast packets.
2. It is possible to set the wireless interface in promiscuous mode to listen to all transmissions from its 1-hop neighbors.
3. It is possible to disable the RTS-CTS handshake by correspondingly setting the RTS threshold.
4. It is possible to limit the number of packets in the card's queue to a couple of packets. This is critical for enabling OML to control packet scheduling because once the packets are in the card's queue, OML has no control over when these packets are sent out.

We note that these assumptions already hold or can be enforced, eventually by modifying the device drivers in most 802.11 cards.

While carrier-sensing is a very important primitive for the design of a MAC protocol, we do not assume that OML can use this primitive. This assumption reflects the fact that most 802.11 cards do not export the current status of the channel or the network allocation vector (NAV)[1] to the higher layers.

4.2 Solution

As noted in the previous section, the only control that OML can exercise over packet scheduling is *when* to send a packet to the network card. Once the packet is enqueued at the network card, OML has no control on when the packet is actually transmitted. Thus, ideally, we would like that when OML sends a packet to the network card, the network card transmits the packet immediately (or at least with a predictable delay).

To implement this idealized scenario, we propose a solution that aims to (a) limit the number of packets queued in the network card, and (b) eliminate interference from other nodes, which is the major cause of packet loss and unpredictability in wireless networks. Goal (a) can be simply achieved by reducing the buffer size of the network card.

To achieve goal (b), we synchronize clocks and we use a TDMA-like solution where we divide the time into slots of equal size l, and allocate the slots to nodes according to a weighted fair queueing (WFQ) policy [19]. We call this allocation algorithm the Weighted Slot Allocation (WSA) algorithm. WSA assigns a weight to each node, and in every interference region[2] allocates slots in proportion to nodes' weights. Thus, a node with weight two will get twice as many slots as a node with weight one in the same interference region. Only nodes that have packets to send contend for time slots, and a node can transmit only during its time slots. Since a time slot is allocated to no more than one node in an interference region, no two sending nodes will interfere with each other. This can substantially increase the predictability of packet transmission, and reduce packet loss at the MAC layer. However, in practice, it is hard to totally eliminate interference. In an open environment there might be other devices out of our control (*e.g.,* phones, microwave ovens), as well as other nodes that run the baseline 802.11 protocol which can interfere with our network. Furthermore, as we will see, accurately estimating the interference region is a challenging problem.

The reason we base WSA on the WFQ policy is because WFQ is highly flexible, and it avoids starvation. WFQ has emerged as the policy of choice for providing QoS and resource management in both network and processor systems [19, 28, 37]. However, note that WSA is not the only algorithm that can be implemented in OML; one could easily implement other allocation mechanisms if needed.

There are three questions we need to answer when implementing WSA: (a) what is the length of a time-slot, l, (b) how are the starting times of the slots synchronized, and (c) how are the times slots allocated among competing clients. We answer these questions in the next three sections.

4.2.1 Slot size

The slot size l is dictated by the following considerations:

1. l has to be considerably larger than the clock synchronization error.
2. l should be larger than the packet transmission time, *i.e.,* the interval between the time the first bit of the packet is sent to the hardware, and the time the last bit of the packets is transmitted in the air.
3. Subject to the above two constraints, l should be as small as possible. This will decrease the time a packet has to wait in the OML layer before being transmitted and will decrease the burstiness of traffic sent by a node.

In our evaluation, we chose l to be the time it takes to transmit about 10 packets of maximum size (*i.e.,* 1500 bytes). We found this value of l to work well in both simulations and in our implementation.

4.2.2 Clock synchronization

Several very accurate and sophisticated algorithms have been proposed for clock synchronization in multi-hop networks [21, 33, 35]. We believe that it is possible to adapt any of these algorithms to OML. However, OML does not require very precise clock synchronization since we use a relatively large slot time. For evaluation of OML, we have implemented a straightforward algorithm that provides adequate performance within the context of our experiments on the simulator and the testbed.

We synchronize all the clocks in the network to the clock at a pre-designated *leader node*. We estimate the one-way latency of packet transmission based on the data-rate, packet size and other parameters of the 802.11 protocol. We then use this estimated latency and a timestamp in the header of the packet to compute the clock skew at the receiver.

4.2.3 Weighted Slot Allocation (WSA)

In this section, we describe the design of WSA. The challenge is to design a slot allocation mechanism that (a) is fully decentralized, (b) has low control overhead, (c) and is robust in the presence of control message losses and node failures.

For ease of explanation, we present our solution in three stages. In the first two stages, we assume that any two nodes in the network interfere. Thus, only one sender can be active in the network at any given time. Furthermore, in the first stage, we assume that all nodes have unit weight. In the final stage, we relax both these assumptions.

Step 1 - Network of diameter one with unit weights: One solution to achieve fair allocation is to use pseudo-random hash functions as proposed in [11]. Each node computes a random number at the beginning of each time slot, and the node with the highest number wins the slot. The use of pseudo-random hash functions allow a node to compute not only its random number, but the random numbers of others nodes without any explicit communication with those nodes.

Let H be a pseudo-random function that takes values in the interval $(0, 1]$. Consider c nodes, n_1, n_2, …, n_c, that compete for time slot t. Then each node computes the value $H_i = H(n_i, t)$ for $1 \leq i \leq c$, where H is a pseudo-random function, and t is the index of the slot in contention. A node n_r wins slot t if it has the highest hash value, *i.e.,*

$$\arg\max_{1 \leq i \leq c} H_i = r. \tag{1}$$

Since H_i is a pseudo-random random number, it is equally likely that any node will win the slot. This results in a fair allocation of the time slots among the competing nodes.

A node n_s will incorrectly decide that it can transmit if and only if it is unaware of another node n_i such that $H_i > H_s$. While the probability that this can happen cannot be neglected (*e.g.,* when a node n_i joins the network), having more than one winner occasionally is acceptable as the underlying MAC layer will resolve the contention using CSMA/CD. As long as such events are rare, they will not significantly impact the long term allocation.

Step 2 - Network of diameter one with arbitrary weights: Let w_i denote an arbitrary weight associated to node i. Then we define $H_i = H(n_i, t)^{1/w_i}$, and again allocate slot t to node r with the highest number H_r. The next result shows that this allocation will indeed lead to a weighted fair allocation.

Theorem 1. *If nodes n_1,…,n_c have weights w_1,…,w_c, and H is a pseudo-random function that takes values in the range $(0, 1]$, then*

$$P[(\arg\max_{1 \leq i \leq c} H_i) = r] = \frac{w_r}{\sum_{j=1}^{c} w_j} \tag{2}$$

Proof. Due to space limitations, we refer the reader to our technical report [32] for the proof.

Step 3 - Larger diameter network: To enable frequency reuse in networks of a larger diameter, WSA must be able to assign the same slot to multiple nodes as long as they do not interfere with each other. Ideally, the decision to allocate a new time slot should involve only nodes that interfere with each other. Therefore, a given node n should use only the hash values computed for nodes in its interference region. Note that WSA will only ensure weighted fairness among the nodes in the same interference region. Nodes in different interference regions can get very different allocations, based on the level of contention in their regions.

Computing the set of nodes that interfere with a given node is a hard problem, and we are not aware of any good distributed algorithm to solve it. We get around this problem by simply assuming that a node can interfere with *all* nodes within k-hop distance, where k is given. When a node wants to contend for a slot it broadcasts its intention to all nodes within k hops. The set of nodes form which a node hears a broadcast message is then the set of nodes it assumes it interferes with.

There is a clear trade-off between the probability of interference and the bandwidth utilization in choosing the value of k. As the value of k increases, both the probability of interference and the utilization decrease. The reason why the utilization decreases is because, as k increases, a k-hop region will cover more and more nodes that do not interfere with each other in the real system. In this paper, we assume two values of k, k=1, which represents an optimistic assumption as the interference range is typically greater than the transmission range, and k=2, which as we found in our experiments is a more conservative assumption. Designing better algorithms to compute the set of nodes in a node's interference region is a topic of future research.

In addition to the fact that our solution only approximates the real interference region, there are two other sources of inefficiencies in the WSA algorithm. First, the node which wins a slot may not have anything to send during that slot. To address this problem we use a simple timer mechanism, called *inactivity timer*. When k=1, each node n_i initializes an inactivity timer at the beginning of each time slot. Let n_j be the winner of that slot. If the timer of node n_i expires, and the node still has not heard any transmissions from n_j, it assumes that the slot is free. If n_i has the next highest hash value after n_j it starts transmitting in that slot. When k=2, nodes within one hop of n_j will announce to n_i that n_j is silent. To suppress multiple announcements from one-hop neighbors of n_j to n_i, we enforce that of all the nodes within one-hop of both n_i and n_j, only the node with the highest hash value will notify n_i. In practice, we set the inactivity timer to the time it takes to transmit three maximum-sized packets. This helps us avoid false-positives due to packet losses.

The second source of inefficiency is due to race conditions caused by overlapping regions. Consider three nodes n_i, n_j, n_k such that n_i and n_k do not interfere with each other, but both n_i and n_k interfere with n_j. Assume the random numbers of the three nodes are such that $H_i{<}H_j{<}H_k$. Then, n_i will not transmit because n_j has a higher random number, and n_j will not transmit because n_k has a higher random number. Thus, although n_i and n_k do not interfere, they would not be able to transmit in the same slot. This problem is alleviated by the technique presented in Section 4.2.4. Specifically, this technique ensures that only a small number of nodes located close to each other compete for a given slot.

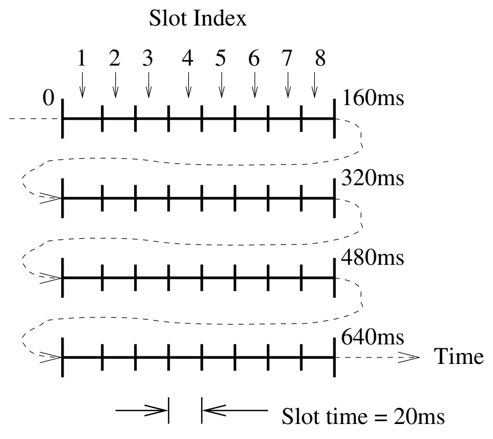


Figure 3: Groups of slots used in the overlay MAC

4.2.4 Amortizing the cost of contention resolution

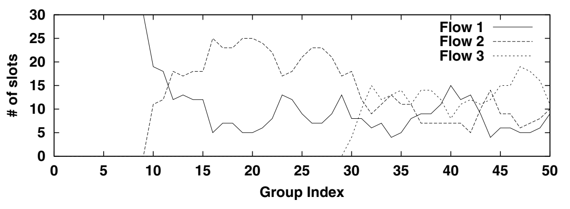
Even though our contention resolution mechanism is fairly light-weight, it is more expensive than the hardware based contention mechanisms used in 802.11, CSMA/CD, and RTS/CTS, respectively. For example, some fraction of a slot might be wasted due to the inactivity timer.

Next we present a simple mechanism to amortize the cost of the contention resolution. The idea is to form groups of N consecutive slots as shown in Figure 3. The *index* of a time-slot is defined as the position of the time-slot within a group of N slots. If a node is allowed to transmit in the slot with index i, the node is implicitly allowed to transmit in the index i of the next group with probability p. In other words, the node will relinquish slot i with probability $1 - p$. Once the node relinquishes the slot, other nodes are allowed to compete for the slot. This mechanism amortizes the cost of contention by a factor of $1/(1 - p)$.

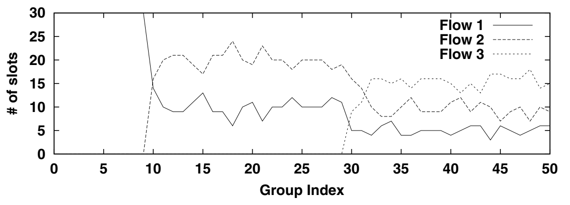
Another advantage of this mechanism is that when a node relinquishes a slot, only the nodes within its interference region are able to compete for the slot. Nodes farther away are likely to be restricted from competing since they are within range of another node that did not relinquish this particular slot. Since contention is typically restricted to nodes within a small region, the race condition described in the previous section is much less likely to occur.

However, these advantages do not come for free. A node needs now to wait for $1/(1 - p)$ slots on average before it can compete. As a result it will take the system longer (*i.e.,* by a factor of $1/(1 - p)$) to converge to the fair allocation when a new node joins or leaves the competition.

To address this issue, we make a slight modification to our earlier definition of of H_i. Let s_i be the number of slots owned by the node n_i when contending for a time



(a)



(b)

Figure 4: Number of slots in every group when using (a) default weights (w_i), and (b) inflated weights (w_i').

slot. We now redefine

$$H_i = H(n_i, t)^{\frac{W s_i / N}{w_i^2}}$$

In other words, if the true weight of node n_i is w_i, node n_i uses a virtual weight $w_i' = \frac{w_i^2}{W s_i / N}$ instead of w_i. In steady state, $s_i \approx \frac{w_i N}{W}$ and hence $w_i' \approx w_i$. Thus, w_i' inflates a nodes weight when it has less than its fair share of slots, but diminishes its weight when it has more than its fair share. Thus, a new sender that becomes active will gain about $(1-p)N$ slots in each round and quickly ramp up to its fair rate.

To illustrate the advantage of using an inflated weight w_i', we use a simple simulation involving three flows with weights 1, 2, and 3, respectively. We assume $N = 30$, and $p = 0.95$. Figure 4 plots the number of slots allocated to each flow as a function of the group index. As expected, when using inflated weights allow new flows to reach their fair allocation faster. In addition, the fluctuations in steady state behavior are smaller than in the case of using the default weights.

The choice of N impacts both the short-term fairness and the time a winner needs to wait to receive subsequent slots without competing again. At the limit, if $N = 1$, the winner is allocated $1/(1 - p)$ consecutive time slots, which hurts the short-term fairness. If $N = \infty$, the winner is allocated one slot at a time as in the baseline algorithm. Based on our experimental results, we chose $N = 20$.

4.2.5 Reducing the control overhead

The basic OML algorithm employs control messages to signal (a) when a node relinquishes a slot, and (b) when a new node enters the competition for a slot. Next, we present two simple optimizations to reduce this signaling overhead.

First, we make use of another pseudo-random function H' to decide if a node should relinquish its slot, *i.e.,* if node n_i owns the slot t, it will give up the slot only if $H'(n_i, t) < (1 - p)$. Again, H' is a pseudo-random function with range $(0, 1]$ and other nodes can compute this value without any communication.

Second, we include the queue length at the sender in the header of each packet. Thus, if the queue becomes empty during the current slot, the node's neighbors are implicitly notified, and the remaining node with the next highest H_i is allowed to transmit for the rest of the slot. If on the other hand the queue of the sender is not empty at the end of the time slot, then its one-hop neighbors will implicitly assume that the node will compete in the next time slot.

In summary, only nodes that join the competition need to transmit individual control messages. All the additional information about active nodes in the interference region is piggy-backed in the packet header. To deal with node failures, we remove a node from the list of contenders if we do not receive an update on its queue-length for an extended period of time.

4.3 Putting everything together

Figure 5 shows the WSA pseudocode. For readability, here we assume that the interference region is the same as the transmission region, *i.e.,* $k = 1$.

Each node maintains a list of all nodes in its interference region that are active, and a map of which time slot is allocated to which node. Since nodes are in promiscuous mode, each node can maintain the list of active nodes by simply inspecting the queue lengths in the headers of the packets it receives. If a node does not hear any message from a neighbor within a predefined interval of time, it assumes that the neighbor is no longer competing and removes it from its list. This allows the algorithm to be robust in the presence of packet losses and node failures.

To implement a two hop interference region, a node simply piggybacks the information about all its one-hop neighbors in the packets it sends. This information is spread across the headers of all packets it sends during the slot to reduce the per-packet overhead. This allows each node to learn about its two-hop neighborhood. Similarly, when a node becomes active, it broadcasts a control message to its two-hop neighborhood.

```
function scheduler()
  while (TRUE)
    getSlotIndex(&slotId, &slotIndex);
    // check whether you already own the slot
    if (owner[slotIndex] ! = myAddress)
      setInactivityTimer();
    if (H'(slotId, owner[slotIndex]) < $(1 - p)$)
      // contendForSlot updates owner[]
      contendForSlot(slotId, slotIndex);
    while (!endOfCurrentSlot())
      if (owner[slotIndex] == myAddress)
        p = getPacket(omlQueue);
        if (p) send(p);

function recvPacket(p)
  if (p.type == CONTEND or p.header.q_size > 0)
    active_list.add(p.src);
    // don't content for this slot
    cancelInactivityTimer();
    // active list may have changed; recompute winner
    computeWinner(slotId, slotIndex);
  if (p.type == RELINQUISH or p.header.q_size == 0)
    contendForSlot(slotId, slotIndex);
  if (p.header.q_size == 0)
    active_list.remove(p.src);
  if (p.type == DATA)
    // deliver packet locally if this node is
    // the destination; otherwise route it
    processData(p);
function contendForSlot(slotId, slotIndex)
  if (!isEmpty(omlQueue) and totalOwnedSlots == 0)
    // other nodes may not be aware this node is active
    sendContentionMessage();
  computeWinner(slotId, slotIdx)

function handleInactivityTimer()
  contendForSlot(slotId, slotIndex);

function cleanActiveList()
  // This function is called periodically by every node
  // to time-out failed nodes from the active_list
  for all n ∈ active_list
    if (curr_time - n.time_added > FAIL_TIMEOUT)
      active_list.remove(n);
```

Figure 5: The WSA algorithm for an one-hop interference region.

5 Experimental Test-bed

In order to motivate our work in Section 3, we use some examples of actual observed performance in an 802.11 testbed. In this Section, we describe the testbed that we use for these results. Our testbed currently consists of 6 wireless nodes based on commodity hardware and software.

The hardware consists of small form-factor computers equipped with a 2.4GHz Celeron processor and 256MB of RAM. Each computer is also equipped with a Netgear WAG511 [6] tri-mode PCMCIA wireless Ethernet adapter. This card is capable of operating in 802.11a,b and g modes, but we conduct all our experiments in 802.11a mode to avoid interference with another production 2.4GHz wireless network operating in same environment.

We have installed Linux (kernel 2.4.22) in all these systems along with the MadWiFi [3] driver for the wireless cards. For routing, we use the Click software router [26] because it provides an easily extensible modular framework. Also, the MIT Grid [4] project provides a readily download-able implementation of DSR [25] and AODV [29] on top of Click. More details about our implementation of OML are available in [32].

6 Simulation Results

In this Section, we evaluate the benefits of OML by using Qualnet [7], a commercial packet-level wireless simulator. Our results can be summarized as follows. First, despite the additional control overhead, the throughput of an OML/WSA network is comparable to the throughput of an 802.11 network. This is because the loss in throughput due to the control overhead is offset by the fact that OML/WSA experiences much lower contention. Second, WSA significantly improves the fairness in a multi-hop network. In particular, while in a multi-hop 802.11 network, a significant percentage of TCP flows are shut-off, OML/WSA completely avoids this problem.

To quantify the fairness, we use the metric defined by Chiu and Jain in [14]. Consider a system with M flows, with weights $w_1, w_2, \ldots, w_M$, each receiving a throughput $x_1, x_2, \ldots, x_M$. The fairness index F is defined as

$$F = \frac{(\sum_i x_i/w_i)^2}{M \sum_i x_i^2/w_i^2}$$

Note that $F = 1$ when each flow's throughput is exactly in proportion to its weight, and $F = 1/M$ when only one flow receives the entire throughput.

We next describe the simulation setting. Each node in the simulation is equipped with an 802.11a network interface operating at 6 Mbps. We use the outdoor two-ray propagation model which yields a radio range of about 350 m. In all experiments, we maintain the density of the network constant at 50 nodes per square km, and place the nodes at random locations. The time it takes to transmit an 1500 byte packet, including the overhead of the MAC and physical layers is about 2 ms. Hence, we choose a slot time of $l = 10$ ms, and a group with $N = 20$ slots. We have disabled the RTS-CTS handshake in all simulations since this maximizes the performance of both the baseline 802.11 and the OML networks. We use the AODV [29] routing algorithm, and run each simulation for one minute (simulation time).

6.1 Collisions and Throughput

As mentioned in Section 4, OML/WSA can achieve better fairness than native 802.11 in a multi-hop network. To evaluate the effects of this trade-off on throughput, in this experiment we vary the size of the network from 15 to 50 nodes, and measure the throughput achieved by a number of simultaneous UDP flows. All flows originate at different nodes, but have the same sink. Such a traffic pattern models a multi-hop network used to access the Internet through a single gateway.

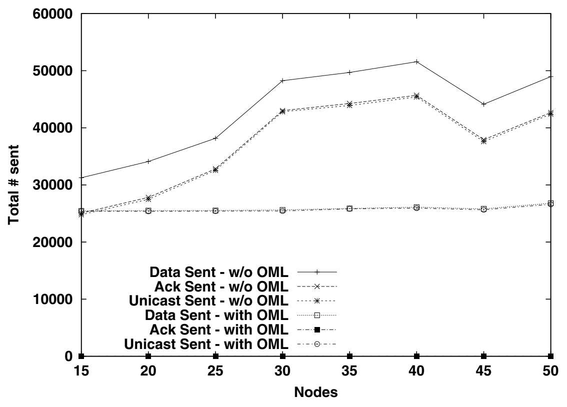


Figure 6: Packet retransmissions in a multi-hop network with 10 flows

Figure 6 shows the effect of the size of the network on the total number of (a) DATA packets and (b) ACKs successfully transmitted at the data link layer, and the (c) unicast packets successfully delivered to the destination. For each network size we consider 10 simultaneous UDP flows. The difference between (a) and (c) gives the total number of packet retransmissions at the MAC layer. Of these (a)-(b) are due to the loss of DATA packets, and (b)-(c) are due to the loss of ACK packets. Because in these simulations OML makes the conservative assumption that all nodes in a two-hop neighborhood interfere, there is less frequency reuse and the total number of transmissions attempted is lower than in 802.11. On the other hand, since OML does not allow competing senders to be active at the same time, there is hardly any MAC layer retransmission. In contrast, in the case of 802.11, up to 25% of the packets are retransmitted. Note that applications that use broadcast transmissions (*e.g.,* routing protocols) can greatly benefit from fewer collisions since broadcast packets are not acknowledged and retransmitted at the MAC layer.

Figure 7 plots the average throughput of 5 or 10 simultaneous flows both with and without OML. Even though OML uses additional control overhead and permits less frequency reuse, the throughput with OML is close to the throughput without OML. Also, there is fundamental trade-off between throughput and fairness in multi-hop networks [22] since longer flows consume more resources than 1-hop flows. But by avoiding collisions and using the medium more efficiently, OML provides much better fairness (see Section 6.2) without sacrificing the throughput.

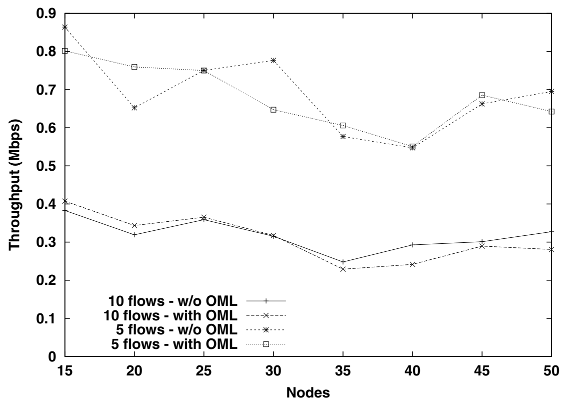


Figure 7: Average throughput in a multi-hop network with and without OML

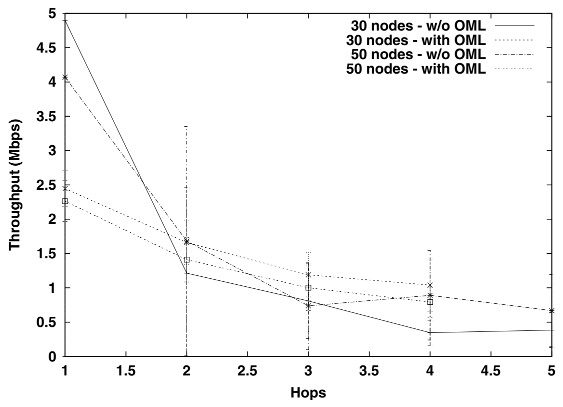


Figure 9: Relationship between throughput and the number of hops

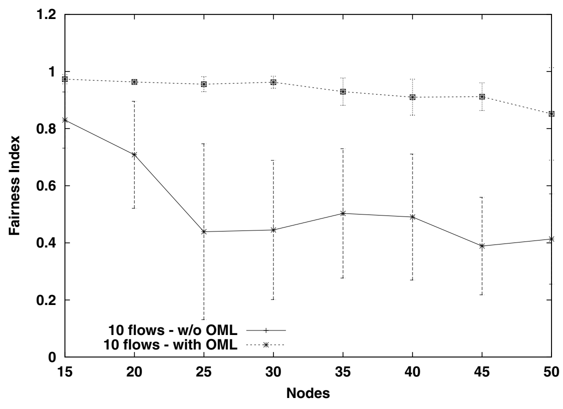


Figure 8: Fairness index of 10 simultaneous flows

6.2 Fairness

A well-known problem with multi-hop 802.11 networks is that short flows, in terms of the number of hops they traverse, receive a much higher throughput than long flows. To show how OML/WSA can alleviate this problem, in the following experiment we set the weight of each node to be equal to the number of unique IP source addresses seen in its output queue; this roughly approximates having unit weight for each flow since there is only one flow starting from each node. Thus a node that forwards packets from m flows will have weight m. Within a single node, we maintain separate queues for each IP source address and implement round-robin scheduling between these queues. If all nodes contend with each other, the weight allocations and the scheduling algorithm will ensure that all flows receive equal throughput irrespective of their length.

Figure 8 shows the fairness index of 10 simultaneous flows averaged over 10 simulation runs. This value can vary from 0.1 (least fair) to 1 most fair in this case. The fairness index in the case of 802.11 is mostly below 0.5 for large networks. This is because in a large network there are more flows with longer routes. In contrast, the fairness index in the case of OML/WSA is above 0.8 even for a 50 node network. The reason the fairness is not 1 is simply because because not all flows compete with all other flows. The only way to achieve $F = 1$ in multi-hop network is to constrain all flows to the throughput of the flow experiencing the most contention, which leads to inefficient use of the channel in the less congested areas.

Figure 9 plots the average throughput of the flows as a function of their length. As expected, 802.11 favors shorter flows, but OML/WSA is more fair. For example, in the case of the 30-node network, the average throughput of four-hop flows under 802.11 is less than 10% of the throughput of one-hop flows. In contrast, with OML, the average throughput of four-hop flows is more than half the throughput of one-hop flows.

7 Results from the testbed

In this section, we evaluate OML/WSA using our experimental testbed. Our main results can be summarized as follows. Section 7.1 shows that OML/WSA can increase the overall throughput of the system by fairly allocating transmission time, instead of transmission opportunities like the 802.11 protocol. Section 7.2 shows that even in a simple chain topology involving one-hop flows, 802.11 can cause a significant number of TCP flows to experience starvation, while OML avoids this phenomenon. Section 7.3 shows that, as expected, the starvation problem is even worse in the case of multi-hop routing, but OML can still handle this problem. In Section 7.4, we evaluate the allocation accuracy of WSA in a simple scenario.

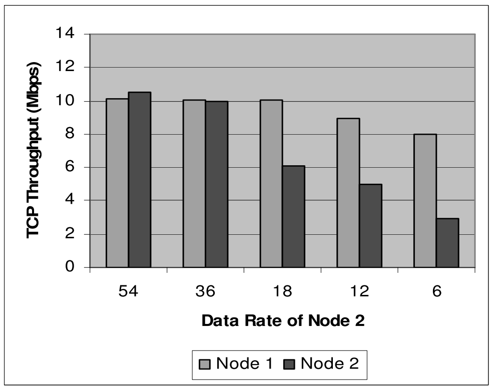


Figure 10: 802.11 throughput in the presence of heterogeneous data rate senders using OML

7.1 Heterogeneous data rates

In this experiment, we reconsider the scenario described in Section 3.1.2 where two nodes are simultaneously sending one flow each to the access point (unlike the rest of this section, we conduct this experiment in infrastructure mode). One node operates at 54 Mbps, and the other node operates at rates ranging from 6 to 54 Mbps. We use the WSA algorithm to allocate the bandwidth, with each node having the same weight. This leads to each node receiving an equal channel-access time, rather than an equal number of transmission opportunities as in 802.11. Note that this allocation implements the temporal-sharing policy as proposed in [34]. Figure 10 shows the throughputs of both flows. The two flows receive throughputs approximately proportional to the rates they are operating at, and the total system throughput drops less than in the case of 802.11 when the second node is operating at 6 Mbps.

7.2 Chain topology

In this experiment, we configure our testbed as a five-hop chain, and pick two random links in this chain. Figure 11 shows the CDF of the flows' throughputs along these links over 50 trials. We consider four scenarios: (a) the flows are started one at a time (b) the flows are started at the same time (*i.e.,* simultaneous flows) using baseline 802.11 (without OML), (c) simultaneous flows using OML assuming one-hop interference regions ($k = 1$), and (d) simultaneous flows using OML assuming two-hop interference regions ($k = 2$). When using OML each flow is assigned a unit weight.

As expected, in scenario (a) when only one flow is active, the flows get very good throughput. In 96% of the cases the throughput is greater than 4Mbps. However,

	OML $k = 1$	OML $k = 2$	OML Oracle	No OML
Throughput	5.7742	5.1784	5.5712	5.8654
Fairness Index	0.821	0.913	0.922	0.803

Table 1: Average system throughput (Mbps) and fairness for two simultaneous flows

in scenario (b) when the flows are simultaneously active without OML, in 24% of the cases, one of the two flows is not even able to establish a TCP connection. This is shown in Figure 11, where 12% of all flows have zero throughput. Furthermore, about 20% of all flows receive less than 1.5 Mbps. In contrast, when using OML with $k = 2$ only about 10% of the flows have a throughput below 1 Mbps. The results when using OML with $k = 1$ are not as good: we only eliminate 6% of the starvation cases. On the other hand, more flows achieve a higher throughput when $k = 1$: around 35% of all flows have throughputs around 4.5 Mbps. These results illustrate the trade-off between using one-hop and two-hop interference regions. Using a two-hop interference region leads to more conservative spatial reuse of the channel. Hence, there are cases in which two nodes cannot transmit simultaneously even though they do not interfere with each other. In contrast, using a one-hop interference region leads to higher throughput, but at the cost of hurting the fairness.

Table 1 further illustrates the trade-off between throughput and fairness as determined by the value of k. The table shows the average aggregate throughput (*i.e.,* the sum of throughput of both flows) and the average fairness index, F, over 50 trials. In addition to the previous experiment scenarios, we consider another one, called "OML with Oracle". This scenario is intended to show the best results OML can achieve when it has complete information about the interference pattern. To implement this scenario, we measure the interference between any two flows in advance and pre-load this information into each node. Recall that for $N = 2$, F ranges from 0.5 (least fair) to 1.0 (most fair). Not surprisingly, in the oracle scenario we obtain the best fairness, without significantly sacrificing the throughput. OML with $k = 1$ comes close to the oracle in terms of throughput, whereas OML with $k = 2$ comes close in terms of fairness. Finally, note that native 802.11 achieves the highest throughput, but the lowest fairness. This is because when one link has a better signal quality than the other, it allows the better link to transmit all the time at the expense of starving the weaker link.

7.3 Multi-hop routing

In this section, we study how OML can improve the flow throughputs in a multi-hop routing network. For this pur-

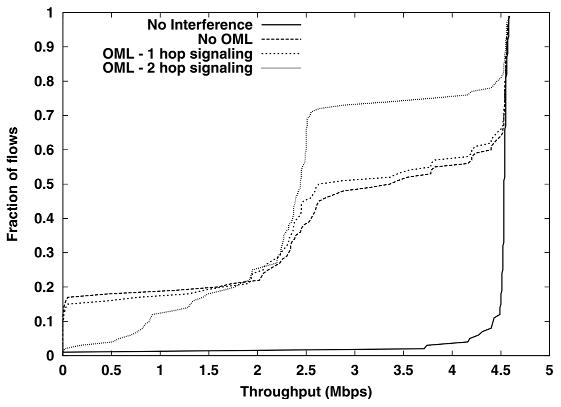


Figure 11: CDF of throughput of 1-hop flows in the network

pose, we arrange the nodes in a simple Y-shaped topology and initiate three TCP flows. Flows A and B originate at nodes 1 and 2, respectively, and terminate at node 4, after being routed via node 3. Flow C is an one-hop flow from node 5 to node 4. The weight of each node is proportional to the number of flows it sees: the weights of nodes 1, 2, and 5 are one, the weight of node 3 is two, and the weight of node 4 is three. In this experiment, we use OML with two-hop interference regions.

First we note that when only one of the three flows is active, flows A and B receive about 2.24 Mbps, while flow C receives 4.55 Mbps, with or without OML.

Table 2 shows the throughputs of each flow when more than one flow is active. Without OML, flow C effectively shuts-off the two-hop flows. In contrast, OML allocates at least 1 Mbps to each flow, when only one of the two-hop flows is active, and at least 0.94 Mbps, when all flows are active. The reason why the throughputs of the two-hop flows are not higher is because the two-hop interference assumption is violated. According to this assumption, transmissions from nodes 1 or 2 should not interfere with with transmissions from node 5. However, this assumption did not hold in our experiment. Let flow D be the one-hop flow from node 1 to 3. When flows C and D were simultaneously active, C received 4.03 Mbps whereas D received only 0.89 Mbps. This shows that transmissions from nodes 1 and 5 do indeed interfere.

7.4 Weighted allocation

In this experiment, we evaluate the accuracy of WSA, and thus its ability to provide per-node service differentiation by setting the node weights. We consider two nodes A and B that send a TCP flow each to a third node C. We set the weight of node A to 1, and assign a weight ranging from 1 to 4 to node B. Table 3 shows the throughput achieved by the nodes for each combination of weights.

Active Flows	Throughput (Mbps)		
	Flow A	Flow B	Flow C
A&C w/o OML	0.14		4.40
A&C with OML	1.69		2.36
B&C w/o OML		0.06	4.48
B&C with OML		1.71	2.08
A&B w/o OML	1.31	1.65	
A&B with OML	1.11	1.11	
A,B&C w/o OML	0.11	0.03	4.42
A,B&C with OML	0.96	0.94	1.96

Table 2: Interaction of flows in an ad-hoc topology

Weight of A	1	1	1	1
Throughput of A	2.26	1.48	1.36	0.92
Weight of B	1	2	3	4
Throughput of B	2.20	2.97	3.06	3.64

Table 3: Throughput received by senders with different weights

As expected, the ratio of throughputs obtained by the two nodes tracks the ratio of their weights closely.

8 Open Issues and Limitations

The current design of OML should be viewed as a first iteration. As we gather more experience with using OML, we expect the OML design to evolve significantly.

Though we believe that the current design of OML does not require any changes to handle *mobility*, since OML relies on neighborhood information being propagated quickly, we do expect a performance penalty as mobility increases. Quantifying this overhead, and further optimizing OML if needed, remains an open problem.

OML does not work well in the presence of interference from native 802.11 clients. In this case, *all* OML nodes will receive an aggregate bandwidth equivalent to that of a *single* 802.11 node. A possible solution would be to detect the presence of competing traffic and allow multiple OML nodes to send data in the same time slot. Finally, in multi-hop networks the interplay between the flow constraints and the interference constraints makes it difficult to precisely formalize the notion of fairness achieved by WSA. We plan on further studying how to define and achieve weighted fairness in a multi-hop network.

9 Conclusions

In this paper, we have described the design and the implementation of an overlay MAC layer (OML) solution which addresses some of the limitations of the 802.11

MAC layer. By using both simulation and experimental evaluation, we show that while 802.11 may cause TCP flows to experience starvation, OML can avoid this problem. In addition, we show that OML can reduce the contention in the network, and provide service differentiation among nodes, with relatively low control overhead.

The power of the OML approach is that it allows us to implement MAC layer functionality *without* modifying the existing 802.11 protocol. In this respect, our approach is reminiscent of the overlay network solutions that aim to implement network layer functionality such as resilient routing on top of the existing IP layer. Ultimately, OML would enable us to experiment with new scheduling and bandwidth management algorithms, and evaluate their benefits to the existing applications, before implementing these algorithms in the MAC layer.

References

[1] AiroPeek NX - Wireless Network Protocol Analyzer. http://www.wildpackets.com/.

[2] Ethereal - Network Protocol Analyzer. http://www.ethereal.com/.

[3] MadWifi. http://madwifi.sourceforge.net/.

[4] MIT grid project. http://www.pods.lcs.mit.edu/grid/.

[5] MIT roofnet. http://www.pdos.lcs.mit.edu/roofnet/.

[6] Netgear. http://www.netgear.com/.

[7] The Qualnet Simulator from Scalable Networks Inc. http://www.scalable-networks.com/.

[8] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP*, 2001.

[9] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *SIGCOMM*, 2002.

[10] L. Bao and J. Garcia-Luna-Aceves. Hybrid channel access scheduling in ad hoc networks. In *ICNP*, 2002.

[11] L. Bao and J. J. Garcia-Luna-Aceves. Distributed dynamic channel access scheduling for ad hoc networks. In *Journal of Parallel and Distributed Computing*, 2002.

[12] V. Bharghavan, S. Lu, and T. Nandagopal. Fair queueing in wireless networks: Issues and approaches. *IEEE Personal Communications Magazine*, 1999.

[13] S. Biswas and R. Morris. Opportunistic routing in multi-hop wireless networks. In *HOTNETS*, 2003.

[14] D.-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Comput. Netw. ISDN Syst.*, 17(1):1–14, 1989.

[15] Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *SIGMETRICS*, pages 1–12, Santa Clara, CA, 2000.

[16] I. Cidon and M. Sidi. Distributed assignment algorithms for multihop packet radio networks. *IEEE Trans. Comput.*, 38(10):1353–1361, 1989.

[17] Wireless networking reference - community wiress/rooftop systems. http://www.practicallynetworked.com/.

[18] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *MOBICOM*, 2003.

[19] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *SIGCOMM*, 1989.

[20] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for multi-hop wireless networks. In *SIGCOMM*, 2004.

[21] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *PDPS Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, pages 186–186.

[22] V. Gambiroza, B. Sadeghi, and E. Knightly. End-to-end performance and fairness in multihop wireless backhaul networks. In *MOBICOM*, 2004.

[23] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *INFOCOM*, San Francisco, USA, March-April 2003.

[24] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *MOBICOM*, 2003.

[25] D. B. Johnson, D. A. Maltz, and J. Broch. DSR: The Dynamic Source Routing protocol for multihop wireless ad hoc networks. In *Ad Hoc Networking*, 2001.

[26] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, 2000.

[27] L. M. S. C. of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11*, 1999.

[28] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. Netw.*, 1(3):344–357, 1993.

[29] C. Perkins. Ad hoc On demand Distance Vector (AODV) routing. *IETF Internet Draft*, 1997.

[30] L. Pond and V. Li. A distributed time-slot assignment protocol for mobile multi-hop broadcast packet radio networks. In *MILCOM*, 1999.

[31] B. Raman and K. Chebrolu. Revisiting MAC design for an 802.11-based mesh network. In *HOTNETS*, 2004.

[32] A. Rao and I. Stoica. An overlay MAC layer for 802.11 networks. Technical Report UCB//CSD-04-1317, University of California, Berkeley, 2004.

[33] K. Romer. Time synchronization in ad hoc networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 173–182. ACM Press, 2001.

[34] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic media access for multirate ad hoc networks. In *MOBICOM*, 2002.

[35] U. Schmid and K. Schossmaier. Interval-based clock synchronization. *Real-Time Systems*, 12(2):173–228, 1997.

[36] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *SIGCOMM*, 2002.

[37] C. A. Waldspurger and W. E. Weihl. Lottery scheduling: Flexible proportional-share resource management. In *Operating Systems Design and Implementation*, pages 1–11, 1994.

[38] Y. Yi and S. Shakkottai. Hop-by-hop congestion control over a wireless multi-hop network. In *INFOCOM*, 2004.

Notes

[1] NAV is maintained internally in the hardware to keep track of RTS, CTS and reservations in the packet header.

[2] We use the term interference region to refer to a set of nodes that compete with each other for access to the channel.