Practical Service Provisioning for Wireless Meshes

Saumitra M. Das, Dimitrios Koutsonikolas and Y. Charlie Hu
School of Electrical and Computer Engineering, Purdue University.

ABSTRACT

Community wireless mesh networks (WMNs) are increasingly being deployed for providing cheap, low maintenance Internet access. For the successful adoption of WMNs as a last-mile technology, we argue that a guarantee of per-client fairness is critical. Specifically, WMNs should support a "bitrate-for-bucks" service model similar to other popular access technologies such as Cable/DSL.

We analyze the effectiveness of both off-the-shelf and theoretically optimal approaches towards providing such a service. We propose the **APOLLO** system that outperforms both these approaches.

APOLLO seamlessly integrates three synergistic components: theory-guided service planning and subscription, rate-based admission control to enforce the planned service, and a novel distributed light-weight fair scheduling scheme to deliver the admitted traffic. We evaluate APOLLO using simulations and testbed experiments.

Categories and Subject Descriptors : C.2.1 [Network Architecture and Design]: Wireless communication
General Terms : Algorithms, Performance
Keywords : mesh networks, service plans, fairness

1. INTRODUCTION

Wireless mesh networks are increasingly being deployed for providing cheap, low maintenance Internet access (e.g. [33, 35, 32]). In contrast to other multi-hop wireless networks such as sensor networks or mobile ad hoc networks, a WMN aims to be a last-mile technology and thus it must compete with existing last-mile technologies. An important feature of other last-mile technologies (such as cable and DSL) is a "bitrate-for-bucks" service model, i.e., a client is typically promised a bandwidth she pays for, such as 128Kbps for $24.99/month. Thus, a fundamental requirement for WMNs is to provide clients with similar service options[1]. This requirement is also key to the adoption of WMNs. Even in places where other access technologies are unavailable or costlier, such a service model is essential to build long-term customer loyalty. The key challenge for such service provisioning is to provide guarantees to the clients of a WMN, especially in the presence of the lossy wireless environment and multi-hop routing in WMNs.

In this paper, we assume that the bitrate-for-bucks promised in a deployed WMN is always achievable by the theoretical capacity of the network [2]. In the absence of such a condition, it is infeasible to provide any guarantee to WMN clients. Given that theoretical capacity exists to support the promised service, we explore the effectiveness of various techniques to provision the "bitrate-for-bucks" service model in WMNs. Wireless meshes are typically constructed using off-the-shelf 802.11 radios since such devices are cheaper due to economies of scale. We thus first explore the service that can be provided using a WMN built from off-the-shelf 802.11 hardware and publicly available software and protocols. We find that the performance of off-the-shelf 802.11 is inadequate for our target service model. This inadequacy is tied to the fact that the 802.11 MAC layer fundamentally is oblivious of fair access to the wireless channel for multi-hop flows.

In search for a good solution, we investigate a theoretically optimal approach termed Fair Scheduling (FS) that assumes time-slotted medium access schedules transmissions to obtain fair service. We find that this approach with modifications to account for practical limitations can indeed provide a "bitrate-for-bucks" service model for specific network topologies and traffic patterns. However, we also find the assumptions made in the centralized scheduling idealistic and limiting the applicability of the solutions for real WMN deployments.

Finally, we propose the **APOLLO** system that leverages off-the-shelf hardware with software modifications to pro-

[1]Enterprise deployments of WMNs where there is no such charging mechanism are outside the scope of this paper.
[2]The capacity value assumes perfect scheduling by an omniscient entity which may not be attained by real protocols.

vision the “bitrate-for-bucks” service. APOLLO seamlessly integrates three synergistic components: (1) A theory-guided service planning and subscription technique to decide how to admit new subscribers. (2) A rate-based admission control to restrict the upload/download traffic from clients in accordance with their service plans. Importantly, APOLLO uses immediate admission control at the access mesh router of the client such that all packets that enter the mesh backbone network are “good” packets that should receive service. This ensures that the traffic in the network is below the capacity of the network. (3) A novel distributed lightweight fair scheduling scheme to deliver the admitted traffic that is robust to unfairness that can arise from interference, variability in wireless channel quality, collisions, etc. This approach rests on the fundamental observation that all such phenomena are concisely represented by an increase in the *queue length* of the affected node. APOLLO prioritizes access to the wireless channel in interference neighborhoods [3] based on the nodes' queue lengths, i.e., nodes with larger queue lengths receive transmission priority. This approach robustly deals with unfairness (and consequent queue backlog). Compared to previous time-slotted fair scheduling approaches [24, 25], APOLLO does not need to schedule particular flows on specific nodes in specific time-slots since all packets in all queues are “paid-for” packets and nodes cooperate and prioritize among each other to get these packets to their destinations.

Our evaluations show that APOLLO can effectively provide a client with the service it paid for in many different network topologies and traffic scenarios. A deployment of an APOLLO implementation over a wireless testbed also demonstrates that APOLLO works well in real-world scenarios. We believe that the use of APOLLO and a “bitrate-for-bucks” service model will aid in the widespread use and adoption of community wireless networks. APOLLO uniquely proposes using admission control in WMNs as a *necessity* since (1) wireless channels have limited capacity which renders overprovisioning of the backbone network difficult; (2) shared wireless media access provides no built-in link-layer support for restricting traffic from greedy clients. Admission control also has favorable implications towards mesh network management since it also enables robust, scalable and secure operations in a mesh by allowing flexible placement of client population, allowing planned deployment, uptime and downtime in a mesh, and preventing DDOS attacks.

2. ARCHITECTURE

This section describes our WMN architecture. We consider a typical wireless mesh network with omnidirectional antennas, in which mesh routers are placed on the rooftops of clients [33] or other infrastructure (e.g., streetlights), and are interconnected via wireless links. One or a few gateways are connected to the Internet and propagate traffic to and from clients. The gateways are not widely deployed

[3]A set of nodes whose transmissions interfere.

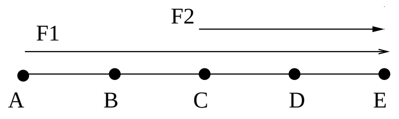


Figure 1: **Sample network topology.**

due to cost and uplink constraints. Clients can be mobile or static and each one is associated with a mesh router (at a given time). We assume mesh routers communicate with their associated clients using different radios/channels from those used to talk to other mesh routers. In fact, the WMN may even use multiple radios for the backbone links. Further we assume that the WMN employs a 802.11 MAC layer.

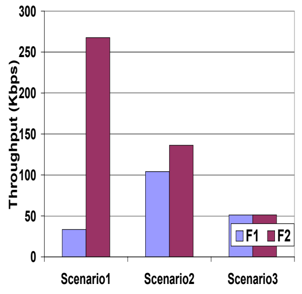
Such a WMN provides Internet access as follows: Each client's packets are first received by the client's *access mesh router*, i.e., the mesh router the client's interface is associated with. These mesh routers then forward the packets to the *gateway mesh router* (GMR) using other mesh routers. The GMR provides Internet connectivity through a high bandwidth wired/WiMax interface. The gateway may perform other functions such as IP address assignment or NAT. All the MRs use a routing protocol (e.g. OLSR [5]) with metrics such as ETX [6] to find routes to each other and to the gateway. The WMN can also be used for peer-to-peer (P2P) traffic between any two clients. In that case packets are not sent to the gateway, but they are routed from the sender client's access mesh router to the receiver client's access mesh router using the same routing protocol used for Internet access.

3. “OFF-THE-SHELF” SOLUTIONS

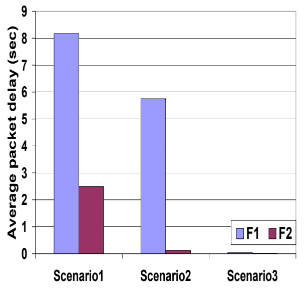
We first explore how current popular off-the-shelf solutions (e.g., 802.11 protocol with ETX routing) work in terms of the service they provide to individual clients. We study the topology in Figure 1 using the Qualnet simulator [23]. In this figure we have 5 nodes, namely A, B, C, D, E, in a chain. All links are of the same distance, have similar loss rates of approximately 10% and each node is within transmission range of its one-hop neighbors. We use the 802.11b MAC layer with a rate of 2Mbps. Nodes A and C each generate one UDP [4] flow to the gateway node E. The per client service plan assigned in this topology is 140Kbps for a given price. We ascertained that the network can support this service plan (see Section 5.1). Further, we assume that the service plan is always chosen to be close to the capacity of the network. Overprovisioning a network is not economically viable.

Figures 2(a)-2(b) show the throughput, and average packet delay for each of the two flows in three scenarios: (1) Users demand more than the service plan (both transmit at 300Kbps) since this solution has no enforcement of service plans; (2) Users demand their service plan (both transmit at 140Kbps); and (3) Users demand less than their service plan (both transmit at 50Kbps).

[4]We use UDP to measure performance since it does not provide a conforming workload. This choice is widely used in wireless



(a) Throughput



(b) Average packet delay

Figure 2: Throughput and delay of 2 flows in the sample network topology.

From these three figures we make the following observations about the performance of off-the-shelf solutions: (1) When the flows are allowed to transmit whatever they desire, the network is overutilized, and the service provided is grossly unfair, i.e., one user ($F1$) gets less than their service plan. (2) This unfairness is also observed when the flows transmit as per their service plans and the network is operated near capacity in scenario 2. This unfairness becomes more prominent if we look at the average packet delays of the two flows. The average packet delay for flow $F2$ is 123 msec, but for flow $F1$ it is 5.75 sec, that is 47 times larger. (3) Finally, when the flows operate below their service plans, the network is underutilized and off-the-shelf solutions work fine because the occurrence of inefficiencies in accessing the medium are outweighed by the time the medium is idle.

Thus, off-the-shelf approaches are inadequate for our target service model and only work when the network is underutilized. When the network is utilized well or overutilized, 802.11 is unable to allocate service in an acceptable manner. The lessons learnt from this experiment are: (1) policing the traffic allowed to enter the network is a *necessity* in order to provide any guarantees as per service plans; (2) service plans can be offered by highly over-provisioning the network which however is not economically viable; (3) operating the network within (close to) network capacity, e.g., by policing the admitted traffic, alone is insufficient due to unfairness, and additional mechanisms are needed to mitigate unfairness. With this in mind, in the next section we investigate a TDMA [5] approach for service provisioning that is theoretically optimal (under a set of assumptions).

4. “OPTIMAL SCHEDULING” SOLUTIONS

As an example of a theoretically optimal solution we consider “Fair Scheduling” (FS) for wireless mesh networks. We first provide a brief background of FS.

4.1 Fair Scheduling (FS)

FS assumes knowledge of routing paths (a tree from the gateway to all clients) and uses a scheduling algorithm based on spatial TDMA [19] to schedule transmissions of mesh routers such that all clients in the network receive a fair share of the bandwidth. The specific example technique for FS used in this paper is from [25] and consists of three phases: compatibility matrix (CM) construction, clique enumeration, and clique selection.

The algorithm starts with the construction of a compatibility matrix (CM) which denotes which links in the network can transmit simultaneously. $CM[ij] = 1$ if links i and j do not interfere and can transmit simultaneously and $CM[ij] = 0$ otherwise. In [25] the CM is constructed based on the following rule: “Two links interfere if the sender of one link is within transmission range of the sender or the receiver of the other link”. The algorithm then enumerates all possible cliques [6] in the CM. Links in the same clique can transmit together to exploit spatial reuse.

Let Cl_k^α denote the α-th clique of cardinality k and L_k^α denote the most loaded link in Cl_k^α. The load of link L_{AB} $l(L_{AB})$ is defined as the number of clients that use this link to transmit traffic to or from the gateway. Let also d_k^a denote the number of slots required to transmit the traffic of the most loaded link in clique Cl_k^α. Then $d_k^a = l(L_k^\alpha)$. Each link L_{AB} in Cl_k^α is activated during $l(L_{AB})$ time slots and is idle during $d_k^a - l(L_{AB})$ time slots. Hence, the clique Cl_k^α generates a gain $g(Cl_k^\alpha) = (\sum_{L_{AB} \in Cl_k^\alpha} l(L_{AB})) - d_k^a$.

A scheduling is defined as a set of cliques s that fulfills the following two conditions: (i) all links that are part of the tree are included in the schedule and (ii) each link is included only once. Finding a schedule s with the minimum cycle length T_s ($T_s = \sum_{Cl_k^\alpha \in s} d_k^a$) provides a fair share while maximizing throughput. Instead of finding a minimum length schedule by exhaustive search, the authors in [25] propose a greedy heuristic according to which they maximize the total gain (g_s) of the schedule instead. The length of the calculated schedule in turn dictates the throughput (bitrate) that can be assigned to each client.

How well does FS perform in practice? To see how efficient FS is in practice, we ran it over the topology in Figure 1. While the length of the calculated schedule dictates a service plan of 140Kbps for both flows, surprisingly, as shown in Figure 3, we found that $F2$ achieves a throughput close to the service plan (118Kbps) while $F1$ starves (58 Kbps). Most importantly, FS performs worse than 802.11. Thus, theoretically optimal solutions directly applied do not work well and the next section identifies why.

4.2 Improving FS

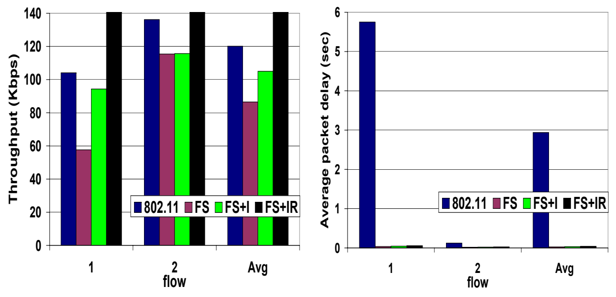
We identified two practical problems related to FS that made it perform badly in practice: simplified interference estimation and ideal link assumption.

Simplified Interference Estimation The interference rule in FS is not enough in a realistic environment. A recent work [21] has shown that simplified heuristics (2-hop interference) usually fail to model interference accurately and a measurement-based approach is more accurate. In reality,

network studies.

[5]Assuming a new MAC layer can be deployed.

[6]While the problem of clique enumeration is known to be NP-hard, computation is done with the Bron-Kerbosch [3] heuristic.



(a) Throughput (b) Average packet delay

Figure 3: Performance comparison of 2 flows in the sample network topology for 802.11, FS , $FS + I$ and $FS + IR$.

a node can be affected by a transmission even if it is at a distance more than twice the transmission range from the sender. Although packets cannot be properly received at such long distances, they can still collide with other transmissions.

To take interference into account, we incorporated the BIR metric [21] into FS by setting $CM[ij] = 0$ if $BIR_{ij} < 0.9$ [7] (we call this $FS + I$, i.e., FS with Interference-awareness). The BIR metric can take any value between 0 and 1 (quantifying the real interference between two links) and is estimated through offline measurements.

Ideal Link Assumption FS, similar to many other scheduling algorithms in literature [15, 16, 17], assumes that the only reason for packet loss is packet collisions when neighboring nodes transmit simultaneously and scheduling avoids these losses. However, in a real network, a second important reason for packet loss is the time variability of the wireless channel. Reflections, scattering and other factors cause multipath fading, which may randomly deteriorate the quality of the links. This causes random packet loss which can severely harm performance. 802.11, although still unfair, provides higher throughput than FS because it includes a reliability mechanism (ACKs and retransmissions) that helps nodes recover from such random packet losses.

We added MAC layer retransmissions to FS by making the sender wait for a certain amount of time in each slot after each packet transmission for an ACK. If no ACK comes, the sender retransmits the packet in its next time slot. A maximum number of retransmissions is allowed for each packet, after which the packet is dropped. This number is a tradeoff between reliability and throughput. We call this the version of $FS + IR$, i.e., FS with both Interference-awareness and Reliability.

Performance of improved FS We reevaluated FS with our proposed fixes and the results are shown in Figure 3. While FS scheduled node A and D together, $FS + I$ schedules them separately since it figures out that these nodes interfere (their transmissions collide at node B). However, the performance of $FS + I$ is still affected by packet loss. After incorporating both fixes, $FS + IR$ provides an almost optimal fair service to both flows; both flows achieve a throughput of 140Kbps, equal to their demand. Most importantly, this is achieved with only 1 retransmission attempt per packet, while 802.11 has much worse performance even with 7 retransmissions. Hence, assuming there is an oracle that has all the necessary information to calculate the optimal schedule all the time, FS enhanced with interference-awareness and a simple reliability mechanism can provide the service model we target.

[7] BIR_{ij} is measured as the ratio of total packet delivery rates over links i and j when both are transmitting over when transmitted individually.

4.3 Practical problems with FS

In practice, however, there are a number of fundamental challenges in implementing a TDMA-based scheme like $FS + IR$ which effectively prevent such a solution from becoming deployable. In the following, we summarize these implementation challenges.

Real-time flow information, schedule calculation and propagation: The optimal schedule in a TDMA approach is directly related to the flows present inside the network, i.e., going through each node. Hence, information about the arrival and departure of flows and changing traffic has to be propagated to the scheduler, which has to compute a new schedule and propagate it to all the nodes, all in a timely fashion. The reliable propagation of flow information and the updated schedule is not a trivial task. Also, requiring all nodes to switch to the new schedule at the same moment (in order to avoid collisions) requires global synchronization.

Dependence on routing path: The optimal TDMA schedule is directly dependent on the routing paths used by the flows. Even if a node has a single associated flow, routing path changes (which are frequent in wireless networks) necessitate frequently changing schedules.

Dependence on fine-grained time slot and synchronization: In simulation, the time slot can be easily set equal to the time required to transmit one packet. Such a fine-grained time slot is difficult to achieve in practice. Hence papers that offer implementations of time-slotted systems always use a much larger time slot that allows for many packet transmissions (e.g., [24, 20]), which cannot offer maximum performance achieved by fine-grained time slotting. Further, tight time synchronization among the wireless mesh routers is a challenge in itself.

Link-quality adaptation: Finally, it is difficult to incorporate effective link-layer adaptation techniques such as rate adaptation into the scheduling algorithm since link quality is a highly dynamic quantity and even if known, would make the schedule computation very expensive.

5. APOLLO: PRACTICAL SERVICE PROVISIONING

Given the performance problems with off-the-shelf approaches and the fundamental problems in adopting fair scheduling for real deployment, we propose the APOLLO system which retains the simplicity of off-the-shelf approaches (by being implemented on top of cheap 802.11 radios) and performs practical scheduling of packet transmissions. One key property makes APOLLO practical: *its scheduling is oblivious to the number, the source and destinations of multi-hop flows present in the network, and the routing paths by these flows, and it does not require fine-grained time synchronization* [8]. This section presents the design (theoretical formulation for service planning, admission control for enforcing service, lightweight scheduling to deal with unfairness) and implementation of the APOLLO system.

5.1 Service Planning

The foremost question for service provisioning is: If a new client wants to subscribe to the network, how can we decide if the client's desired rate plan can be supported based on the network link and interference characteristics, existing clients and their rate plans, etc. If not, a closely related followup question is to analyze what hardware configuration and provisioning need to be changed to accommodate the client at the lowest cost. This is an interesting research problem by itself which we are investigating. We now outline some methods that can be used for such service planning.

Planning could be done using $FS + IR$ we described in Section 4. Using this, we can estimate whether a new customer can be added and with what maximum rate plan. If TDMA fails to find a schedule to accommodate the new client and rate plan, there is not enough capacity to add the client. Note that these calculations will provide some insight but will not be exact. In fact, one benefit of APOLLO is that it can tolerate temporary fluctuations caused due to this inexact calculation, and the consequent transient congestion in operational networks. However, $FS + IR$ cannot model variable link quality, complex interference relationships (multi-way interference [7]), multiple gateways and requires routes to be predefined.

A more general and thorough albeit heavyweight service planning technique is through a linear program shown below adapted from a general model in [11]. The WMN of N nodes is modeled with a connectivity graph C whose vertices are the mesh routers (N_c) and edges the wireless links (L_c) between the routers annotated with a link capacity (Cap_{ij}) calculated via offline network measurements. f_{ijk} corresponds to the amount of flow on link l_{ij} for connection k. To model download according to a rate plan, we use a multi-commodity flow formulation with one connection for each client from its best gateway. The set of gateways is S, set of clients D, set of connections K (one per client). A virtual sink node is linked to each *existing* client with a capacity equal to the rate plan subscribed. This link is special and does not cause interference. We also derive a conflict graph of the network through measurement [21] in which each link in L_c becomes a node and edges between nodes denote interference. This is required by the LP to determine which links can be scheduled simultaneously (spatial reuse).

Given the above inputs, the LP below can check if adding a new client at a specific location in the graph with a specific service plan is feasible as follows: A new client+plan is added to the existing graph C with sinks denoting rate plans of existing subscribers and the LP evaluated for feasibility to admit the new client. The constraint in Eq. 1 is flow conservation while the constraints in Eq. 2 say that the incoming flow to sources is 0 while outgoing flow from destinations is 0. Eq. 3 disallows negative flows. Eq. 4 and Eq. 5 force the LP to consider single-path routing (a common protocol semantic) since multiple network paths can have adverse effects on TCP; and also indicate that the amount of flow per link cannot exceed the link capacity. Eq. 6 provides a lower bound on optimal throughput using the conflict graph to determine which links can be scheduled together (details in [11]). H_x in Eq. 6 defines a schedulable set of links that can simultaneously be active and k' is the number of schedulable sets found. $\lambda_x, 0 \leq \lambda_x \leq 1$ denotes the fraction of time allocated for set x, i.e. the links $l_{ij} \in H_x$. Finally, the LP can also be used to model improvements (more radios, mesh routers, gateways) to support the new client.

$$max \sum_{s \in S} \sum_{l_{si} \in L_C} f_{sik} \forall k \in K \ \ subject\ to:$$

$$\sum_{l_{ij} \in L_C} f_{ijk} = \sum_{l_{ji} \in L_C} f_{jik},\ n_i \in N_C \setminus \{n_s, n_d\}, \forall k \in K \tag{1}$$

$$\sum_{s \in S} \sum_{l_{si} \in L_C} f_{isk} = 0 \quad \sum_{d \in D} \sum_{l_{di} \in L_C} f_{dik} = 0, \forall k \in K \tag{2}$$

$$f_{ijk} \geq 0\ \forall i,j,k \mid l_{ij} \in L_C,\ k \in K \tag{3}$$

$$\sum_{k \in K} f_{ijk} \leq cap_{ij} \cdot z_{ijk}\ \forall i,j \mid l_{ij} \in L_C,\ z_{ijk} \in \{0,1\} \tag{4}$$

$$At\ each\ node\ n_i, \sum z_{ijk} \leq 1 \tag{5}$$

$$\sum_{x=1}^{k'} \lambda_x \leq 1,\ \sum_{\kappa \in K} f_{ijk} \leq \sum_{l_{ij} \in H_x} \lambda_x \cdot cap_{ij} \tag{6}$$

Note that link quality fluctuations are unavoidable in wireless networks and thus capacity may change over time. Link fluctuations can be measured and characterized [8] to be taken into account in service planning by considering average case/worst case performance of the link.

5.2 Admission Control

The second component in APOLLO's design is rate-based *admission control*. This component is fundamental in all last-mile techniques such as cable and DSL. In APOLLO, this is achieved in software since shared wireless media access provides no built-in link-layer support for restricting

[8]APOLLO provides a bitrate guarantee similar to Cable/DSL and while it improves delays, strict per-packet delay guarantees would require more complex scheduling. It is not clear if it is practical to implement fine-grained scheduling in multi-hop wireless networks.

traffic from greedy clients. Hence, similarly as in all Internet technologies, in APOLLO, each client's traffic is restricted at the client's access mesh router to the client's paid service plan. If the client tries to send or receive more traffic than that allowed by its service plan, this traffic will be dropped at its access mesh router and it will not enter the WMN. An immediate implication of this policy is that all packets that enter the WMN backbone are "paid for" and should be delivered.

5.3 Priority Scheduling

The third component of APOLLO's design is to address unfairness that can occur when operating the network close to capacity (Section 3) as well as due to channel variation, loss and replacement of routes and external interference. In these scenarios, APOLLO provides a reactive fair scheduling without requiring a fine-grained time-slotting. The basic premise is that since all packets on the backbone are "paid for", they should be handled with equal priority.

5.3.1 Key concept

The design of APOLLO's lightweight distributed priority scheduling is based on the key observation that while there are many different causes of unfairness in a WMN environment (interference, fading, packet collisions, 802.11 backoff algorithm), all of them typically result in a single symptom: *an increase in the queue length of the affected node*. Rather than identifying and attempting to cure the "disease" (which is significantly harder to identify and address), APOLLO directly addresses the "symptom", i.e. the queue length. An increase in the queue length increases the average packet delays and reduces the throughput. Finally when the queue becomes full, packets start being dropped. Hence, it is critical to schedule packet transmissions in a way that will prevent increases in the queue lengths. Following this guideline, APOLLO prioritizes access to the wireless channel in each interference neighborhood to nodes based on their queue lengths, i.e., nodes that have higher queues receive transmission priorities. This is done as follows: Each node is made aware of the queue lengths of all other nodes with which it interferes (we describe the way this information becomes available in the next section). If a node has the highest queue in its neighborhood, it acquires the channel and starts transmitting packets. All other nodes defer their transmissions. As long as the information about the neighbors' queue lengths is not outdated, all nodes in a neighborhood will make the same decision on which node should transmit; hence there will be no packet collisions. The node that obtained the right to transmit will keep sending data packets reducing the length of its built-up queue, until the moment that some other node obtains a larger queue length. In essence, APOLLO performs priority scheduling temporarily when a node's queue builds up but does not rely on any a priori schedule, which make implementation feasible. In addition, if the maximum queue in a neighborhood is small (underutilized network), APOLLO behaves similarly to 802.11, which works fine when utilization is low. Note that in some corner cases, it is possible that APOLLO reduces spatial reuse of the channel because of dependence among nodes in overlapping interference neighborhoods, i.e., node A defers to node B which defers to node C, but node A and C could transmit together. While this is a bigger concern for slot based scheduling algorithms [24] which assign turns every cycle, APOLLO is less affected since queue backlogs are typically transient and localized. Nonetheless, a solution such as an inactivity timer [24], i.e., a deferring node that senses a channel free for some time can transmit, can be used in APOLLO as well.

5.3.2 State dissemination

APOLLO requires nodes to maintain correct, up-to-date information about their neighbors' queue lengths. The design of state dissemination focuses on (i) to which nodes this state is disseminated and (ii) what is the dissemination mechanism.

Dissemination neighborhood Although a reasonable answer to the first question seems to be "as many nodes as possible", such a choice is both impractical and incorrect. First, we want information to be propagated quickly, so that all nodes make a correct decision as soon as possible. Second, propagating information beyond an interference region might harm throughput. For example, if all nodes in the network know each other's queue lengths, then there will be only one node transmitting in the whole network, thus not exploiting spatial reuse. Hence, information should only be propagated to nodes within an interference region. Nodes that belong in different interference regions make different decisions. The goal is that at any given time, each interference region should have a winner node that transmits, exploiting spatial reuse at the maximum degree, while preventing collisions.

Unfortunately, identifying an interference region is not a simple task. The most common solution is to assume that all nodes within a k-hop distance interfere with each other, where k is a tradeoff between the probability of interference and channel utilization. [24] uses a 2-hop interference region. However, their simulation model is not realistic, since they only used the two-ray propagation model, without simulating any channel variability (noise, fading, etc.). As we saw in Section 4, with a more realistic physical model, even nodes in a 3-hop distance can interfere. We have also verified it experimentally in our 32-node testbed [7]. Hence, in APOLLO we use a 3-hop interference neighborhood. While this is a heuristic, it makes the system practical. Measuring continuously changing interference patterns is impractical and is difficult to perform online.

Dissemination mechanism APOLLO uses two CONTROL messages: **BEACON** and **LEAVE** messages. The BEACON message broadcast every 200ms[9] contains the node's

[9]This value was obtained after experimental sensitivity analy-

current queue length, a sequence number used for recency information and a TTL value initialized to 3 to implement the 3-hop interference neighborhood. To reduce overhead, BEACONs are sent only if the difference between current queue size and the last advertised queue size is more than 5% of the maximum queue size.

While BEACONs allow nodes in an interference neighborhood to periodically evaluate all the queue lengths and decide who should transmit for the next period, the winning node may finish all its packets before the next BEACON exchange. To minimize idle time, whenever a node that is currently transmitting has no more packets to send, it initiates a LEAVE message. LEAVE messages are also broadcast in a 3-hop neighborhood. Each node receiving a LEAVE message, updates the queue length of the node that initiated the LEAVE to 0, and then it decides to transmit if its queue has become the largest one. The problem is that the moment a node receives a LEAVE message, the information it has for all other nodes is outdated; it is what the node had learnt in the last BEACON exchange phase. For this reason, upon receiving a LEAVE message, a node in addition to rebroadcasting it, also piggybacks its current queue length in the packet. Since decisions to transmit/defer are remade on every LEAVE/BEACON message receipt, wrong decisions because of incomplete or outdated information are corrected soon.

Since recency is important for BEACON and LEAVE messages, these packets have higher priority than data packets. One approach is to transmit them immediately using 802.11 broadcast. However, since 802.11 broadcast uses no control packets such as RTS/CTS/ACK and transmits each packet only once, the CONTROL packets will likely be lost due to collisions with other CONTROL or data packets. For this reason, we use a separate channel to transmit BEACON and LEAVE messages. Using a separate control channel and radio potentially wastes the capacity of the WMN. However, we argue the control radio is justified for a number of reasons: (1) When WMNs are deployed as operational commercial networks, there is a need to separate the control and data plane which enables real operation and management of WMNs and will probably be required in most cases so that network debugging, code updates, interference measurements, channel assignment, etc can be done without hurting paid customer traffic. (2) Additionally, this single control radio can be used to provision a system with multiple data radios (which are commonly available). For multiple data radios, BEACON messages now will contain a vector of the queues on each radio of a node and a LEAVE message has to specify for which radio it has been sent. (3) Finally, many newly proposed protocols ([30], [26], [22]) for WMNs require a control radio which can be leveraged by APOLLO.

As a side note, we did try other priority schemes based on queue lengths such as prioritizing nodes based on DIFS and backoff [1], before converging to our final scheme. We also tried a scheme where a node's decision on transmitting or not is not discrete (the node transmits if it has the largest queue) but continuous, i.e., based on the queue length a node decides to transmit with probability P, and P is higher for nodes with larger queues. None of these schemes worked efficiently in resolving contention, because they are probabilistic [29]. Essentially, probabilistic schemes did not respond fast enough to alleviate congestion and resulted in lower throughput due to delays and packet drops.

sis [27].

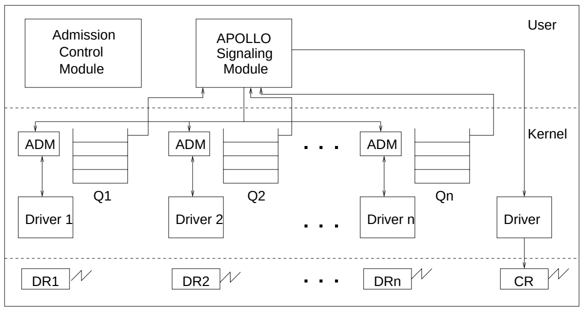


Figure 4: **APOLLO architecture.**

5.4 Implementation

We implemented APOLLO in Linux 2.6.8. The APOLLO architecture is shown in Figure 4. The two main functions of APOLLO are admission control (implemented by the APOLLO Admission Control Module (AACM)) and priority scheduling (implemented by the APOLLO Signaling Module (ASM) and the APOLLO Driver Modules (ADMs) (one for each radio)). As mentioned in Section 5.3.2, control messages are sent through a different radio. In Figure 4, we denote it by CR (Control Radio) to distinguish it from the radios used for data (DRs).

APOLLO Admission Control Module While admission control according to rate plans can be implemented using the `tc` utility (http://lartc.org/howto/), we perform this function using our own user-space module because: (1) Backlog in the actual device queue is used for deciding scheduling priority. So packets sent exceeding the service rate should not be allowed to fill that queue and thus should be blocked in user-space. (2) We can implement rate plans across multiple radios easily by doing it in user-space. The AACM at the access mesh router uses *libipq* and Netfilter to perform packet capturing and rate limiting for outgoing packets for associated clients according to a service plan. Similarly, incoming flows to a client from multiple sources can be limited in aggregate similarly. However, the flows must be responsive to loss (i.e. TCP) so that packets that will be ultimately dropped at the destination (due to the destination's incoming admission control) in the WMN slow down the sending rate at the sources. This means that only TCP or TCP-friendly traffic will be allowed to enter the mesh: we argue this restriction is necessary given the bandwidth scarce operational environment and does not preclude many applications. To handle mobility, each client is identified by its MAC address on sign-up and the client's plan is made known to all mesh routers. Thus, if a client moves and associates with a different mesh router, the admission control functionality is enabled at the new access mesh router.

APOLLO Signaling Module The ASM decides when the driver should transmit or defer, and exchanges queue length information through BEACON and LEAVE messages. ASM reads the queue length by communicating with the kernel. For each packet being transmitted from the IP layer, the *dev_queue_xmit()* procedure is called. It queues a packet in the qdisc associated with the output interface, determined by routing. Then, if the device is not stopped, all packets in the qdisc are handled by *qdisc_restart()*, which finally calls the *hard_start_xmit()* method, implemented in the driver code. ASM uses the number of backlogged packets in the qdisc to decide if the card should transmit or not, and in sending LEAVE and BEACON messages.

APOLLO Driver Module The ADM consists of changes to drivers to support APOLLO transmit/defer decisions and are driver-specific. To make APOLLO portable across drivers we made the interface of the ADM and ASM very simple: ADM for each interface communicates (through the */proc* file system) with ASM and applies the decisions made by the ASM on whether this particular interface should transmit or not. When in deferred state, ADM blocks the kernel from calling *hard_start_xmit()* using *netif_stop_queue()* and when transmitting, *netif_wake_queue()* is called to allow the kernel to call *hard_start_transmit()* and send packets to the wireless card. In this paper we used the *madwifi* driver [18]. Similar modifications can be easily done in any other driver.

6. EXPERIENCE WITH APOLLO

In this section, we describe our performance experience with the Apollo system. We first evaluate APOLLO using the Qualnet simulator in Section 6.1, in order to understand its behavior in a controlled environment (with controlled node positions, distances, channel conditions, etc). Next, in Section 6.2, we evaluate the APOLLO implementation in our testbed.

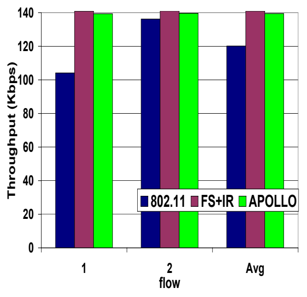
6.1 Simulation Evaluation

We first evaluate APOLLO in two simple scenarios using the 5-node chain of Figure 1 by comparing it with 802.11 and $FS + IR$ from Section 4. We then evaluate its performance in more general topologies, comparing it with 802.11.

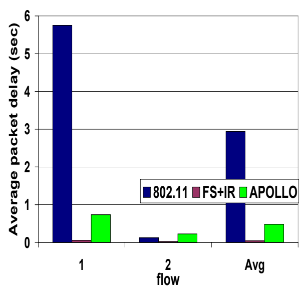
For our simulations we use the Qualnet simulator [23]. In all the scenarios the *two-ray* propagation model is used. For realism, we added random noise creating 10-15% loss in each link. Each simulation was averaged over 10 runs and UDP transport[10] was used. The nominal bit rate in both channels is 2Mbps[11]. The transmission range is set to 250m.

[10]We use UDP to demonstrate the capacity available at the MAC layer under a given protocol. The fact that TCP may not be able to attain the MAC layer capacity due to inefficiencies operating in multi-hop wireless is an orthogonal transport layer design issue.

[11]The bit rate chosen is simply to demonstrate results. We could also choose a higher bit rate 11Mbps and correspondingly a higher rate service plan or a large number of clients.

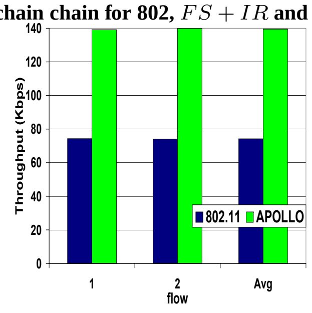


(a) Throughput

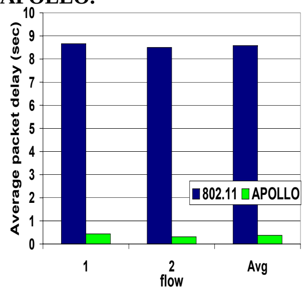


(b) Average packet delay

Figure 5: Performance comparison of 2 asymmetric flows in a 5 node chain chain for 802, $FS + IR$ and APOLLO.



(a) Throughput



(b) Average packet delay

Figure 6: Performance comparison of 2 symmetric flows in a 5-node chain for 802.11 and APOLLO.

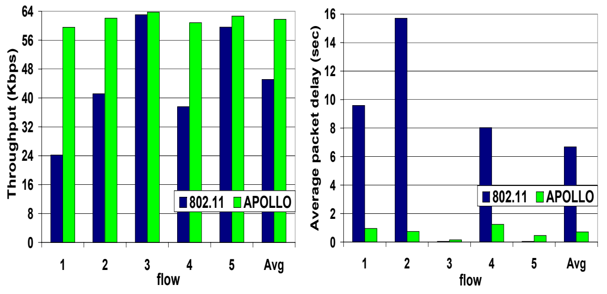
For all the experiments, LQSR [9] is used as the routing protocol. The service plans used in all scenarios are made sure to be achievable according to Section 5.1.

6.1.1 Chain topologies

Asymmetric flows We first use the scenario of Figure 1 with the two asymmetric flows initiated by 2 clients, A and C to gateway E. Each flow has a service plan of 140Kbps. Note that these flows traverse multiple hops and interfere so the overall capacity required to support these clients is close to 1Mbps which roughly corresponds to the actual transport layer throughput obtained on a 2Mbps 802.11 link.

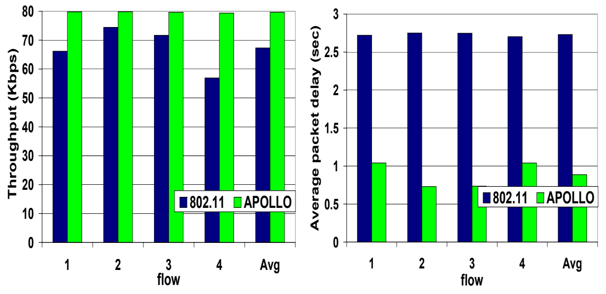
Figure 5 shows that the performance of APOLLO is almost as good as the performance of the optimal $FS + IR$ protocol. Both flows achieve throughputs equal to 139Kbps. Hence APOLLO provides to both clients the service they paid for, without the fine-grained synchronization required for the TDMA-based $FS + IR$. Of course, this lack of fine-grained synchronization affects the average packet delay and APOLLO cannot achieve packet delays as low as $FS + IR$. However, compared to the off-the-shelf approach, APOLLO improves the average packet delay over both flows by a factor of 6.

Symmetric flows We now evaluate APOLLO under a scenario consisting of two symmetric flows (equal distance away from the gateway) in the same chain topology with node C now as the gateway and nodes A and E with a 140Kbps service plan sending packets to node C.



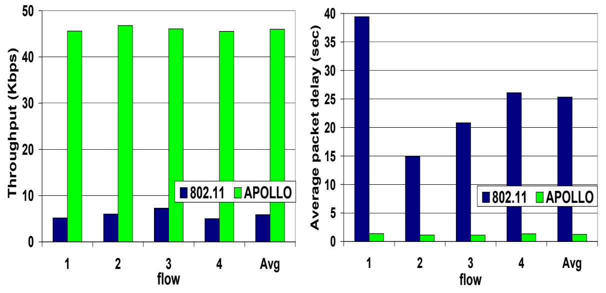
(a) Throughput (b) Average packet delay

Figure 7: **Performance comparison of 802.11 and APOLLO in a P2P scenario.**



(a) Throughput (b) Average packet delay

Figure 8: **Performance comparison of 802.11 and APOLLO in a download scenario.**



(a) Throughput (b) Average packet delay

Figure 9: **Performance comparison of 802.11 and APOLLO in an upload scenario.**

The results in Figure 6 show that, as opposed to the asymmetric scenario, in this case 802.11 is fair, and each flow achieves a throughput of 74Kbps. However, the achieved throughputs are still very far from the service the two clients require – 140Kbps. On the contrary, APOLLO provides again a close to optimal service to both flows, with a throughput of 139Kbps. Also, the average packet delay with APOLLO is 23 times lower than with 802.11. 802.11 suffers from hidden terminal [12] problem in this case while APOLLO is not affected by it since it coordinates the nodes based on queue lengths.

6.1.2 Large networks

We now evaluate the performance of APOLLO in a large network of 49 nodes forming a grid topology in a $1500 \times 1500m^2$ area. Again the distance between any two nodes is such that each node is in transmission range with its 1-hop neighbors, but not with its 2-hop neighbors.

P2P traffic This topology consists of 5 flows with randomly selected distinct sources and destinations , $F1$ – $F5$, each with a 64Kbps service plan.

The performance results in Figure 7 show that APOLLO achieves the client's service plan. On the other hand, 802.11 fails in providing the service plan with $F1$, $F2$ and $F4$ unhappy. The starved flows also have high delays due to which the average delay under 802.11 is 9 times larger than APOLLO.

Flow $F3$ is physically far from the other flows and thus it is relatively unaffected, achieving the maximum throughput, while $F5$ captures the channel more frequently than $F4$ with which it is intersected, resulting in performance degradation for $F4$. Also, flows $F1$ and $F2$ interfere with each other, and with $F4$, and this results in low throughput and high packet delays for both of them. On the other hand, APOLLO can arbitrate better whenever such capture and competition occur in each competing neighborhood.

Download from the gateway This is the most common scenario in a WMN, where clients download traffic from the Internet through the gateway. In this case we have placed the gateway at the center of the network. We have 4 download flows in a cross-shape from the gateway to 4 different clients. Two of these flows travel two hops and the other two travel three hops. Each client has a plan of 80Kbps.

This scenario is less stressing on the MAC layer, since contention is primarily near the source, but not near the receivers of the 4 flows. Since the gateway transmits received packets from the Internet in FIFO order, it fairly schedules its transmissions for all four flows. The results in Figure 8 show that with APOLLO, all four clients get exactly what they pay for. With 802.11 there is less unfairness compared to the P2P scenarios and no flow starves, but the two 2-hop flows ($F2$ and $F3$) achieve higher throughput than the two 3-hop flows ($F1$ and $F4$). Although its performance is improved, 802.11 still fails to provide the requested service. Also, the average packet delays are 2-4 times larger with 802.11 than with APOLLO.

Upload to the gateway This scenario is the inverse of that in the previous paragraph, where clients send traffic to the gateway. Note that this is the most difficult scenario to handle, since contention (i.e. number of senders) increases as packets from all four flows travel towards their common destination. Since, in general, upload traffic is expected to be lower than download, in this scenario each of the 4 clients has a 50Kbps service plan.

The results in Figure 9 show that 802.11 completely fails in this scenario. All four flows starve getting throughputs lower than 8Kbps, only 16% of their service plan. Also, in Figure 9(b) we observe that the average packet delays are extremely high for all four flows, varying between 15 sec and 40sec. On the other hand, APOLLO can fairly deal even with this challenging scenario. All four flows receive similar throughputs, 45-46Kbps, very close to the service plan,

[12]We shut off RTS/CTS as do many operational networks [9, 2, 28].

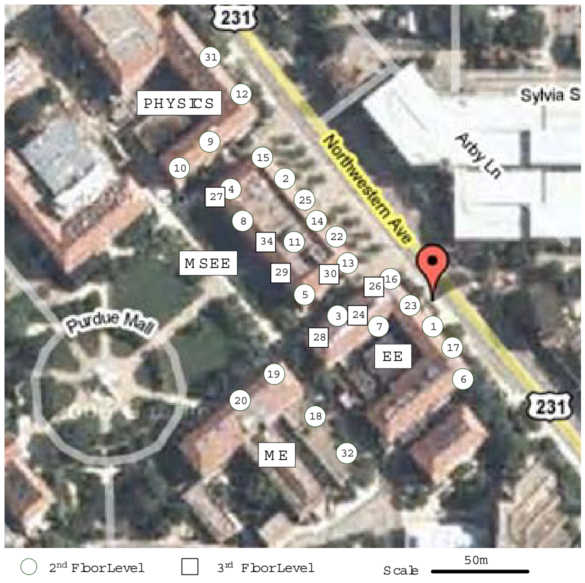


Figure 10: **Top view of the MAP testbed topology.**

and delays are 13 to 28 times lower compared to 802.11. Since the queues of all 4 nodes around the gateway are built equally fast (all flows have the same rate), APOLLO assigns transmission rights to them in a round-robin fashion, eliminating collisions. On the other hand, with 802.11 all nodes compete for the channel, but without any coordination and this finally results in large packet delays, due to exponentially increased backoff times and many retransmissions, and finally to packet drops, since queues are gradually being filled.

6.2 Testbed Evaluation

We now evaluate APOLLO on our wireless network testbed MAP [34], shown in Figure 10. MAP currently consists of 32 mesh routers (small form factor desktops) spread out across four academic buildings on the Purdue campus (EE, MSEE, PHYSICS and ME). Each router has two radios. Each radio is attached to a 2dBi rubber duck omnidirectional antenna with a low loss pigtail to provide flexibility in antenna placement. Each mesh router runs Mandrake Linux 10.1 and the open-source *madwifi* drivers are used to enable the wireless cards. IP addresses are statically assigned. The testbed deployment environment is not wireless friendly, having floor-to-ceiling office walls instead of cubicles as well as some laboratories with structures that limit the propagation of wireless signals. We used channels 1 and 11 of 802.11b to operate our network since they provide the largest frequency separation. Autorate is turned on. The routes used in all scenarios have been found by *OLSR* [5] using the ETX metric.

We evaluate APOLLO in various scenarios and report our experience.

Dominant flow scenario In this scenario, we assume node 1 is the gateway and nodes 22 and 16 generate traffic to node 1. Node 22 can reach node 1 through a 2-hop path ($22 \rightarrow 13 \rightarrow 1$) while node 16 can reach node 1 directly. We assume that clients that use node 22 as their access mesh router can be supported at an aggregate rate of 400Kbps and clients that use node 16 can be supported at an aggregate rate of 450Kbps. We emulate this scenario by generating two UDP flows $F1$ (400Kbps service plan) and $F2$ (450Kbps service plan), from nodes 22 and 16 to 1, respectively, using `iperf`.

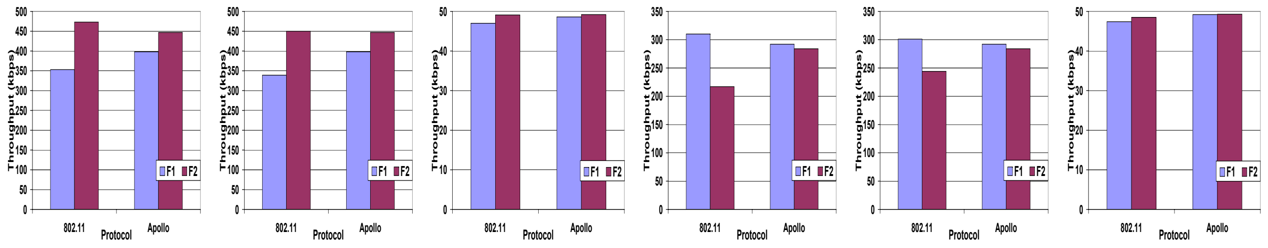
The results in Figure 11(a)(b)(c) show that when clients transmit data close to their service plan in 802.11, $F2$ can achieve throughput equal to the service plan of 450Kbps, but $F1$ receives only 339Kbps. The reason is that $F2$ is a 1-hop flow, while $F1$ is a 2-hop flow. Moreover, from Figure 10 we observe that node 16 is closer to the destination (node 1) than node 13 (the intermediate hop of flow $F1$). Hence, node 16 can capture the channel more frequently than node 13, which also suffers higher packet loss. On the other hand with APOLLO, we observe that both flows achieve throughputs almost equal to their service plans (398Kbps and 447Kbps, respectively). APOLLO detects that node 13 has a larger queue than node 16 and allows it to transmit when it experiences a queue backlog. APOLLO is similar to 802.11 when the plans are underutilized (with sending rate 50Kbps) and better than 802.11 if clients are able to send more than their service plan (with sending rate 800Kbps).

Balanced-flows scenario In this scenario we have two 2-hop UDP flows, $F1$ ($11 \rightarrow 5 \rightarrow 28$) and $F2$ ($20 \rightarrow 18 \rightarrow 28$), each of which has a service plan of 300Kbps. However, now both flows have equal number of hops.

The results in Figure 11(d)(e)(f) also show that with 802.11, when the clients transmit close to their service plan, flow $F1$ achieves a throughput equal to the service plan, while flow $F2$ achieves a throughput of 244Kbps, 81% of the service plan. Thus, unfairness can also manifest between flows of similar hop-lengths due to link quality differences. With APOLLO, the unfairness becomes much smaller and the two flows achieve throughputs of 292Kbps and 284Kbps, respectively, or 97.3% and 94.6% of their service plans. Again, APOLLO is similar to 802.11 when the plans are underutilized (with sending rate 50Kbps) and better than 802.11 if clients are able to send more than their service plan (with sending rate 600Kbps).

TCP upload While there have been many attempts to build new transport protocols for multi-hop wireless networks, TCP is still predominantly deployed in operating systems used on clients and mesh routers. We thus measured TCP performance as part of a service plan. We used three flows $F1$ (Node $1 \rightarrow 28$ with route 1-16-24-28), $F2$ (Node $16 \rightarrow 28$ with route 16-24-28) and $F3$ (Node $24 \rightarrow 28$ with route 24-28).

Table 1 shows the performance of these flows when uploading to gateway node 28 individually under 802.11, combined under 802.11, and combined under APOLLO with a service plan. The results show that when the TCP flows from the 3 clients run together in 802.11, F1 is almost shut off. However, APOLLO is able to enforce a rate plan of 256Kbps at each node. Thus, in this scenario, APOLLO allows multi-



(a) A (Overutilized) (b) A (Close to plan) (c) A (Underutilized) (d) B (Overutilized) (e) B (Close to plan) (f) B (Underutilized)

Figure 11: Throughput comparison of 802.11 and APOLLO in A: dominant flow, and B: balanced flow scenario.

	Auto upload (Kbps)			Auto download (Kbps)		
	802.11 self	802.11 comb.	A-256k plan	802.11 self	802.11 comb.	A-300k plan
F1	274	38	255.2	295	89	293.4
F2	1061	356	256.1	1001	455	297.1
F3	1887	1100	256.1	1765	1200	298.3

Table 1: Throughput achieved by three flows F1, F2 and F3 from a gateway by themselves, combined, and with APOLLO. A-xk refers to an APOLLO x Kbps plan.

hop TCP flows from clients to achieve their service plans.

TCP download We repeated the experiment for the case of downloading from the gateway node. In this case, the three flows are reversed. Again the results in Table 1 shows that APOLLO can support a service plan of 300Kbps at each mesh router thus supporting up to 9 clients with plans of 100Kbps. Unlike in 802.11, APOLLO can allocate service to clients more flexibly, e.g. support more clients at mesh router 1 if needed by reducing clients elsewhere.

Link failure APOLLO has the ability to deal with temporary fluctuations in performance due to route recomputation, external interference, and fading. We emulate such a scenario by injecting a transient link failure artificially in the driver. For this experiment, we sent two UDP flows, from nodes 16 and 22 to node 13, both one hop away from 13. We assume both mesh routers have clients generating traffic of 700Kbps. When running simultaneously, the two flows could both get this throughput. We artificially disturbed the link 22 $\rightarrow$ 13 multiple times during a 70 second period. In both cases, the throughput of node 16 remained unaffected, but throughput of node 22 was different under 802.11 and APOLLO. With 802.11, node 22 achieved only 433Kbps or 62% of its service plan, while with APOLLO the throughput was 657Kbps, or 93% of its service plan. The reason for this difference is as follows. In the intervals following the failures, APOLLO notices that node 22 has a large queue length and it gives it transmission rights for most of the time, allowing it to send the queued packets. The fact that throughput of node 16 is not reduced shows that the capacity is enough for both nodes to achieve 700Kbps. In spite of that, node 22 cannot recover with 802.11. In this case, 802.11 does not take into account the queue built in node 22, and lets the two nodes contend for the channel in the normal link operation intervals. This results in time wasted due to collisions and backoffs, and finally does not give node 22 sufficient time to recover from the failures.

Caveat APOLLO does not guarantee that a TCP flow reaches the bandwidth prescribed by the service plan. APOLLO relies on increased queue length as a symptom. If there are many losses due to interference or bad links, some TCP flows may be unable to increase their window. Similar problems with TCP flows cause low gain from network coding [14]. This is not a problem per se with the MAC layer capacity allocated by APOLLO, but that TCP does not achieve the underlying bandwidth made available. Reducing transport layer unfairness in lossy scenarios remains a place where further research is needed. APOLLO provides an environment where a better designed transport protocol (which can for example distinguish random packet losses from congestion) can achieve good performance.

7. RELATED WORK

There is a lot of work on priority scheduling in wireless ad hoc networks [1, 12, 13, 31]. In [1], three schemes varying different parameters of 802.11 MAC protocol are compared: backoff interval, DIFS, and maximum frame length and the DIFS-based scheme is found to perform well in noisy environments. Another backoff-based priority scheme is described in [12, 13] which piggybacks the priority tag of a node's head-of-line packet onto RTS and data packets. Nodes use information about priorities of all flows in a 2-hop neighborhood to set their backoff. Finally [31] proposes to use uses two narrow-band busy tone signals (BT1 and BT2) to ensure medium access for high priority stations. DIFS periods of high/low priority stations are set accordingly to ensure that low-priority stations in a 2-hop neighborhood will hear BT1 or BT2 before they try to transmit a packet and will defer. In all these works, priorities of different flows are statically assigned based on QoS requirements, while in our case priorities change dynamically, based on queue lengths, in order to achieve a fair service. Hence our scheduling has a more challenging objective, namely to avoid starvation of any flow, as opposed to differentiation mechanisms where the goal is to ensure that high-priority flows will not starve because of a low-priority flow. Moreover, all these schemes are probabilistic and they cannot effectively provide service plans as explained in Section 5.3.

Several works exist on fair scheduling in multihop wireless networks, e.g., [15, 16, 17, 10, 25, 24]. In [15, 16, 17], fair queuing to maximize spatial reuse while ensuring a minimum fair bandwidth allocation for each flow is proposed. The algorithms are centralized and suffer from many of the problems related to TDMA. A core node is required to calculate/propagate the schedule. Moreover, the schedule needs to be recalculated every time a flow comes or leaves the network. In [17], a distributed approach to the fair queuing problem is proposed which is backoff-based (probabilistic), and hence it is not guaranteed that it will perform well in realistic, noisy environments [29]. Also, the fairness model is different and there is no implementation evaluation.

In [10], per-mesh-router (TAP) fairness and a per-TAP fairness scheduling is proposed in which adjacent TAPs exchange periodically information about offered load and link capacities which is used to compute the time shares of different flows on each link and enforce the estimated time shares using rate limiting. The algorithm is only evaluated using simulations and assuming perfect information on offered loads and link capacities. In practice it is not easy to propagate this information, especially when network conditions change rapidly. In contrast, APOLLO requires much less information to be propagated (queue lengths) and we demonstrate its efficiency through a testbed implementation. Additionally, per-TAP fairness does not allocate per-client service plans. The only work that considers a per-mesh-client fairness model is Fair Scheduling (FS) [25], which we described in Section 4. This algorithm also has the same practical problems as TDMA (Section 4).

All the previous algorithms were evaluated only using simulations in ideal environments. *Overlay MAC Layer* (OML) in [24] provides an implementation. OML uses a TDMA-like solution, in which time is divided in slots (with a coarser time granularity) and slots are allocated to nodes according to a WFQ policy, using pseudo random hash functions. OML still suffers from the same problems as TDMA, which we described in Section 4.3. The weights depend on the number of flows, the routing paths used and the type of traffic used. Hence it is very hard or even infeasible (in case of P2P traffic or auto rate adaptation) to assign weights to the nodes in order to support given service plans. In [4], the authors identify severe unfairness between flows in mesh networks. They found that rate limiting improves the fairness of TCP flows. In our work we implement admission control as one of the basic components of APOLLO, and we show that it alone cannot always guarantee a fair service.

8. CONCLUSIONS

This work argues that a “bitrate-for-bucks” service model is key to the successful adoption of WMNs as a last-mile technology. We demonstrate that neither an off-the-shelf approach nor an optimal scheduling approach are viable in providing such a service. We then propose, design, implement, and evaluate APOLLO, a system that implements service provisioning. We believe this work addresses a critical challenge in the deployment and management of WMNs.

9. REFERENCES

[1] I. Aad and C. Castellucia. Differentiation mechanisms for ieee 802.11. In *Proc. of IEEE Infocom*, 2001.

[2] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proc. of ACM MobiCom*, 2005.

[3] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16, 1973.

[4] J. Camp, J. Robinson, C. Steger, and E. Knightly. Measurement driven deployment of a two-tier urban mesh access network. In *Proc. of ACM Mobisys*, 2006.

[5] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L.Viennot. Optimized link state routing protocol (OLSR). RFC 3626, Oct 2003.

[6] D. S. J. D. Couto, D. Aguayo, J. C. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of ACM MobiCom*, 2003.

[7] S. M. Das, D. Koutsonikolas, Y. C. Hu, and D. Peroulis. Characterizing multi-way interference in wireless mesh networks. In *Proc. of ACM WiNTECH 2006*, 2006.

[8] S. M. Das, H. Pucha, K. Papagiannaki, and Y. C. Hu. Studying wireless routing link dynamics. In *Proc. of IMC*, 2007.

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

[10] V. Gambiroza, B. Sadeghi, and E. W. Knightly. End-to-end performance and fairness in multihop wireless backhaul networks. In *Proc. of ACM MobiCom*, 2004.

[11] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu. Impact of interference in multihop wireless network performance. In *Proceedings of ACM Mobicom*, 2003.

[12] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. W. Knightly. Distributed multi-hop scheduling and medium access with delay and throughput constraints. In *Proc. of ACM Mobicom*, 2001.

[13] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. W. Knightly. Distributed priority scheduling and medium access in ad hoc networks. *ACM Wir. Ntws.*, 8, 2002.

[14] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. Xors in the air: Practical wireless network coding. In *Proc. of ACM SIGCOMM*, August 2006.

[15] H. Luo and S. Lu. A topology-independent fair queueing model in ad hoc wireless networks. In *Proceedings of IEEE ICNP*, 2000.

[16] H. Luo, S. Lu, and V. Bharghavan. A new model for packet scheduling in multihop wireless networks. In *Proc. of ACM MobiCom*, 2000.

[17] H. Luo, P. Medvedev, J. Cheng, and S. Lu. A self coordinating approach to distributed fair queueing in ad hoc wireless networks. In *Proceedings of IEEE Infocom*, 2001.

[18] madwifi. `http://madwifi.org`.

[19] R. Nelson and L. Kleinrock. Spatial TDMA: A colllision-free multihop channel access protocol. *IEEE Transactions on communications*, 33(9), 1985.

[20] M. Neufeld, C. Doerr, J. Fifield, T. Weingart, D. C. Sicker, and D. Grunwald. Multimac: An adaptive mac framework for dynamic radio networking. In *Proc. of DySPAN*, 2005.

[21] J. Padhye, S. Agarwal, V. Padmanabhan, L. Qiu, A. Rao, and B. Zill. Estimation of Link Interference in Static Multi-hop Wireless Networks. In *Proceedings of IMC*, 2005.

[22] Y. Qu and A. Srinivasan. Multi-channel olsr with dedicated control interface. In *Proc. of SPECTS*, 2006.

[23] QualNet. http://www.scalable-networks.com.

[24] A. Rao and I. Stoica. An overlay mac layer for 802.11 networks. In *Proceedings of Mobisys 2005*, April 2005.

[25] N. B. Salem and J.-P. Hubaux. A fair scheduling for wireless mesh networks. In *Proc. of WiMesh*, 2005.

[26] J. Shi, T. Salonidis, and E. W. Knightly. Starvation mitigation through multi-channel coordination in csma multi-hop wireless networks. In *Proc. of ACM MobiHoc*, 2006.

[27] S.M.Das, D. Koutsonikolas, and Y.C.Hu. Practical Service Provisioning for Wireless Meshes. Technical report, TR-ECE, Purdue University, November 2007.

[28] A. P. Subramanian, M. M. Buddhikot, and S. C. Miller. Interference Aware Routing in Multi-Radio Wireless Mesh Networks. In *Proc. of WiMesh*, 2006.

[29] N. Vaidya and P. Bahl. Fair scheudling in broadcast environments. Technical report, TR-99-61, Microsoft Research, Dec 1999.

[30] S.-L. Wu, C.-Y. Lin, Y.-C. Tseng, and J.-P. Sheu. A new multi-channel mac protocol with on-demand channel assignment for multi-hop mobile ad hoc networks. In *Proc. of I-SPAN*, 2000.

[31] X. Yang and N. H. Vaidya. Priority scheduling in wireless ad hoc networks. In *Proceedings of ACM Mobihoc*, 2002.

[32] Champaign-Urbana community wireless network. `http://www.cuwireless.net`.

[33] MIT Roofnet. `http://www.pdos.lcs.mit.edu/roofnet`.

[34] `http://www.engineering.purdue.edu/MESH`.

[35] Seattle wireless. `http://www.seattlewireless.net`.