Scalable Real-time Gateway Assignment in Mobile Mesh Networks

Edward Bortnikov   Israel Cidon   Idit Keidar
Department of Electrical Engineering
The Technion, Israel Institute of Technology
ebortnik@techunix.technion.ac.il   {cidon,idish}@ee.technion.ac.il

ABSTRACT

The perception of future wireless mesh network (WMN) deployment and usage is rapidly evolving. WMNs are now being envisaged to provide citywide "last-mile" access for numerous mobile devices running media-rich applications with stringent quality of service (QoS) requirements. Consequently, some current-day conceptions underlying application support in WMNs need to be revisited. In particular, in a large WMN, the dynamic assignment of users to Internet gateways will become a complex traffic engineering problem that will need to consider load peaks, user mobility, and handoff penalties. We propose QMesh, a framework for user-gateway assignment that runs inside the WMN, and is oblivious to underlying routing protocols. It solves the handoff management problem in a scalable distributed manner. We evaluate QMesh through an extensive simulation (mostly of VoIP), in two settings: (1) a real campus network, with user mobility traces from the public CRAWDAD dataset, and (2) a large-scale urban WMN. Simulation results demonstrate that QMesh achieves significant QoS improvements and network capacity increases compared to traditional handoff policies, and illustrate the need for intelligent gateway assignment within the mesh.

1. INTRODUCTION

Wireless mesh networks, or WMNs, are a rapidly maturing technology for providing inexpensive Internet access to areas with limited wired connectivity [9]. While initially designed for small-scale installations (e.g., isolated neighborhoods), WMNs are now envisioned to provide citywide access and beyond through deploying thousands of access points and supporting thousands of simultaneous users [6].

WMN users access the Internet through a multi-hop backbone of fixed wireless routers. Some of these routers, called gateways, are connected to the wired infrastructure. The WMN assigns each user to a gateway upon initial connection, and can migrate it between gateways over time. In traditional implementations, the gateways provide only Internet access. However, QoS-sensitive applications will probably be supported by high-level services at the network edge, similarly to the recent trend in wireline networks [5]. We envision a future WMN gateway that also provides application-level support, e.g., acts as a SIP proxy, a media server cache, or a full-fledged game server [15]. This trend extends the scope of the gateway assignment problem to a large variety of applications and services.

This paper considers gateway assignment – a traffic engineering (TE) problem that seeks optimizing the QoS or fully exploiting the network's capacity for a specific application. The solution must take into account the parameters that incur QoS degradation and additional costs, e.g., network distances and congestion, server (gateway) loads, and application-level handoffs. Mature networking systems employ TE technologies (e.g., MPLS [23]) on top of their existing routing infrastructure, to allow scalability of management. We believe that in future WMN's, traffic engineering solutions like gateway assignment will deployed atop other performance optimizations that are already in place (e.g., multiple radios [10], smart routing metrics [17], etc.).

It is common practice in small-scale WMNs to always assign a user to the nearest gateway (e.g., [11]). In this approach, gateway handoffs (macro-mobility) are tightly coupled with link-layer access point (AP) handoffs (micro-mobility). That is, when a user moves and associates with an AP that is closer to a different gateway than its current one, it automatically performs a gateway handoff too. This simple approach suffers from two drawbacks. First, it cannot adapt to load peaks within the WMN by load-balancing among multiple gateways. Second, it does not consider the application-level impact of such gateway handoffs. For example, in VoIP, handoffs are relatively low-cost, due to a small state associated with a session, whereas in online gaming, the performance penalty of transferring the cached application state between two servers may be very high. Hence, there is a need to decouple AP transitions from gateway handoffs. While the former are purely location-based, application-transparent, and do not incur a high performance impact [11], the latter are not transparent, and should be driven by service-specific QoS considerations.

We propose QMesh (Section 4) – a framework for dynamically managing gateway assignments in future WMNs that can be instantiated with application-specific policies. QMesh is most beneficial for applications that allow gateway handoffs. Traditional applications that do not handle handoffs are supported, but might receive a degraded QoS. QMesh manages two types of decisions for each mobile user: (1) *when* to migrate it between two gateways, and (2) *which* gateway to choose upon a transition. QMesh employs application-specific considerations to balance the tradeoff between two conflicting goals: assigning the user to a gateway that provides it with the best QoS at any given time, and reducing the number of costly gateway handoffs. QMesh does not require any extension of the underlying routing infrastructure, in particular, it does not introduce any non-scalable mechanisms like host-specific routes. Since QMesh makes decisions on a per-user basis, migrating a single user does not directly affect others, thus avoiding traffic oscillations.

QMesh manages gateway handoffs in a scalable distributed way, through a low-overhead signaling protocol that runs within the mesh transparently to the mobile user's networking stack. It *monitors* the QoS of application traffic flows to determine the handoff times, and *probes* the prospective QoS to in a shadow process to select the candidate handoff targets. The key to the protocol's efficiency is its adaptive approach, which performs probing (1) at distances proportional to those required for dissipating the load, and (2) at the frequency required to satisfy the QoS needs. For example, in a low-utilized mesh with little mobility, where a near gateway is likely to provide a good performance, QMesh infrequently performs very few probes limited to the close neighborhood. In contrast, if load is high and current QoS is unsatisfactory, QMesh is more aggressive in probing distant gateways more frequently.

We evaluate QMesh's impact on the application QoS in a WMN through extensive simulations, mostly of VoIP but also of other real-time applications that are more handoff-sensitive (e.g., online games). The studied network topologies and mobility models are described in Section **??**, whereas Appendix A extends on the assumed MAC architecture and the traffic scheduling policies. We first explore a campus-scale WMN (600 APs) with topology and mobility traces drawn from the public CRAWDAD database [3]. Since our main interest is in large-scale networks, we also study a citywide WMN (4000 APs) with highly mobile users. To this end, we experiment with two user populations: (1) a near-uniform distribution, generated by the popular random waypoint (RWP) mobility model [31], and (2) a more realistic distribution biased toward the residential centers, induced by an *alternating weighted waypoint* (AWWP) model for urban traffic [21]. The numerical results demonstrate QMesh's significant advantage over naïve nearest-gateway assignment for all workloads. The QoS achieved by QMesh is close to that of a theoretical Best-Match algorithm that uses instantaneous perfect information. Finally, we show that QMesh adjusts its overhead to workload in a scalable way.

2. RELATED WORK

Handoff optimizations in mobile systems have been extensively addressed since the early 1990's, mostly in the context of cellular networks (e.g., [26]). These studies primarily focused on optimizing the network capacity. Handoffs in cellular networks are triggered by physical metrics, and are handled at the link layer. Our work is different, because we consider the network layer and above. In this context, handoffs are optional, they can improve the QoS over time, but their potential performance hit is not negligible.

Recently, Amir et al. presented a design and implementation of SMesh - a prototype WMN with mobility support [11]. They concentrated on seamless mobility of users between mesh access points. SMesh adopts the nearest-gateway handoff policy, i.e., the users of each AP are automatically assigned to the gateway closest to this AP. This approach is appropriate in a small-size installation described in that paper (about 20 access points and two gateways on the same LAN segment). However, this policy can lead to poor QoS in a wide-area mesh, as shown herein.

Many mature networking solutions address QoS optimizations as as a traffic engineering (TE) problem on top of the existing routing infrastructure (e.g., MPLS in carrier networks [23]). Almost all modern routing protocols (e.g., OSPF [25]) are traffic-independent, thus separating the concern of optimizing the QoS of individual flows to higher-level TE solutions. A different approach, adaptive QoS routing, has been actively studied by the research community (e.g., [20, 24]), originating at Gallagher's seminal work on minimum delay routing [18]. Many load-adaptive routing algorithms are designed for static or quasi-static workloads and suffer from slow convergence in highly dynamic situations. Moreover, they are complex to implement, and their behavior is hard to predict and manage. QMesh's design adopts the first approach for WMNs.

While most TE solutions optimize the unicast QoS, the problem of instantaneously optimal gateway assignment is equivalent to *anycast* routing [32] that seeks connecting each user to some service node among a given set, so as to minimize the average delay. However, we are not aware of any work that handles dynamic anycast of flows with mobile endpoints while considering handoff costs, and proposes scalable real-time solutions.

Adaptive probing of multiple mobile anchor points (MAPs) was proposed in the context of hierarchical mobile IPv6 routing [16]. However, in that work, handoffs are fully dictated by geography (rather than by QoS), and the simulation scale is small (a few MAPs, and a few tens of users). Ganguly et al. [19] suggested a number of VoIP performance optimizations in a WMN. In particular, they proposed maintaining the assignment of each flow to a single gateway, while constantly probing multiple user-gateway paths and opportunistically re-routing the traffic through the best path. Unlike QMesh, this approach tightly couples between gateway selection and routing, and induces non-scalable host-specific paths within the mesh.

We studied a theoretical problem of online assignment of mobile users to service points while balancing between network distances and migration costs [13]. However, that work completely ignored the issue of load. We also addressed the problem of assigning multiple static users to servers so as to minimize the maximum service delay [14] which was modeled as a sum of a network-incurred delay, depending on the number of hops to the server, and a server-incurred delay, stemming from the load on the server. This delay model may be inaccurate for WMNs since it does not consider congestion delays within the network. Moreover, the algorithms presented in both papers are centralized, and their running time is inadequate for real-time systems. In the current work, we use realistic delay models and workloads, and employ a fundamentally different approach of adaptive probing to achieve scalability.

3. DESIGN GOALS

The QMesh framework handles dynamic assignment of mobile users to WMN gateways. We pursue the following goals for this service:

- Satisfying application QoS requirements as closely as possible, in the presence of user mobility.
- Handling a variety of applications with different QoS requirements and handoff penalties.
- Maximizing the service capacity in the presence of load peaks.
- Low-overhead, scalable, and fully distributed network management.
- No proprietary client protocol stack extensions.

4. QMESH FRAMEWORK

In this section, we introduce the QMesh solution, which implements the design goals listed in Section 3. Section 4.1 outlines the QMesh network architecture, and describes the methods and parameters that must be deployed within a WMN to support QMesh. Section 4.2 introduces QMesh's gateway assignment protocol.

4.1 Network Architecture

QMesh provides mobile mesh users with access to real-time application services. The users perceived the WMN as a standard 802.11 LAN, and are oblivious to the mesh's internal multihop structure. At all times, each user associates at the link level with some mesh router within the radio transmission range, called the user's current AP. APs provide basic connectivity within the WMN. As the user moves out of the radio range of its current AP, it associates with a new AP to preserve connectivity. Upon initial connection, QMesh associates each user with a single gateway, which provides it with the high-level service (e.g., Internet access, SIP proxy, or game server). QMesh may later migrate this user to a new gateway when the QoS of the original one becomes poor due to mobility or congestion, while considering an application-specific handoff penalty. QMesh gateway handoffs (macro-mobility) are completely independent of the underlying WMN's AP handoffs (micro-mobility).

Applications that seek optimal QoS must explicitly register with QMesh to receive gateway identity change notifications. This can be done through the application's standard signaling protocol, e.g., SIP. For traditional applications that cannot function correctly in the presence of gateway handoffs, QMesh can be configured to either never re-assign the gateway, or to employ tunneling through the initially assigned one (e.g., [12]), at the cost of QoS degradation. Below, we focus on the former kind of applications.

Application Deployment: QMesh offers a generic framework for supporting multiple applications. The needs of each application are captured by its *service cost* which combines multiple QoS-degrading factors. This cost is accumulated over time. For example, the cost of a VoIP application can be reflected as the number of dropped or late voice packets. We distinguish between *continuous* costs, which stem from network distances and load peaks, and *one-time* costs incurred upon gateway transitions. The gateway assignment algorithm balances the tradeoff between these two kinds of cost. Figure 1 specifies the methods and parameters that applications using QMesh deploy at the mesh nodes.

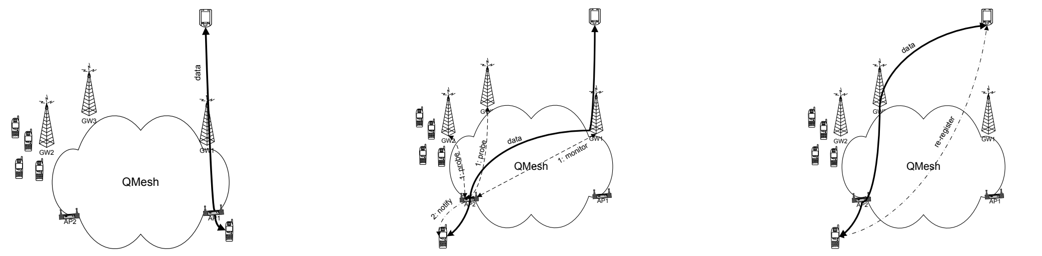
4.2 Gateway Assignment Protocol

QMesh manages gateway handoffs in a fully distributed fashion, by running the assignment protocol independently on each mesh router. Each AP router

Method	**Semantics**	**Example**	**Implementation**
$monitor(u)$	return the *monitored* QoS of user u's gateway.	VoIP delay/jitter	RTCP within the user's flow
$probe(g)$	query the *prospective* QoS of gateway g	VoIP delay/jitter	RTCP over a test connection
$cost(q)$	return the *cumulative* cost incurred by the QoS measure q	VoIP packet loss	

Parameter	**Semantics**
τ_m	Monitoring interval: the rate of running $monitor()$.
T_{min}, T_{max}	The lower and upper bounds on the probing rate (the actual interval τ_p is set adaptively, depending on the QoS level).
P	The number of simultaneous random probes (a larger P can offer better QoS at the cost of higher overhead).
H	Handoff threshold: the cumulative cost since the last transition that triggers a gateway handoff (a smaller H means more aggressive handoffs).
Δ	QoS threshold for the probing rate control (the probes are run more frequently if the QoS is poor).

Figure 1: Methods and parameters deployed at the mesh nodes by applications using QMesh.



(a) Initial connection (b) After micro-mobility (c) After macro-mobility

Figure 2: Handoff of a VoIP session between two NAT gateways in QMesh. (a) Initial assignment to GW1 by access point AP1. (b) Micro-mobility to access point AP2, in parallel with monitoring and probing. (c) Macro-mobility to gateway GW3. GW2 is congested and consequently not selected.

performs the protocol on behalf of its users. Handoff management entails two kinds of decisions for each user, namely, *when* to request a gateway handoff, and *which* gateway to transition to. The first decision is driven by *monitoring* the user's recent QoS (e.g, by tracking the RTCP control packets within a VoIP media flow). The second one is based on *probing* multiple gateways (e.g., in VoIP, the AP-gateway delay can be tested over a low-bandwidth dedicated connection; in an online game, an AP can predict the average request delay by reading the response time statistics from a server, through a remote invocation of a standard application resource monitoring (ARM) API [29]). Monitoring and probing are performed by each AP in the background, transparently to the mobile users. When an AP decides to re-assign some user to a different gateway, it selects the one that offered the best QoS in the last probe.

Figure 2 illustrates a handoff of a media session (e.g., VoIP). The gateways provide an Internet connection service. Each gateway is attached to a different IP subnet, and functions as a NAT router. Initially, the mobile user is served by access point AP1, which associates it with gateway GW1 (Figure 2(a)). The second party resides in the public Internet and communicates with the user through GW1's IP address. The user then moves to access point AP2 (Figure 2(b)), which forwards its packets to GW1 over mesh links. Consequently, the packet latency is degraded. AP2 monitors the session's quality, and in parallel probes gateways GW2 and GW3 for their prospective QoS. At some point, AP2 decides to transfer the user from GW1 to GW3. GW2 is not selected despite its proximity to AP2 because it is currently congested with other users. AP2 sends a notification with GW3's IP address to the user, through the application's natural signaling protocol (e.g., SIP). In parallel, it re-routes the UDP media flow within the mesh via the new gateway (Figure 2(c)). The user re-registers its new IP address with its peer. Before the re-registration is complete, the peer's traffic continues to arrive to GW1, and is dropped there. This loss is the handoff cost.

A handoff management algorithm must balance the tradeoff between two conflicting goals. On the one hand, it would like to always assign each user to the best gateway, in order to minimize continuous costs. On the other hand, one would like to decrease the number of handoffs, in order to reduce one-time costs. QMesh balances this tradeoff by controlling the fraction of one-time costs in the total cost. The algorithm is configured with a *handoff threshold* H. QMesh monitors each user's cumulative cost since the last handoff (not inclusive), and allows a new transition only when this cost exceeds H. For example, if the application-dependent handoff cost is C, then the *total* cost of each assignment period (including the handoff in the end) is bounded by $C + H$, and therefore, the fraction of the handoff cost within the total cost is bounded by $\frac{C}{C+H}$.

The pseudocode of the QMesh assignment protocol appears in Figure 3. Cost monitoring (Lines 4–10) happens every τ_m time units. Once the cumulative cost of user u, denoted $\texttt{cost}[u]$, exceeds H, the user's gateway is re-assigned. $\texttt{cost}[u]$ is tracked by its current AP and sent to the new one upon an AP handoff (Lines 11–13).

The AP runs the gateway selection procedure `nextchoice()` (Lines 24–41) once in τ_p time units, independently of cost monitoring. `nextchoice()` selects the next assignment for *all* local users of the same application jointly. The `GWID` variable holds the selected gateway's identity, and is used upon subsequent handoffs of all users served by this AP. Waiting a long time between invocations results in using stale choices, which translates to suboptimal assignments in dynamic workloads. On the other hand, running `nextchoice()` at a high rate incurs undesirable control overhead. In order to balance between the two, each AP sets the value of τ_p *adaptively*, using the feedback on the quality of the current choice. If the QoS below a configured threshold Δ, then τ_p is exponentially reduced, otherwise, it is linearly increased. The possible values of τ_p are constrained by the lower and upper bounds T_{min} and T_{max}.

Most QoS metrics are distance-sensitive, i.e., an optimal gateway is likely to be near to the user, and the primary reason for picking a remote gateway is network congestion around the close ones. Therefore, QMesh always probes the nearest gateway first, and probes further gateways only if they can help dissipating the local load. More distant gateways are probed only if moving further continues to improve QoS (which happens in case of high load peaks). Remote gateways are randomly load-balanced.

Assume that the distance between the AP and the closest gateway is D network hops. The algorithm works in phases. In phase $i \geq 0$, it probes in parallel P random candidates at distances $2^{i-1}D < d \leq 2^i D$ from the AP. That is, the probed nodes are drawn from concentric rings of doubling width around the AP (the empty rings are skipped, Line 29) – see Figure 4 for illustration. The number of rings is logarithmic with the network diameter ψ, and hence, the worst-case number of probes in a time unit is $\frac{P \log \psi}{T_{min}}$. Note that in the first phase, only the nearest gateway is probed. A gateway chosen multiple times is probed only once. The algorithm stops either if the result of a phase does not improve the result of the previous phases, or if all the rings are sampled. Using a small number of probes is the key to the algorithm's scalability with the network size. We later show through simulation (Section 5) that using $P = 1$ suffices for most workloads, and the average number of probes in a time unit is very close to $\frac{2}{T_{max}}$ – far below the pessimistic upper bound.

```
 1: Initialization
 2:    $\tau_p \leftarrow T_{max}$
 3: {Cost monitoring - per user}
 4: Every $\tau_m$ time do for user $u$
 5:    $\texttt{cost}[u] \leftarrow \texttt{cost}[u] + cost(monitor(u))$
 6:    $\texttt{q}[\texttt{gwid}[u]] \leftarrow monitor(u)$
 7:    if $\texttt{cost}[u] \geq H$ then
 8:       $\texttt{gwid}[u] \leftarrow \texttt{GWID}$, $\texttt{cost}[u] \leftarrow 0$
 9:    end if
10: end
11: upon AP handoff$(u)$ do
12:    $send(\texttt{cost}[u])$ to the new AP
13: end upon
14: Every $\tau_p$ time do
15:    {Gateway selection - shared for all users}
16:    nextchoice()
17:    {Adjust the invocation period}
18:    if $(\texttt{q}[\texttt{GWID}] < \Delta)$ then
19:       $\tau_p \leftarrow \max(\tau_p/2, T_{min})$
20:    else
21:       $\tau_p \leftarrow \min(\tau_p + 1, T_{max})$
22:    end if
23: end
24: procedure nextchoice()
25:    $G' \leftarrow \emptyset$
26:    $D \leftarrow \min_{g \in G} \texttt{distance}(g)$
27:    while $(G' \neq G)$ do
28:       $\texttt{ring} \leftarrow \{g \in G | \frac{D}{2} < \texttt{distance}(g) \leq D\}$
29:       if $(\texttt{ring} \neq \emptyset)$ then
30:          $\texttt{choices} \leftarrow \{P$ random choices from $\texttt{ring}\}$
31:          $\texttt{results} \leftarrow probe(\texttt{choices}) \bigcup \{\texttt{q}[\texttt{GWID}]\}$
32:          $\texttt{best} \leftarrow c$ with the best $\texttt{results}[c]$
33:          if $\texttt{best} \neq \texttt{GWID}$ then
34:             $\texttt{GWID} \leftarrow \texttt{best}$
35:          else
36:             stop
37:          end if
38:       end if
39:       $D \leftarrow 2D$, $G' \leftarrow G' \bigcup \texttt{ring}$
40:    end while
41: end
```

Figure 3: The QMesh gateway assignment.

Note that QMesh's distributed opportunistic assignment policy cannot guarantee the best system-wide cost at all times. For example, an AP in a congested area may start choosing different gateways, thus using longer routes and amplifying the network load in other regions. In some cases, the network may even stabilize in an equilibrium point which is far from optimal. This problem is common to many game-theoretic scenarios (e.g., [27]). However, under the VoIP traffic, most of the congestion happens close to the gateways, and hence, the route length affects the network delay only weakly. Our simulations show that on average, QMesh does not stretch the user-gateway routes by much, and hence, the probability of the worst-case scenarios is small.

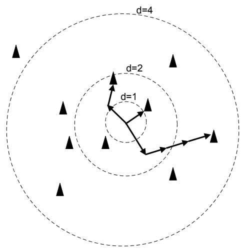


Figure 4: Selecting candidates for a probe in the QMesh assignment protocol. The number of random probes in each phase is $P = 1$. The process stops after probing the 3rd gateway that fails to provide a better QoS than the 2nd one.

5. EVALUATION

We empirically compare QMesh to alternative assignment policies, through extensive simulations. Most of our simulation focus is on VoIP. We study the algorithms' QoS and service capacity, as well as their adaptiveness to mobility and load. Section 5.1 presents our cost model for VoIP QoS evaluation, and Section 5.2 describes two policies that QMesh is compared to.

We first evaluate the protocols in a campus network with real user mobility traces extracted from a public dataset (Section 5.3). However, the scale of this network is around 600 APs, and a limited capacity (150 users). Therefore, we turn to simulating a projected citywide mesh (Section 5.4) with 4096 APs, and address two spatial distributions of mobile users: a near-uniform distribution, as induced by the widely adopted random waypoint (RWP) mobility model [31], and a more realistic distribution with load peaks in residential and business centers, produced by an Alternating Weighted Waypoint (AWWP) model of urban traffic. Finally (Section 5.5), we show the importance of service-specific handoff policies using an example an application which is more sensitive to handoffs (e.g., an online game).

5.1 VoIP Traffic and Cost Model

We consider RTP-over-UDP VoIP flows generated by a standard G.729 codec, i.e., a constant bit rate (CBR) flow of 50 packets per second (20ms inter-packet delay). The typical one-way delay required to sustain a normal conversation quality is 100ms [19]. A VoIP packet is considered lost if it fails to arrive to its destination within an admissible delay. We attribute most of the delay to the mesh infrastructure, and set the admissible threshold to 80 ms, thus allowing a small slack for additional delay incurred by the wired Internet.

We evaluate the VoIP QoS in terms of average packet loss ratio, which is the most dominant component in Mean Opinion Score (MOS) – the standard VoIP quality metric [1]. MOS values range from 0 to 5; values above 3.8 are considered acceptable; values above 4.0 are considered good. For a given workload, we define the service *capacity* as the maximum number of users that can be served within an acceptable MOS. In order to visualize our simple metric, we draw two MOS levels, 4.0 (corresponding to 1% of loss) and 3.8 (2% of loss) on most of our performance plots.

We focus on VoIP calls between mesh users and peers in the public Internet. In this context, a gateway handoff involves a change in the user's external IP address, and triggers application-level signaling to re-route the traffic. This results in one second of connectivity loss, during which all the VoIP packets are lost. Thus, the handoff cost is $C = 50$ (packets).

A VoIP flow starts losing packets if its path to the currently assigned gateway becomes long or congested. Excessive packet delays are the primary reason for continuous loss. Network delay is incurred by accessing the various kinds of mesh links (user, backbone, and gateway connection), and by queuing at the mesh routers. Appendix A extends on the models used by MeshSim. The link-level delays are characterized by the MAC architecture, whereas the queuing delays depend on the VoIP traffic scheduling policy.

In order to allow for large-scale simulations with thousands of users and access points, we developed a flow-level mesh network simulator, MeshSim [7]. Packet-level simulation tools [4, 8] cannot handle such a scale. MeshSim models the delays incurred to VoIP flows at each infrastructure node and link. It uses an accurate 802.11 link delay model [30], and implements two state-of-the-art optimizations: (1) multiple antennae at each node, with channels carefully allocated to minimize cross-link interference, and (2) VoIP aggregation (e.g., [19], and also supported by the 802.11n standard). We describe MeshSim in more detail in Appendix A.

5.2 Assignment Policies

We compare QMesh to two simple assignment policies, NearestGateway and BestMatch. NearestGateway assigns the user to the gateway closest to its current AP. That is, gateway handoffs are tightly bound to AP transitions. The BestMatch policy is a realistically impossible variant of QMesh, which runs the greedy selection procedure upon every AP handoff request, and assumes instantaneous correct information. That is, it performs an exhaustive search of the best candidate rather than random sampling of one, and moreover never uses stale information.

QMesh and BestMatch are instantiated with cumulative packet loss as the QoS cost function. The handoff threshold is set to $H = 10$ packets. This relatively small value is chosen because the handoff cost is low ($C = 50$ packets), and given the user speeds, the loss of 10 packets is a sufficient indication for changing the assignment. QMesh uses a single probe in each phase of `nextchoice`() (i.e., $P = 1$). It adaptively adjusts the interval between invocations of `nextchoice`() within the range $[T_{min} = 1\text{sec}, T_{max} = 15\text{sec}]$. The QoS threshold for accelerating the probes is $\Delta = 50$ ms.

5.3 Campus Scale Simulation (CRAWDAD)

Our first case study is mobile VoIP performance in an unplanned mesh deployed within a large neighborhood or a campus. We draw the network topology and the mobile users' motion traces from CRAWDAD [3], a community resource for archiving wireless data at Dartmouth college, thus avoiding the need to speculate about the simulation's input. The original Dartmouth network is a single-hop WLAN. The network includes over 600 irregularly placed access points. While in a WLAN, APs are connected via a wired infrastructure, in our WMN setting, they communicate through wireless interfaces. All routers use omnidirectional antennas with a transmission radius of 133m – a minimal value for which the network remains connected. We place the Internet gateways in a way that minimizes the mean distance (in the number of hops) from each AP to the nearest gateway. For this purpose, the network is partitioned into 5 clusters using a K-Means algorithm [22], and within each cluster, the router closest to the centroid as selected to serve as a gateway. Figure 5(a) illustrates the WMN's topology (the campus map is due to [2]). The APs are depicted as dark dots, and the selected gateways as triangles with a dot in the middle.

We employ the 2001–2003 movement dataset [28] that contains the mobility traces of more than 6200 users, collected over a period of many months. Each trace contains a sequence of (timestamp, AP id) pairs that describe the history of the user's associations with wireless APs. The majority of users are either static or quasi-static (occasionally hopping between close APs once in a few minutes) most of the time. Their locations are heavily biased toward the faculty buildings.

We explore the scalability of the assignment policies with network load, as follows. For each data point L, we builds a set of scenarios in which L users generate a continuous VoIP stream, as follows. We extract from the trace a set of time intervals, all at least 10 minutes long, in which the number of online users is exactly L. Since the database is very large, each set contains hundreds of intervals for each L. We simulate NearestGateway, BestMatch and QMesh on the traces of 50 intervals selected uniformly at random from each set, and average the loss rates among the runs. Figure 5(b) depicts the results. The loss of BestMatch and QMesh remains acceptable as long as the number of users does not exceed 125 (the service capacity). Therefore, in the absence of mobility, BestMatch and QMesh efficiently balance the costs incurred by network distances and gateway loads. Only under high loads, some differentiation between the two appears, because the latter searches for the candidate more carefully and locates it immediately. On the other hand, NearestGateway cannot handle even 25 users, due to its inability to exploit multiple gateways. QMesh's adaptive nature becomes even more pronounced as we study the dependency between the congestion and the user-gateway distances. For small loads, QMesh and NearestGateway produce an identical average distance of 2.1, while for high loads, QMesh stretches the routes to 4.9 to optimize the assignment.

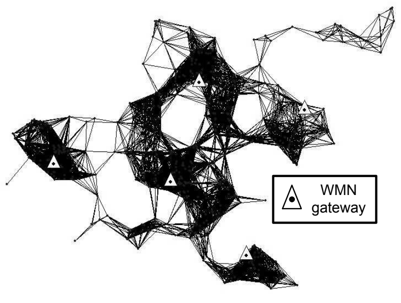
Following this, we examine QMesh's scalability in the presence of concurrent TCP flows generated by traditional data applications. We repeat the previous experiment, for a varying number of TCP connections (0% to 20% of the number of users, with the rest running VoIP flows). All TCP flows are handled in a traditional way, namely, each of them is initially assigned to the closest gateway, and never reassigned again. In order to prevent starvation of the VoIP traffic by TCP flows, we allocate the latter with at most 50% of available transmission bandwidth, and schedule their packets at a lower priority. Thus, the VoIP capacity of the shared links decreases, but the QoS of the admitted flows is guarantee. Figure 5(c) shows that the average loss ratio increases with the fraction of TCP flows, but the impact is not dramatic within the admissible load range.

5.4 City Scale Simulation

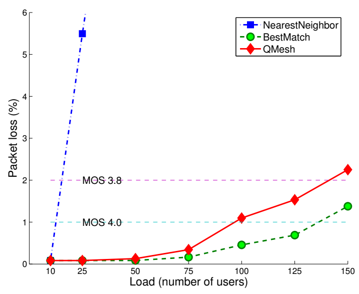
Our ultimate goal is studying the performance of QMesh in a very large-scale WMN with highly mobile users. For this, we turn to simulating a citywide mesh that exceeds the campus deployment by an order of magnitude in the spanned area and the population.

We consider an urban geography of size 8×8 km^2. There are five population areas – four residential neighborhoods and a commercial downtown. User locations within each area follow a Gaussian distribution around the area's center with variance σ, which is called the area's *effective radius*. The downtown's effective radius is 1km, and its center is co-located with the center of the grid at coordinates (4km, 4km). Each neighborhood's effective radius is 500m, and their centers are located at coordinates (1km,1km), (1km, 7km), (7km, 1km), and (7km, 7km). Figure 6(a) depicts this topology. Areas are depicted as circles, and gateways as small triangles. The Internet access is provided through a regular grid of 64 gateways, spaced 1km apart. The wireless backbone is a fine grid of 4096 mesh routers, spaced 125m apart. The transmission radius is 125m. Our simulation employs two stationary distributions of mobile users, each generated by a different mobility model:

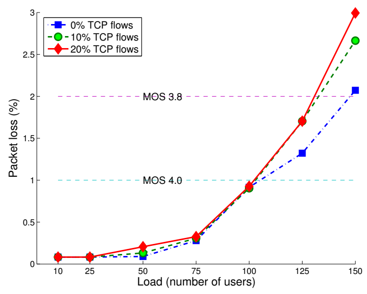
1. A near-uniform distribution, produced by the popular random waypoint model (RWP) [31]. The node uniformly chooses the destination and moves toward it at a constant speed $v = 20$ m/s (an urban driving speed).
2. A more realistic distribution that biases the users toward the population areas (e.g., neighborhoods



(a) The Dartmouth network map and gateway placement



(b) Loss ratio scalability: BestMatch and QMesh vs NearestGateway



(c) Loss ratio scalability: QMesh in the presence of TCP flows

Figure 5: Scalability evaluation of the gateway assignment algorithms in an unplanned campus WMN, with topology and user mobility traces drawn from the Dartmouth CRAWDAD public dataset.

or downtown), produced by the projected alternating weighted waypoint (AWWP) model. At any given time, a mobile node is either stationary in some area, or moving on a highway between two areas at a constant speed $v = 20$ m/s. The popularity of different areas varies during the day.

5.4.1 The Alternating Weighted Waypoint Model

AWWP is one plausible way to create a clustered user distribution. It is inspired in part by [21], which explored preferences in choosing destinations of pedestrian mobility patterns. The nodes' transitions between the areas are governed by a Markov process that switches its transition probability matrix every 12 hours. The system is modeled by two super-states, each of which is a Markov chain. Each state in a chain corresponds to a single area. Each probability matrix designates the users' preferred locations at a certain time of day. The moving node's destination point within the target area is a random variable, drawn from the Gaussian distribution described above. In the morning, most users drive to the downtown and stay there during the working hours, whereas in the evening, most users drive back to their neighborhood and stay at home during the night. Direct transitions between the neighborhoods are not allowed.

Figure 6(b) depicts this random process. We denote the downtown by D, and neighborhood i by N_i. The transition probabilities are (symmetric for all i):

	Morning/day	**Evening/night**
$p_{D,D}$	0.9	0.1
p_{D,N_i}	0.025	0.225
$p_{N_i,D}$	0.9	0.1
p_{N_i,N_i}	0.1	0.9
p_{N_i,N_j}	0	0

The stationary distributions of the Markov chains are:

	Morning/day	**Evening/night**
π_D	0.9	0.1
π_{N_i}	0.025	0.225

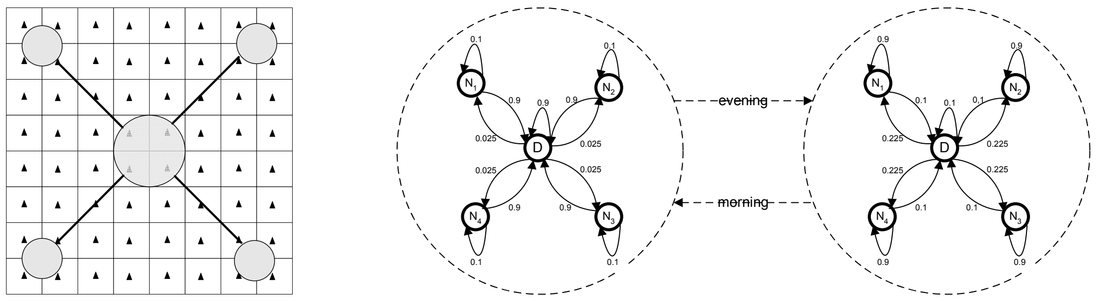
A mobile user's behavior is deterministic between transition times. Upon a self-transition, a node remains at its current location for a period of t. In case of a transition of the user to another area, it picks a destination point from the distribution induced by the destination area, and moves to it with a speed of v. For simplicity, we assume that all users wait for the same time and move with the same speed. We set $t = 4$ min. Note that the waiting time is equal to the driving time between the centers of the downtown and neighborhood areas. In this setting, the motion can be approximated as a discrete-time Markov chain, in which the time slot length is 4 min. All state transitions (including the probability matrix switch) happen on slot boundaries. During a single slot, the user either moves between two areas, or remains in one of them.

In each super-state (day or night), the users are mostly stationary, except in a short time after the transition, when they mostly move to their new preferred areas. Upon switching the super-state, the convergence to a new matrix's stationary distribution is short (3-4 time slots). Therefore, the 15 min following the super-state transition are considered a *transition period*, after which the system enters a *stable* period.

We also experimented with richer models, e.g., non-straight movement trajectories, and constrained motion within the population areas. However, they yield almost the same results because the most important factor is the load peaks. Hence, our simulations focus on the presented simple model.

5.4.2 Numerical Results

We compare the loss rates and overhead of QMesh to BestMatch and NearestGateway, for the near-uniform and skewed stationary distributions produced by the RWP and AWWP mobility models, respectively. Every data point is averaged over 20 runs. For AWWP, we separately study four different times of day: morning (neighborhoods-to-downtown movement), day



(a) City topology (b) Alternating Weighted Waypoint

Figure 6: Urban Simulation Settings: (a) The city's topology (downtown and four neighborhoods) and the gateway grid. (b) The random process behind the AWWP mobility model.

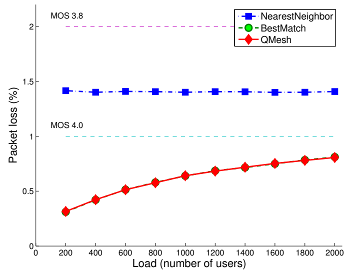


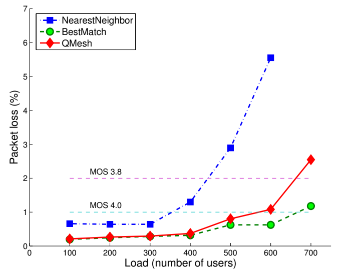
Figure 7: Scalability evaluation of the gateway assignment algorithms in a citywide WMN, for a near-uniform distribution (RWP model).

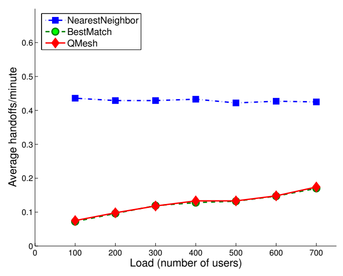
(mostly staying in the downtown), evening (downtown-to-neighborhoods movement), and night (mostly staying in the neighborhoods). Day and night are stable periods, morning and evening are transition. The morning and evening scenarios are simulated for 15 min (the transition period time, see Section 5.4.1). The day and night scenarios are insensitive to the measurement period; we used 30 min periods for them. The RWP experiments were initialized with the uniform distribution of users, and preserved it over time [31]. Each experiment simulated 15 min of user motion.

We first study the the dependency between load and loss for the three algorithms. Figure 7 depicts their behavior for near-uniform distribution induced by the RWP mobility pattern, with loads ranging from 200 to 2000 users. At all times, NearestGateway succeeds in accommodating each user at the closest gateway, because no cell's load exceeds its capacity. All loss is due to handoffs, and depends only on the user's speed, and hence, it is constant for all loads. The BestMatch and QMesh policies incur identical costs, since upon a handoff, the local gateway is almost always the best choice that cannot be improved by further probing. They improve the loss over NearestGatewayby sustaining a user's association with its gateway beyond the grid cell's boundaries, as long as the QoS permits. The maximal admissible user-gateway distance diminishes with load, and hence, handoffs become more frequent, thus causing BestMatch's and QMesh's loss rates.

The shortcomings of NearestGateway become evident as we apply the same experiment for a more realistic biased distribution of load generated by the AWWP model. We separately explore the morning scenario featuring a transition of load from the periphery to the center (Figure 8(a)), and the day scenario that reflects a stationary congestion in the downtown (Figure 8(c)). In both cases, NearestGateway does not scale beyond 300 users due to its inability to resolve the congestion in the downtown area to the other gateways. On the other hand, QMesh can accommodate 600 users – just slightly below the baseline BestMatch. Figure 8(b) differentiates the part of handoffs in the packet loss (by depicting the average handoff frequency), in the morning scenario. QMesh's frequency is low and congestion-adaptive (growing slowly with load), while NearestGateway's is high and load-insensitive.

In the next experiments, we continue using the more challenging AWWP workload. Figure 9(a) depicts the distribution of costs achieved by NearestGateway, BestMatch and QMesh by the time of day, for a load of 600 users. Note that NearestGateway's loss is even higher during the day than in the morning, due to the stationary congestion in the downtown. The price of this congestion is higher than the cost of excessive handoffs during the morning transition. Since the measured transition period also captures some resting time in the steady-state area for most nodes, NearestGateway's loss in the morning is higher than in the evening, when these areas are not congested. The same disadvantage of NearestGateway is observed when we examine the relationship between a user's mobility level (the fraction of time in which the user changes its location) and its loss rate. Figure 9(b) and Figure 9(c) depict the distribution of loss among the mostly stationary users (below 20% mobility) and the mostly mobile ones (above 20%) achieved by NearestGateway and QMesh, respectively. (Note that a small fraction of users remains highly mobile even in the stable regime, since transitions between population centers are not instantaneous). QMesh has the desirable property that the stationary users experience smaller loss rates than the mobile ones. That is,


(a) Loss ratio, transition period


(b) Handoff frequency, transition period

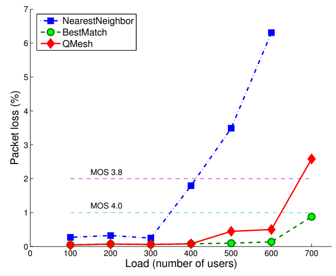
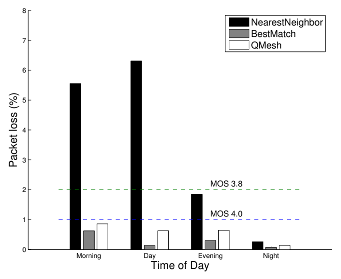
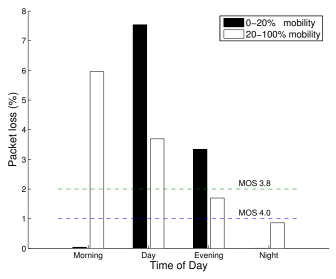

(c) Loss ratio, stable state

Figure 8: Scalability evaluation for a clustered distribution (AWWP model): (a) Loss ratio – morning. (b) Handoff frequency (average number of handoffs per minute) – morning. (c) Loss ratio – day.


(a) QMesh vs NearestGateway and BestMatch


(b) NearestGateway: stationary vs mobile users

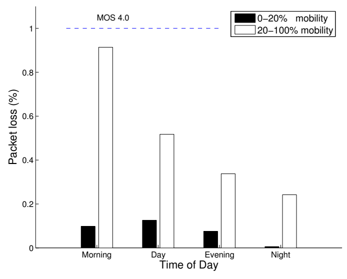

(c) QMesh: stationary vs mobile users

Figure 9: Average loss ratio distribution by the time of day, for the a skewed workload of 600 users (AWWP mobility model): (a) Comparison between 3 assignment policies, (b,c) Comparison between the mostly stationary (below 20% mobility) and the mostly mobile users, for two separate policies.

most of the mobile users' packet loss stems from handoffs (which do not happen to the stationary users), while the congestion-oriented loss is minimized for both categories thanks to opportunistic assignment. In contrast, under NearestGateway, stationary users in congested areas suffer from continuous loss, which exceeds the occasional handoff-related loss incurred to mobile users.

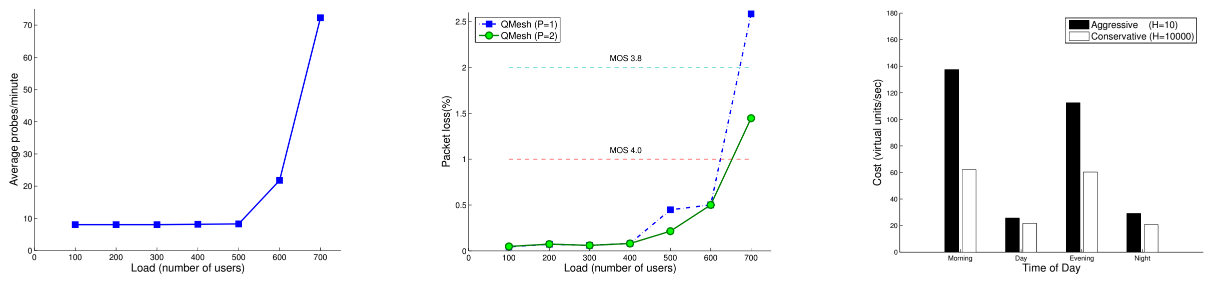
Following this, we examine QMesh's control overhead – the average number of probes per minute performed by each AP. We focus on the day scenario when the network congestion is most heavy. The overhead depends on the number of probes per selection as well as on the probing rate. Our measurements show that for most values of load, it is enough to apply `nextchoice()` once in 15 seconds to achieve an acceptable loss ratio. The average number of probes applied upon gateway selection never exceeds 2.5, as opposed to the theoretical limit of the logarithm of the network size. Moreover, for most values of the load, the number of probes is almost exactly 2 – the minimal possible value. Figure 10(a) summarizes these results in a single plot, which shows that the overhead is very small for most workloads.

Finally, we study the potential QoS benefit of increasing the number of random probes made by QMesh. We compare two instantiations of the algorithm using $P = 1$ and $P = 2$, in the day scenario. Figure 10(b) shows that increasing P does not bring any performance impact for light loads (below 400), and has a minor impact for heavy loads. Moreover, QMesh partially masks the disadvantage of applying a single random probe by adaptively adjusting the probing interval τ_p. Note that at a load of 600, QMesh with $P = 1$ starts increasing its probing rate due to QoS degradation, which reduces the gap between it and QMesh with $P = 2$.

5.5 Service-Specific Handoff Policies

In all the above experiments, QMesh used a very low handoff threshold, and migrated each user almost immediately as the user's delay became inadmissible. Setting a low threshold ($H = 10$) was correct because the handoff cost was also low ($C = 50$), and hence, there was no benefit in delaying the new assignment. However, this policy is not necessarily true if the handoff cost is very high, e.g., in an online game, in which a handoff entails a substantial state transfer. Consider, for example, the same traffic model as described in Section 5.1, the same continuous cost (1 lost packet = 1 unit), and the handoff cost of $C' = 50000$ units. We provide this example for insight only, and do not claim that a realistic online game's traffic/cost model is used.

Figure 10(c) illustrates the comparison between two instances of QMesh parametrized by $H = 10$ and $H = 10000$, respectively, under a light load (400 users).



(a) Scalability of the number of probes (b) Impact of P on the loss rate (c) Impact of H on the loss rate

Figure 10: Studying the effect of QMesh's tuning parameters: (a) Scalability of the number of probes per minute with load, (b) Impact of increasing the number of simultaneous probes P. (c) Impact of handoff threshold for an application with a high handoff cost (50000): aggressive policy ($H = 10$) vs. conservative policy ($H = 10000$).

The second instance, which is much more conservative in applying costly handoffs, consistently achieves a better cost with all mobility patterns; i.e., tuning the handoff threshold in accordance with the application-specific handoff cost is crucial for achieving a good overall cost.

6. CONCLUSIONS

Future mesh networks will be expected to accommodate a high capacity of mobile users running media-rich applications. In order to satisfy the QoS requirements of such applications, gateway assignment policies will need to take into consideration factors like load peaks, mobility, and application-specific handoff costs. Future WMN architectures will need to employ scalable mechanisms to this end. We introduced QMesh, a novel scalable solution for dynamic assignment of mobile users to gateways in a large-scale WMN, which can be instantiated with application-specific handoff policies. We studied QMesh through simulation in different settings of a wide-area urban WMN. Our results show that QMesh scales well and adapts to network loads. It satisfies application QoS requirements for service capacities significantly exceeding those of traditional policies.

7. REFERENCES

[1] `http://davidwall.com/MOSCalc.htm`.

[2] `http://www.dartmouth.edu/~maps`.

[3] CRAWDAD: a Community Resource for Archiving Wireless Data at Dartmouth. `http://www.crawdad.cs.dartmouth.edu`.

[4] JiST/SWANS - Java in Simulation Time. `http://jist.ece.cornell.edu`.

[5] Riverbed Technology. `http://www.riverbed.com`.

[6] Strix Systems. `http://www.strixsystems.com`.

[7] The MeshSim Simulator and Traces. `http://comnet.technion.ac.il/magma/software/meshsim.tar.gz`.

[8] The Network Simulator – ns-2. `http://www.isi.edu/nsnam/ns`.

[9] I.F. Akylidiz, X. Wang, and W. Wang. Wireless Mesh Networks: a Survey. *Computer Networks Journal (Elsevier)*, March 2005.

[10] M. Alicherry, R. Bhatia, and Li (Erran) Li. Joint Channel Assignment and Routing for Throughput Optimization in Multi-Radio Wireless Mesh Networks. *ACM MobiCom*, 2005.

[11] Y. Amir, C. Danilov, M. Hilsdale, R. Musaloiu-Elefteri, and N. Rivera. Fast Handoff for Seamless Wireless Mesh Networks. *MobiSys*, 2006.

[12] Y. Amir, C. Danilov, R. Musaloiu-Elefteri, and N. Rivera. An Inter-domain Routing Protocol for Multi-homed Wireless Mesh Networks. *IEEE WoWMoM*, 2007.

[13] E. Bortnikov, I. Cidon, and I. Keidar. Nomadic Service Assignment. *IEEE TMC*, 6(8), August 2007.

[14] E. Bortnikov, I. Cidon, and I. Keidar. Scalable Load-Distance Balancing. *Pelc, A. (ed.) DISC 2007, LNCS 4731, Springer-Verlag*, 2007.

[15] J. Chen, B. Knutsson, B. Wu, H. Lu, M. Delap, and C. Amza. Locality Aware Dynamic Load Management form Massively Multiplayer Games. *PPoPP*, 2005.

[16] W. Chung and S. Lee. Improving Performance of HMIPv6 Networks with Adaptive MAP Selection Scheme. *IEICE Trans. Comm.*, E90-B(4), 2007.

[17] R. Draves, J. Padhye, and B. Zill. Routing in Multi-radio, Multi-hop Wireless Mesh Networks. *ACM MobiCom*, September 2004.

[18] R. G. Gallagher. A Minimum Delay Routing Algorithm Using Distributed Computation. *IEEE ToC*, 25:73–84, 1977.

[19] S. Ganguly, V. Navda, K. Kim, A. Kashyap, D. Niculescu, R. Izmailov, S. Hong, and S. Das. Performance Optimizations for VoIP Services in Mesh Networks. *JSAC*, 24:2147–2158, November 2006.

[20] R. A. Guerin and A. Orda. QoS Routing in Networks with Inaccurate Information: Theory and Algorithms. *IEEE/ACM ToN*, 1999.

[21] W. Hsu, K. Merchant, H. Shu, C. Hsu, and A. Helmy. Weighted Waypoint Mobility Model and its Impact on Ad Hoc Networks. *ACM MC2R*, 9:59–63, 2005.

[22] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[23] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the Tight Rope: Responsive yet Stable Traffic Engineering. *ACM SIGCOMM*, 2005.

[24] C. R. Lin and J-S. Liu. QoS Routing in Ad Hoc Wireless Networks. *IEEE JSAC*, 17, 1999.

[25] J. Moy. OSPF Version 2. Technical report, IETF. `http://www.ietf.org/rfc/rfc2328.txt`.

[26] G. P. Pollini. Trends in Handover Design. *IEEE Communications Magazine*, 34, 1996.

[27] T. Roughgarden and E. Tardos. How Bad is Selfish

Routing? *Journal of the ACM*, 2002.
[28] L. Song, D. Kotz, R. Jain, and X. He. Evaluating Location Predictors with Extensive Wi-Fi Mobility Data. *IEEE INFOCOM*, 2004.
[29] The Open Group. Application Response Management - ARM. `http://www.opengroup.org`.
[30] O. Tickoo and B. Sikdar. Queueing Analysis and Delay Mitigation in IEEE 802.11 Random Access MAC Based Wireless Networks. *INFOCOM*, 2004.
[31] J. Yoon, M. Liu, and B. Noble. Sound Mobility Models. *ACM MobiCom*, 2003.
[32] W. T. Zaumen, S. Vutukury, and J. Garcia-Luna-Aceves. Load-Balanced Anycast Routing in Computer Networks. *ISCC*, 2000.

APPENDIX

A. DELAY MODELING IN MESHSIM

We briefly describe the delay models used by MeshSim (available for download at [7]), a flow-level simulator we developed to provide network scalability beyond that of packet-level simulation tools [4, 8].

MAC Architecture and Link Delays: We assume that each router is equipped with distinct interfaces for user access (802.11b) and backbone (802.11a) communication. These interfaces use different wireless bands, and hence, the access and backhaul traffic flows do not interfere. 802.11a is chosen for its abundance of orthogonal wireless channels (12), which are exploited to minimize interference among the mesh links (this is also a common practice in commercial WMNs [6]). A router employs two cards for communicating within the mesh - one for egress traffic and the other for ingress traffic. This facilitates a parallel transmission and reception at the backbone, and hence, a simultaneous upstream and downstream forwarding. The ingress interface is operated at a fixed wireless channel. Whenever a router needs to communicate with some neighbor, it switches its egress interface to the channel of this neighbor's ingress card. Hence, a single ingress interface is shared by the links emerging from the router's neighbors.

The low-degree topologies utilized by our experiments and a substantial number of available channels allow performing ingress channel assignment in a way that no pair of routers within two hops from each other share the same ingress channel. Therefore, the only kind of MAC contention at the backbone arises when two nodes simultaneously transmit to the same neighbor. That is, we assume that no interference exists between two backbone links without a common endpoint.

Since at each mesh node, all the incoming backbone links share the same ingress interface, the delay on each outgoing link depends on the cumulative load on this link's target. The mesh forwards each flow along the shortest path between its AP and gateway. Therefore, a particular assignment of users to gateways determines the load on each link, and hence, the total link delay incurred to each user. We use the model by Tickoo and Sikdar [30] to compute the expected latency of traversing a shared 802.11 link (either access or backbone).

VoIP Aggregation and Queueing Delays: We assume that VoIP flow aggregation(e.g., [19], also adopted by 802.11n) is employed in order to overcome the capacity limitation that is inherent to wireless VoIP, namely, a high overhead of transmitting small packets over the 802.11 medium. The VoIP traffic at mesh routers is handled through a VoIP-specific scheduling policy. A packet that needs to be forwarded over an egress link is placed into the queue of this link. The link's scheduler sets the time for transmitting the next outgoing packet. At this time, the queued packets are aggregated into a super-packet, which is transmitted over the medium as a single frame. Upon arrival to the neighbor, the super-packet is de-multiplexed, and the individual packets are handled independently.

By rate-limiting the super-packet generation process, the scheduler controls the capacity/delay tradeoff at the wireless link. The scheduler transmits a single packet in a fixed-length time slot, which can be implemented, e.g., through a simple token-based traffic shaping. With this policy, if the arrival rate exceeds the transmission rate, the packets are queued on average for a half-slot time, and otherwise, they are forwarded immediately.

In the chosen delay model [30], a link can sustain an inter-packet delay of 20ms for at most 10 independent flows without dropping packets. For the backbone links, we take a conservative approach, and rate-limit each egress queue to one packet in 10ms. Since the maximal node degree is 4, at most 8 (aggregated) packets contend for each shared ingress link in 20ms, thus approximating the behavior of eight concurrent VoIP flows. The average queueing time is therefore 5ms for a fully backlogged egress queue.

The maximal capacity of the backbone links is constrained by the number of RTP packets that can be multiplexed into a single super-packet. The size of an RTP packet with a G.729 voice payload is 60 bytes. Assuming the super-packet size of 1500 bytes, without RTP header compression [19], the number of voice packets that can be multiplexed into a super-packet is 25. Since a single egress queue schedules transmissions each 10ms (twice the packet arrival rate in a single flow), its capacity is $2 \times 25 = 50$. Hence, the capacity of a shared ingress link is $4 \times 50 = 200$ flows (4.7 Mbps bandwidth).

Finally, the gateway connection introduces its own delay, which depends on the wired link's capacity. Since a typical WMN is expected to use an available inexpensive wired infrastructure, assume the use of the ADSL technology, in which the uplink is the bandwidth bottleneck. The fastest available ADSL2 uplink rate today is 3.5 Mbps. We assume that it supports 120 flows (2.75 Mbps effective bandwidth), and employ the M/M/1 model for delay calculation.