Abstract—In this paper we investigate link scheduling for Wireless Mesh Networks (WMNs) carrying real-time (i.e., delay-constrained) traffic. We show that the problem of computing a conflict-free link schedule with end-to-end delay constraints can be formulated as a mixed-integer non linear problem that can be optimally solved in reasonable time (i.e., minutes) for relatively large WMNs (up to 20-30 nodes). We use the above result to explore the schedulability region of a WMN with a given routing and input traffic, assessing whether and when aggregating flows which traverse the same path makes a given input flow set schedulable. Furthermore, we devise a heuristic solution strategy, which computes good suboptimal solutions within up to few seconds, thus being amenable for online admission control.

Keywords—Link Scheduling; Wireless Mesh Networks; Real-time Traffic; Network Calculus

I. INTRODUCTION

Wireless Mesh Networks (WMNs) [1] represent one of the most promising technologies to provide broadband access to mobile clients located at the edge of wireline networks, or in remote, rural, or non cost-effective areas, e.g. offices and home environments. In WMNs, end-users are served by stationary mesh routers, connected through wireless links. Moreover, some mesh routers are generally connected to the Internet through wires, and thus act as gateways for the entire WMN. Communication between mesh routers is multi-hop, with intermediate routers acting as relays for endpoints not in the transmission range of each other. Many of the radio resource management issues in a WMN are common to multi-hop wireless networks. However, as mesh routers are fixed, problems such as energy consumption (typical of ad hoc networks) are no longer an issue. This makes it sensible to opt for a centralized network management, as opposed to the distributed approaches used for ad hoc networks. In this case, nodes are coordinated by a network entity which determines the management based on the global knowledge of the topology and on additional conditions.

Each mesh router broadcasts packets, which are received by all neighbors tuned on the same frequency and within the transmission range. To avoid signal interference at mesh routers not meant to be the receivers, link scheduling is used to guarantee conflict-free operation in the context of Time Division Multiple Access (TDMA, [2]), where time is slotted and synchronized. Through link scheduling, only sets of non-interfering links are activated simultaneously in each slot. Cross-layer approaches where link scheduling and routing are jointly addressed have been extensively studied [3]-[7] in the past few years due to their application to TDMA MAC protocols. However, to the best of our knowledge, few works published so far have considered end-to-end delay bounds as constraints on link scheduling. Real-time applications such as voice, video, or mission critical, require in fact firm guarantees on their maximum end-to-end delay. As WMNs are already and will be supporting these types of traffic, computing conflict-free link schedules that guarantee delay bounds is necessary to provide users with a suitable Quality of Service.

In this paper, we formulate and solve a link scheduling problem for leaky-bucket constrained flows on a TDMA WMN. Each flow traverses a pre-established route and requires an end-to-end delay bound guarantee. As a first contribution we show that, if flows are not aggregated as they travel within the network (i.e., at core mesh routers), this problem can be formulated as a mixed integer-nonlinear programming problem, whose objective is to minimize the maximum delay violation, i.e. the distance between the guaranteed and the required end-to-end delay bounds. This class of problems can be solved optimally (at least for WMNs of few tens of nodes) by a general purpose solver, which implies that a feasible schedule can be found if there exists one. For these relatively large WMNs, a solution can be computed in the order of minutes or hours, which is affordable in the perspective of network provisioning timescales. Minimizing the maximum violation produces robust schedules, where the parameters of some flows can be varied (even by large amounts) with limited impact on the delays.

As a second contribution, we investigate how aggregation of flows traversing the same path affects the schedulability. The traffic of a set of flows traversing the same path can be

978-1-4244-7265-9/10/$26.00 © 2010 IEEE
buffered in either a single per-path aggregate queue or separate per-flow queues. We show that the schedulability of a set of flows depends on flow aggregation, and we propose guidelines to assess if and when to aggregate flows in a WMN.

As a third contribution, we consider abating the link scheduling computation time, so as to make it viable for online admission control. We devise a heuristic which allows suboptimal schedules to be computed within a tunable time, e.g. up to few seconds. Furthermore, capitalizing on the intrinsic robustness of our solutions, we show that it is not always necessary to alter the link scheduling when new flows join the network, and we describe how to indentify such cases at a negligible computational cost.

Few related works exist on delay-constrained link scheduling. In [8], authors study 802.16-based WMNs, with the objective of minimizing the delay. However, there is no indication that the algorithm devised therein actually achieves this purpose. Other works ([9]-[11]) consider a different, though apparently close objective, namely that of minimizing the overall TDMA delay introduced by multi-hop transmission. Some works ([9]-[10]) describe heuristic algorithms, with no optimality guarantee: [9] considers both CBR (voice) and VBR (video) flows, assuming that their traffic statistics are known, while [10] only accounts for CBR traffic, under the questionable assumption of non interfering channels. In [11], instead, a WMN is modeled as a stop-and-go system and a min-max problem on the round-trip TDMA delay introduced by link scheduling is formulated and optimally solved. However, minimizing the TDMA delay is a different problem with respect to computing delay-constrained link schedules, and – as we show later on in Section V – even optimal solutions of the former are often infeasible for the latter. This is even more likely with bursty traffic, e.g. video, for which queuing delay (not accounted for in [11]) is often the largest component. Other works ([12]-[14]) derive delay bounds in wireless (sensor) networks, using Network Calculus [15], i.e. the same technique exploited in this paper. These works, however, compute delay bounds given a network configuration, whereas our aim is the opposite, i.e. to configure the network (i.e., its link scheduling) based on delay constraints. A recent work of ours, [16], marked a first step in this direction. In that work, sink-tree WMNs, where aggregation is in place at core routers, were analyzed. It is shown therein that it is very hard to solve the link scheduling problem optimally in the above settings (except for trivial instances). In fact, [16] proposes a heuristic solution approach. On the other hand, the analysis in [16] can only be applied to sink-tree WMNs, as the delay bound expression only holds for that topology. Although sink trees are common, they are not the only case of interest.

The rest of the paper is organized as follows: Section II reports the system model, and the problem is formulated in Section III. We analyze the properties of optimal schedules in Section IV. In Section V, we show how the schedulability depends on the parameters and flow aggregation. Section VI describes heuristics for online admission control. Conclusions are reported in Section VII, which also highlights directions for future work.

II. SYSTEM MODEL

The framework developed in this paper relies on basic Network Calculus concepts, i.e. arrival curve, service curve and delay bound. Interested readers can find background in [15], from which we also borrow notation.

We assume that each mesh router is equipped with a single time-slotted channel. Transmission slots of a fixed duration $T_s$ are grouped into a frame of $N$ slots, which is periodically repeated every $N \cdot T_s$ time units. For instance, in 802.16 networks the frame length is usually set to 5 ms. Each slot is assigned to a set of non-interfering links through conflict-free link scheduling. At every slot, a subset of links may be activated for transmission only if no conflicts occur at the intended receivers. The WMN is modeled through a connectivity graph $G = (V, E)$, whose nodes $V = \{v_1, \ldots, v_n\}$ are mesh routers and whose edges $E = \{e_1, \ldots, e_m\}$ are directed links connecting nodes within transmission range. We assume that each link $e$ has a constant transmission rate $W_e$.

Nodes are traversed by flows (i.e., distinguishable streams of traffic). Each flow has a delay constraint, specified as an end-to-end delay bound or deadline $\delta_e$. At the ingress node, its arrivals are constrained by a leaky-bucket shaper, with a burst $\sigma_e$ and a rate $\rho_e$. We assume that each flow $q$ has a pre-established path $P_q \subseteq E$ between two nodes. This implies assuming that routing is given. Investigation of joint routing and link scheduling, which is the natural extension of this work, is left for the future. As far as buffering is concerned, we consider two options: a per-flow framework, where packets of each flow are buffered separately at each link. Thus, a link handles as many queues as the flows traversing it. Alternatively, in a per-path framework, packets of flows traversing the same path, i.e. the same set of links, are buffered in a single queue. This way, a link handles as many queues as the number of paths traversing it. The purpose of this paper is to describe a link scheduling algorithm that computes in both cases a conflict-free schedule which does not violate the required delay bounds whenever it is possible to do so. We first identify the constraints that ensure the conflict-free property, and then move to describing those related to delay feasibility.

The physical interference phenomenon is modeled by means of the widely used protocol interference models ([4], [17]-[18]). For each edge of the network $e \in E$ we define a conflicting set of edges $\mathcal{I}(e)$ which includes all the edges belonging to $E$ which interfere with $e$ (i.e. $\mathcal{I}(e)$ contains $e$ itself); the interference condition is straightforwardly defined as follows:

$$\sum_{i \in \mathcal{I}(e)} x_i(t) \leq 1, \text{ if } e \text{ is active in slot } t = 1, 2, \ldots, N,$$

where $x_i(t)$ is a binary variable, such that $x_i(t) = 1$ if link $e \in E$ is active in slot $t$, and 0 otherwise. This means that, if edge $e$ is active in slot $t$, the associated interfering set $\mathcal{I}(e)$ must contain one active edge only (which is the edge $e$ itself).

We translate the interference condition to a conflict graph $G_c = (E, C)$, shown in Fig. 1, whose nodes are the set of links of the connectivity graph and whose edges $C = \{e_1, \ldots, e_m\}$ model the conflicts within the network.
Half-duplex constraints are implicitly accounted for into the interference constraints, links being unidirectional. Hence a set \( I(e) \) can be easily obtained by retrieving the one-hop neighborhood of \( e \) in the conflict graph, e.g. for Fig. 1 we have \( I(7,8) = \{(4,7),(5,8),(8,7)\} \). Given a conflict graph \( C \), only conflicts between active links, i.e. those with a non-null flow, have to be considered. We thus define \( C_f \subseteq C \) as the subset of conflicts involving active links:

\[
C_f = \{(i,j) \in C : f_i > 0 \text{ and } f_j > 0\},
\]

where \( f \) denotes the flow going trough link \( i \).

Following the notation in [11], [16], we define an activation offset \( \pi_e \) for link \( e \), \( 0 \leq \pi_e \leq N \), and its transmission duration \( \Delta_e \). Since time is slotted, both are non-negative integers. Fig. 2 shows the above quantities, plus others that will be defined in the following.

The schedule must ensure the conflict-free condition: while a link is transmitting, all conflicting links must refrain from transmitting. For any pair of links \( i \) and \( j \) which are neighboring nodes in \( C_i \), we have:

- if \( j \) transmits after \( i \), it must wait for \( i \) to complete the transmission, i.e. \( \pi_i - \pi_j + \Delta_i \leq 0 \).
- Otherwise, the symmetric inequality holds, i.e. \( \pi_j - \pi_i + \Delta_j \leq 0 \).

In order to linearize the combination of the above constraints, we introduce a binary variable \( o_{ij} \), \( (i,j) \in C_f \), which is 1 if \( i \) transmits after \( j \), 0 otherwise. The left-hand side of the previous constraints can thus be upper bounded by \( N \) regardless of the relative transmission order, as \( \pi_i \) and \( \Delta_i \) belong to \([0,N]\). This completes the formulation of the conflict-free constraints, which are necessary and sufficient conditions:

\[
\pi_i - \pi_j + \Delta_i \leq N \cdot o_{ij} \quad \forall (i,j) \in C_f
\]

\[
\pi_j - \pi_i + \Delta_j \leq N \cdot (1-o_{ij}) \quad \forall (i,j) \in C_f
\]

For a schedule to be valid, each link must also complete its transmission within the frame duration, i.e.:

\[
\pi_i + \Delta_i \leq N \quad \forall i \in E
\]

Additional constraints are needed to keep into account the end-to-end delay requirements. We first formulate them for per-flow queuing, and then show how to modify them to allow for per-path queuing. During its activation, each link \( e \) transmits traffic of several flows. We can therefore partition the link’s \( \Delta_e \) among them, i.e. \( \Delta_e = \sum_{q \in E} \Delta_e^q \). \( \Delta_e^q \) is the link activation quota reserved for flow \( q \), which needs not be an integer, since when a link \( e \) is activated it can switch among backlogged queues regardless of slot boundaries. We assume that backlogged flows traversing \( e \) are served in the same (arbitrary) local order, and we call \( I_e \) the ordered set of the flow indexes. We assume that each backlogged flow \( q \) is served for \( no \ less \ than \ \Delta_e^q \). If a flow is idle, its service time can be exploited by other backlogged flows at \( e \), as long as the transmission from any flow \( z \) starts within at most \( \sum_{q \in E} \Delta_e^q \) from \( \pi_e \).

Therefore, flow \( q \) has a guaranteed rate equal to \( R_q = W \cdot \Delta_e^q / N \) at link \( e \). Since each flow has a single transmission opportunity in a frame, then a maximum inter-service time is guaranteed for that flow, and it is equal to \( \theta_q^e = (N - \Delta_e^q) \cdot T_s \), irrespective of the local ordering at each link. Therefore, each link of a mesh router is a rate-latency server [15] for the flows traversing it, with a rate \( R_q^e \) and a latency \( \theta_q^e \). Accordingly, each flow has an end-to-end delay bound equal to (see [15]):

\[
D_q = \frac{\sum_{c \in E} \theta_{qc}^e + \sigma_q}{R_q^e} \text{ if } \rho_q \leq R_q^e, \\
\infty \text{ otherwise}
\]

where \( R_q^e \) is set to the minimum of \( \{ R_q^e \} \).

The above bound is tight, i.e. \( D_q \) is actually the worst-case delay [15]. The first addendum in (3) is called latency delay, and it is due to link scheduling and arbitration of the flows at the links. The second is called burst delay, and it is the time it takes for the flow’s burst to be cleared at the minimum guarantee rate. If per-path queuing is used, instead, a set of flows \( q_1, \ldots, q_s \) traversing the same path can be modeled as a single flow. The latter has a required delay bound \( \delta_q = \min_{c \in E} \{ \delta_q^c \} \) and leaky bucket parameters \( \sigma = \sum_{c \in E} \sigma_q^c \), \( \rho = \sum_{c \in E} \rho_q^c \). Hence, the above modeling is still valid, including (3).

### III. Problem Formulation

Given the above traffic and network characterization, our aim is to find a conflict-free schedule which is also feasible from a delay point of view. To achieve this, we should solve the following end-to-end delay feasibility problem (E2DFP):

\[
\begin{align*}
\text{find} & \quad o_{ij}, \pi_i, \Delta_i, \Delta_e^q \\
\text{s.t.} & \quad D_q \leq \delta_q^e \quad \forall q \in Q \\
& \quad \Delta_e = \sum_{q \in E} \Delta_e^q \quad \forall e \in E \\
& \quad \pi_i - \pi_j + \Delta_i \leq N \cdot o_{ij} \quad \forall (i,j) \in C_f \\
& \quad \pi_j - \pi_i + \Delta_j \leq N \cdot (1-o_{ij}) \quad \forall (i,j) \in C_f \\
& \quad \pi_i + \Delta_i \leq N \quad \forall i \in E
\end{align*}
\]
Problem (4) is a feasibility problem, i.e. it is solved if a feasible solution is found. We transform it into a min-max optimization problem, whose objective function (to be minimized) is the maximum delay violation $V_{\text{max}} \triangleq \max_{q \in Q} \{ D_q - \delta_q \}$. We will call the latter the Minimum Max Violation Problem (MinMVP):

$$\begin{align*}
\min & \quad V_{\text{max}} \\
\text{s.t.:} & \quad D_q - \delta_q \leq V_{\text{max}} \quad \forall q \in Q \\
& \quad R_{\min} \leq W_q \cdot \Delta_q^f / N \quad \forall e \in P_q, \forall q \in Q \\
& \quad \Delta_q^f \geq N \cdot \rho_q / W_q \quad \forall e \in P_q, \forall q \in Q \\
& \quad \Delta_q^f \geq \sum_{q \in P_q} \Delta_q^f \quad \forall e \in E \\
& \quad \pi_e + \Delta_e \leq N \quad \forall e \in E \\
& \quad \pi_e - \pi_{f_e} + \Delta_e \leq N \cdot o_{f_e} \quad \forall (i, j) \in C_f \\
& \quad \pi_{j_e} - \pi_{f_e} + \Delta_e \leq N \cdot (1 - o_{f_e}) \quad \forall (i, j) \in C_f 
\end{align*}$$

Note that the fact that $R_{\min}^f = \min_{q \in P_q} \{ W_q \cdot \Delta_q^f / N \}$ is ensured by the second constraint; the weak inequality cannot imply that some delay bounds go to infinite (which would make the 1st constraints inherently wrong according to (3)). The finiteness of all delay bounds for all feasible points is indeed guaranteed by the third constraint. For the flow $q$ with the max violation, equality holds in at least one of the second constraints, i.e. $R_{\min}^f = \min_{q \in P_q} \{ W_q \cdot \Delta_q^f / N \}$. Instead, inequality may always hold in the second constraints for flows whose violation is not the maximum one. Clearly, the set of feasible solutions for (4) is the set of points where the objective function in (5) is non positive. The MinMVP formulation leads to a Mixed Integer Non-Linear (MINLP) problem, whose non-linear constraints are convex. The latter can be solved optimally with a general purpose MINLP solver (e.g. [19],[20]).

Re-writing a feasibility problem, i.e. the E2EFP, as an optimization problem, i.e. the MinMVP, bears considerable advantages. First of all, we will show that it allows us to explore the schedulability region, assessing the relationships between the schedulability and the various parameters involved (i.e., flow deadlines, burst, rates). To this end, $V_{\text{max}}$ is a good indicator of how much the WMN is loaded, i.e. whether it might support more traffic or tighter deadlines (or, if $V_{\text{max}}$ is positive, which flow is the most critical). Second, it allows us to quantify how robust a link schedule is with respect to variations (or uncertainties) in the flow parameters (i.e., burst and rates). We remark that, in order to formulate the MinMVP problem, we need: i) the conflict graph of the WMN, ii) the path of each flow, and iii) the traffic parameters (i.e., $\sigma, \rho, \delta$) of each flow/aggregate traversing a path. Therefore, this method does not rely on the WMN having a specific topology.

IV. DISCUSSION

We first provide some insight on how the solver distributes the slots in the MinMVP problem while striving to minimize $V_{\text{max}}$: we do this in a case-study binary tree of 15 nodes, shown in Fig. 3, where the root acts as gateway and routing is not an issue. One flow is originated at each node and terminated at the gateway. Links are numbered after their source node for conciseness. The frame length is $N=100$ slots, with $T=0.05$, whereas the link capacity is $W_e=9600$. We set $\delta=20$, $\sigma=100$, $\rho=300$. For all flows, in which case flows starting at leaf nodes achieve the maximum violation. We vary the parameters of flow $f_o$, first $\sigma_o$ in $[100;4000]$, then $\rho_o$ in $[300;3500]$, and we solve the MinMVP optimally for each instance, studying how the optimal solution changes with $\sigma_o$ and $\rho_o$.

Fig. 4 shows that $V_{\text{max}}$ (always equal to $V_{\rho}$) increases with $\sigma_o$. However, neighboring flows, i.e. those traversing links conflicting with those of path $P_o$, amortize the increasing scheduling burden imposed by $f_o$’s burst. Their (negative) delay violation increases, so that $V_{\rho}$ only increases by 1.5% in the whole range of bursts. This is confirmed by examining how the $\Delta^f_{\rho}$ change in the optimal solutions, for both $f_o$ and $f_i$ (Fig. 5). For $f_o$, the initial decrease on link 0 is compensated by an increase on the other two links 8 and 12. The overall amount of slots on the path $CB_p=\sum_{f^e} \Delta^f_{\rho}$ increases with $\sigma_o$.

We repeat the same analysis varying $\rho_o$. Fig. 6 shows both $V_{\text{max}}$ and $V_{\rho}$ as a function of $\rho_o$. On the one hand, $V_{\text{max}}$ increases by only 30% when $\rho_o$ increases by seven times. On the other hand, $V_{\rho}$ decreases because increasing its rate forces the solver to increase the lower bound of $\Delta^f_{\rho}$ on the links in $P_o$. As in the former case, neighboring flows see their resource share reduced, which increases their delay. Fig. 7, showing the slots assigned to $f_o$ and $f_i$, confirms the analysis, and shows $CB_i$ decreasing with $\rho_o$.

A. Robustness analysis

We now show how robust a given MinMVP schedule is to variations in the flow parameters (i.e. $\sigma, \rho$). We want to identify the region where those parameters can vary without missing the deadlines. This would allow a network administrator to compute a link schedule based on expected, possibly imprecise, traffic requirements, and stick to that with reasonable confidence. Unlike in the former case, where a new optimal schedule was computed for each traffic scenario, here we proceed as follows: we fix the traffic parameters and solve the MinMVP optimally once and for all, obtaining a given set of $\pi_{e}, o_{e}, \Delta_{e}$. Then, we assess how robust $\pi_{e}$, $o_{e}$ and $\Delta_{e}$ is. We repeat the solution varying $\rho_o$.
max \[ \sum_{q \in Q} \sigma_q \] 
s.t.: \[ \text{s.t.:} \quad \sum_{q \in Q} \sigma_q + \varepsilon_q^{\mu} - \delta_q \leq 0 \quad \forall q \in Q \]
\[ \Delta'_V \geq N \cdot \rho_i/W \quad \forall e \in P_i, \forall q \in Q \] 
\[ R^q_{\min} \leq W \cdot \Delta'_V/N \quad \forall e \in P_i, \forall q \in Q \] 
\[ \Delta'_V \geq \sum_{q \in \epsilon} \Delta'_V \quad \forall e \in E \] 
\[ \varepsilon_q^{\mu} \geq 0 \quad \forall q \in Q \]

In other words, it entails maximizing the sum of the excess bursts \( \varepsilon_q^{\mu} \) while keeping the delays within the requirements (first and second constraints). The fourth constraint guarantees that the link schedule is untouched. A non-weighted sum of \( \varepsilon_q^{\mu} \) may not be the most interesting objective for a given scenario (e.g., some flows may be more critical or important than others). However, this does not hamper the generality of the exposition, as weights can be easily added back if required. The MPP formulation for the rate can be easily inferred from (6) by the alert reader and is thus omitted.

The MPP is a continuous convex problem, hence solvable in polynomial time ([21]) and however always orders of magnitude quicker than the MinMVP. In Table I we report the values of \( \varepsilon_q^{\mu} \) (for some flows) obtained by solving the MPP in a homogeneous network, for different values of the flow bursts (left, with flow rates equal to 300) and rates (right, with bursts equal to 100). \( \delta = 20 \) in both cases, and \( V_{\max} \) is also reported.

Variations of the bursts within the box defined by \( \varepsilon_q^{\mu} \) can be accommodated without changing the link schedule. We first observe that the network is moderately to highly loaded, with a homogeneous network, for different values of the flow bursts (left, with flow rates equal to 300) and rates (right, with bursts equal to 100). \( \delta = 20 \) in both cases, and \( V_{\max} \) is also reported.

As a function of the bursts, hence allowing for larger bursts to be accommodated when varying the burst, the slack on the \( \varepsilon_q^{\mu} \) progressively decreases from the leaves to the root, so that only the shortest paths have non null values. However, when varying the rates, the slack increases for longer flows, and decreases for the shortest one. This is because varying \( \rho \) also increases the lower bound on \( \Delta'_V \), and accordingly \( R^q_{\min} \), hence allowing for larger bursts to be accommodated. However, this imposes tighter bounds on \( \Delta'_V \), as \( \Delta'_V \) has to remain constant due to the fourth constraint in (6).

Note that, with \( \rho = 650 \), the WMN is almost saturated (the highest tolerable rate being \( \rho = 685 \) in a homogeneous scenario).

![Figure 4. \( V_{\max} \) against \( \sigma_0 \).](image)

![Figure 5. Slots assigned to \( f_0 \) and \( f_1 \) against \( \sigma_0 \).](image)

![Figure 6. \( V_{\max} \) and \( V_0 \) against \( \rho_0 \).](image)

![Figure 7. Slots assigned to \( f_0 \) and \( f_1 \) against \( \rho_0 \).](image)

### Table I

\( \varepsilon_q^{\mu} \) as a function of the bursts (left) and rates (right), homogeneous network.

<table>
<thead>
<tr>
<th>( \sigma )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>( V_{\max} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow id</td>
<td>2194</td>
<td>2344</td>
<td>2199</td>
<td>2164</td>
<td>3103</td>
<td>3103</td>
<td>-7.93</td>
</tr>
<tr>
<td>500</td>
<td>1844</td>
<td>1703</td>
<td>1664</td>
<td>1679</td>
<td>2594</td>
<td>2594</td>
<td>-7.38</td>
</tr>
<tr>
<td>1000</td>
<td>1344</td>
<td>1017</td>
<td>1168</td>
<td>969</td>
<td>2279</td>
<td>2094</td>
<td>-8.64</td>
</tr>
<tr>
<td>1500</td>
<td>664</td>
<td>883</td>
<td>703</td>
<td>844</td>
<td>1594</td>
<td>1594</td>
<td>-6.26</td>
</tr>
<tr>
<td>2000</td>
<td>203</td>
<td>344</td>
<td>368</td>
<td>344</td>
<td>1094</td>
<td>1094</td>
<td>-5.65</td>
</tr>
<tr>
<td>2500</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>594</td>
<td>594</td>
<td>-5.02</td>
</tr>
<tr>
<td>3000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>94</td>
<td>94</td>
<td>-4.39</td>
</tr>
<tr>
<td>3500</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>33623</td>
<td>-3.75</td>
</tr>
<tr>
<td>4000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27113</td>
<td>-3.07</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \rho )</th>
<th>50</th>
<th>150</th>
<th>250</th>
<th>350</th>
<th>450</th>
<th>550</th>
<th>650</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow id</td>
<td>572</td>
<td>673</td>
<td>630</td>
<td>673</td>
<td>870</td>
<td>72553</td>
<td>-7.96</td>
</tr>
<tr>
<td>100</td>
<td>1125</td>
<td>861</td>
<td>951</td>
<td>1048</td>
<td>1500</td>
<td>66322</td>
<td>-7.92</td>
</tr>
<tr>
<td>200</td>
<td>2055</td>
<td>1528</td>
<td>1690</td>
<td>1917</td>
<td>2465</td>
<td>55072</td>
<td>-7.85</td>
</tr>
<tr>
<td>300</td>
<td>2671</td>
<td>2443</td>
<td>2780</td>
<td>2780</td>
<td>3528</td>
<td>44197</td>
<td>-7.77</td>
</tr>
<tr>
<td>400</td>
<td>3328</td>
<td>3328</td>
<td>3687</td>
<td>3317</td>
<td>4611</td>
<td>33697</td>
<td>-7.68</td>
</tr>
<tr>
<td>500</td>
<td>3857</td>
<td>4148</td>
<td>4075</td>
<td>4076</td>
<td>5726</td>
<td>23572</td>
<td>-7.53</td>
</tr>
<tr>
<td>600</td>
<td>4695</td>
<td>4468</td>
<td>4663</td>
<td>4695</td>
<td>6927</td>
<td>13822</td>
<td>-7.18</td>
</tr>
</tbody>
</table>
As a second step, we address whether and when opting for a per-path framework increases the schedulability of a set of flows, i.e., allows larger bursts and/or smaller deadlines to be accommodated. In order to do so, we vary both the burst and the deadlines of the set of flows. Our results show that, if the WMN is traversed by homogeneous flows, whatever their burst, rate and required delay bound, a per-path framework always yields smaller delay violations. Fig. 9 reports $V_{\text{max}}$ against the number of flows per node; the deadline is set to 20, and the overall $\sigma, \rho$ are constant and equal to 5000 and 300 respectively. The above values are divided among an increasing number of homogeneous flows (1 to 50), with 1 representing in fact a per-path framework. The figure shows that the gain with a per-path framework increases with the number of flows. Thus, aggregating a large number of smaller flows (besides leading to more manageable implementations, due to the reduced number of queues to be managed) improves the overall performance. This is also true if we relax the assumption of homogeneous flows, at least up to some extent. Fig. 10 reports the average $V_{\text{max}}$ for 100 instances with heterogeneous random flows (confidence intervals are not reported since they are not visible): rates and bursts are generated uniformly between $[0.8;1.2] \cdot 300/K$ and $[0.8;1.2] \cdot \sigma/K$ respectively, with $\sigma$ ranging from 1000 to 10000 and $K = 30$ being the number of flows at each link, all of which have $\delta = 20$. Although the two lines represent averages, $V_{\text{max}}^{\text{per-flow}} \leq V_{\text{max}}^{\text{per-path}}$ holds for each instance of the problem. For instance, Fig. 10 suggests that the maximum aggregate burst schedulable in a per-flow framework is 4500, whereas in a per-path framework it is 6200, i.e. 38% larger. This corresponds to approximately 11 additional flows per node. More to the point, the behavior does not change if we allow for a larger spread for the bursts. We have increased the above interval from $[0.8;1.2]$ to $[0.2;1.8]$, without experiencing any noticeable change in the outcome. Even considering non-uniform distributions for the flows, e.g. $[0.2;0.4]$ for one half of the flows and $[1.6;1.8]$ for the other, has no significant impact on $V_{\text{max}}$. This seems to suggest that, as long as flows have the same deadline, aggregating heteroge-
neous flows improves the delay performance, and that the latter depends on the overall burst rather than on how it is distributed.

The behavior changes if we aggregate flows with different deadlines. In that case, using a per-path framework may or may not be beneficial, depending on the flow parameters. Fig. 11 shows a case with 3 flows per node having the same $\sigma, \rho$, with $\rho = 300$ and $\sigma$ ranging from 0 to 2000, but different required delay (30, 60, and 100 respectively). In this case, for bursts smaller than 500, per-path aggregation performs better (i.e., the network can accommodate a smaller minimum required delay), whereas the opposite is true for larger bursts. This can be explained by considering that, depending on the burst size, either the latency or the burst delay may be predominant in (3). On one hand, aggregating flows always reduces their latency delay, since a larger transmission duration is given to the aggregate. On the other hand, tighter delay requirements are imposed on the aggregate, which instead increases the maximum violation. The first effect dominates for small bursts.

VI. ONLINE ADMISSION CONTROL

The presence of integer ($\pi_{ij}, \Delta_{ij}$) and binary ($o_{ij}$) variables makes the MinMVP complex. The problem can be solved optimally for WMNs of few tens of nodes, which is the expected scale for current and future WMNs. Fig. 12 shows the distribution of the computation times for solving 100 instances of three different WMNs of 15 nodes, i.e. a balanced binary tree, a star topology, and a random tree. Computations are done on a PC equipped with an Intel Core2 Duo E6400 2.1 GHz, 2 GB RAM and a Linux kernel 2.6.18, using CPLEX as a solver [19]. Solving larger instances (e.g. 20-30 nodes) may take tens of minutes or hours, depending on both the topology and the load. These are certainly affordable times when compared to the timescales of network (re)engineering, but not so when compared to the timescale of admission control decisions. Moreover, for even larger networks, an optimal solution might be impossible to compute due to memory and time constraints, i.e. we must trade optimality for computation time and resources. We now consider the problem of reducing the computation time, so as to incorporate link scheduling within an online admission control procedure. We first introduce a heuristic to solve the MinMVP, and then we discuss when it is not necessary to solve it anew to admit new flows.

A. Heuristic solution approach

We observed that the solver efficiency improves considerably when priority is given to the $o_{ij}$ variables. Furthermore, once the latter are set, relaxing the integrality for $\pi_{ij}, \Delta_{ij}$ yields a continuous and convex problem, which can therefore be solved in polynomial time. Capitalizing on this, we can devise a heuristic based on the following steps: i) we assign values to each $o_{ij}$ variable, producing several binary strings representing conflict orientations; ii) we solve each reduced problem, i.e. MinMVP problem given a conflict orientation string, using a heuristic approach; iii) we use the above solutions to assess the fitness of each conflict orientation string in an evolutionary perspective, and we step back to point i). As we show later on, the quality of the solutions improves with the number of iterations.

Therefore, we can use this heuristic scheme for two purposes: on one hand, to solve (suboptimally) infeasibly large MinMVP problems, given enough iterations (i.e., computation time). On the other hand, we can limit the iterations, and have it complete (still suboptimally) in a predictable and reasonably short time (e.g., seconds), so as to use it for online admission control. Hereafter, we first explain how to solve the reduced problem; we then describe the genetic approach, and finally we present some performance results for this scheme.

Link transmission orders are represented by binary variables $o_{ij}$. Setting the $o_{ij}$ variables implies orienting the edges, which is the expected scale for current and future WMNs. The variables determine the activation of the conflict-free constraints (1). For a fixed conflict orientation string, i.e. a given set of values for the $o_{ij}$ variables, each pair of constraints can be replaced by either of the following:

$$\begin{align*}
\pi_{ij} + \Delta_{ij} &\leq \pi_{ij} \quad \text{if} \quad o_{ij} = 0 \\
\pi_{ij} + \Delta_{ij} &\leq \pi_{ij} \quad \text{otherwise}
\end{align*}$$

(7)

Assuming that the conflict orientation string is given, our heuristic solution approach consists in solving two continuous convex non linear subproblems and using a rounding procedure, as shown in Fig. 13.

The first step of the proposed approach consists in formulating a nonlinear convex program starting from the MinMVP, relaxing the integrality constraints on variables $\pi_{ij}$ and $\Delta_{ij}$, and substituting (7) for (1), i.e.:

$$\begin{align*}
\min \quad & V_{\max} \\
\text{s.t.:} \quad & D_q - \delta_q \leq V_{\max} \quad \forall q \in Q \\
& R_q^{\min} \leq W_e \cdot \Delta_{e}^{\min} / N \quad \forall e \in P_q, \forall q \in Q \\
& \Delta_{e}^{\min} \leq N \cdot \rho_e / W_e \quad \forall e \in P_q, \forall q \in Q \\
& \Delta_{e}^{\min} \geq 1 + \sum_{q \in \pi_e} \Delta_{q}^{\min} \quad \forall e \in E \\
& \pi_{e} + \Delta_{e} \leq \hat{S} \quad \forall e \in E \\
& \pi_{e}, \Delta_{e} \text{ are rounded} \quad \forall e \in E
\end{align*}$$

where $\hat{S}$ concisely denotes constraints (7). Then, the solutions of this reduced model ($\pi_{ij}, \Delta_{ij}$) are rounded to their integer part. Note that a “+1” is required in the 4th constraint to ensure that $\Delta_{e}^{\min} \geq \sum_{q \in \pi_e} \Delta_{q}^{\min}$, i.e. to prevent the rounding from reducing the minimum guaranteed rate below the required one. The rounded solution is still feasible from a conflict-free point of view, since:
We use a population deliberately chosen to make no assumptions on it (such as those in [11]), as this does not hamper the practicability of this framework.

We now show that the heuristic solutions are close to the optimal ones. Their quality improves with the number of iterations. However, for medium-sized topologies, good solutions are available within relatively few iterations, so that in most cases a decision can be made in few seconds. Fig. 15 and Fig. 16 are related to a sink-tree WMN with 27 nodes (26 links) carrying uplink traffic to the gateway. We ran the solution algorithm on several instances, obtained by varying the parameters uniformly with \( \sigma \in [0,10000], \rho \in [50,400], \delta \in [40,100] \), in a per-path framework. Fig. 15 shows both the number of instances for which a feasible solution (i.e., one with a negative \( V_{\text{max}} \)) is found and the time to compute it against the number of iterations. As the figure shows, few iterations are enough for all the instances. The solving time is linear with the number of iterations, and within few seconds on the same system mentioned in the previous subsection. The genetic algorithm is implemented using GaLib [23]. A reasonable speedup is likely to be harvested by simply using more performing (though still off-the-shelf) hardware and optimizing the solver. This leads us to think that this approach is viable for online admission control in WMNs of comparable size. As far as optimality is concerned, we show in Fig. 16 that 75% of the instances are below 10% from the corresponding optimal value, and only 4% are above 30%, which we find acceptable.

### B. Reducing the occurrence of link schedule computations

In Section IV.A we introduced a simple and effective technique to analyze the robustness of a MinMVP schedule to variations of the flow parameters. A similar approach can be used as a quick admission control test. If the current link schedule can tolerate the admission of new flows without violating the deadline requirements, then no re-optimization is required, which saves computations. Given a schedule, and considering new flow requirements \( \sigma_q^*, \rho_q^*, \delta_q^*, q \in Q \), a quick
admission control test is obtained by solving the following problem:

\[
\min \quad V_{\text{max}} \\
\text{s.t.} \quad \sum_{q \in \mathcal{Q}} \theta_q + \frac{\sigma_{\text{max}}}{\delta_{\text{max}}} - \delta_q \leq V_{\text{max}} \quad \forall q \in \mathcal{Q} \\
\Delta^q \geq N \cdot \rho_j / W_q \quad \forall e \in P, \forall q \in \mathcal{Q} \quad (10) \\
R_{\text{max}}^q \leq W \cdot \Delta^q / N \quad \forall e \in P, \forall q \in \mathcal{Q} \\
\Delta_i \geq \sum_{q=1}^{\mathcal{Q}} \Delta^q \quad \forall e \in E
\]

and checking whether \( V_{\text{max}} \leq 0 \). The alert reader will notice that (10) is in the same form as (6), i.e. continuous and involving only variables \( V_{\text{max}} \), \( \Delta^q \), and \( R_{\text{max}}^q \), thus solvable in polynomial time. The main difference with (6) is that (10) is again a min-max problem, hence \( V_{\text{max}} \) is kept as small as possible. If the test succeeds, the new flows can be admitted, and all that is required is to communicate the new \( \Delta^q \) values to the mesh routers. Otherwise, a new MinMVP has to be solved, either optimally or using the heuristic approach, with the new flow parameters as an input.

VII. CONCLUSIONS AND FUTURE WORK

This paper has addressed the problem of link scheduling for real-time traffic in Wireless Mesh Networks. Real-time traffic requires end-to-end delay bounds, and the known link scheduling algorithms do not take the latter into account. We showed that, given a conflict graphs and flow routes, the problem is integer-non linear and can be solved optimally for WMNs of few tens of nodes. Thanks to the formulation as a min-max optimization problem, the solutions are robust to even large variation in the flow parameters. Furthermore, we explored the parameter space, highlighting the dependencies between the solutions and the network and flow parameters. More specifically, we show that using a per-path framework normally improves the performance, at least if deadlines are homogeneous. Finally, we proposed a heuristic solution approach, that computes good suboptimal schedules within smaller and controllable times. The latter completes in seconds or less in the above WMNs, making it viable to use it as an admission control scheme for real-time traffic.

This work can be extended in several directions, which are actively being pursued at the time of writing. The first one is to bring routing back into the framework, i.e. to address the problem of finding a joint optimal routing and link scheduling, subject to end-to-end delay guarantees. Furthermore, exploring more sophisticated and possibly faster heuristics, which capitalize on the structure of the problem, is also being considered.

REFERENCES