Optimal multipath forwarding in planned wireless mesh networks

Germán Capehourat[a,*], Federico Larroca[a], Pablo Belzarena[a]

[a] *Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República*
Julio Herrera y Reissig 565, ZC 11300, Montevideo, Uruguay

Abstract

Wireless Mesh Networks (WMNs) have emerged in the last years as a cost-efficient alternative to traditional wired access networks. In the context of WMNs resources are intrinsically scarce, which has led to the proposal of dynamic routing in order to fully exploit the network capacity. We argue instead in favour of separating routing from forwarding (i.e. *à la* MPLS). Our proposal is a dynamic load-balancing scheme that forwards incoming packets along several pre-established paths in order to minimize a certain congestion function. We consider a particular but very typical scenario: a planned WMN where all links do not interfere with each other. We use a simple and versatile congestion function: the sum of the average queue length over all network nodes interfaces. We present a method to learn this function from measurements and several simulations to illustrate the framework, comparing our proposal with the IEEE 802.11s standard.

Keywords:
wireless mesh networks, traffic engineering, load-balancing

1. Introduction

Wireless Mesh Networks (WMNs) [1] are no longer just a promise for the future but a reality today, thanks mainly to the advantage offered in terms of cost compared to traditional wired access networks. In particular, outdoor community mesh networks [2] and rural deployments [3, 4] based on IEEE 802.11 have seen tremendous growth in the recent past. An example is Plan Ceibal [5] which provides connectivity to every school in Uruguay, where WMNs are used to reach suburban and rural schools. Lately even service providers are beginning to use this technology, resulting in an increasing presence of carrier-class equipment in the market [6].

Under this scenario, the typical architecture (see Fig. 1) includes one or more internet gateways and several relay routers. Clearly, this intermediate routers increase the coverage of the access network without requiring more, and probably expensive, connections to the internet. However, several problems arise that are specific of this kind of architectures.

The main challenge for this kind of networks, at the wireless mesh backbone level, is routing and forwarding. In the current IEEE 802.11s standard [7] (and in several other proposals [8]) each link has an associated metric value as cost. This cost is expected to change over time, and reflect current conditions (propagation conditions, interference, etc.), so as to maximize a certain criteria (e.g. throughput). To choose a path to its destination, each router executes a shortest path algorithm. This procedure is essentially the same than the one used in wired networks. The main difference is that, just like in the internet until the early eighties, link costs are allowed to change at a time scale of some seconds [9]. The more static configuration that is used nowadays is due to the oscillations caused by these dynamic costs. It seems like history is repeating itself, since early experiments with WMNs have also reported routing oscillations [10, 11].

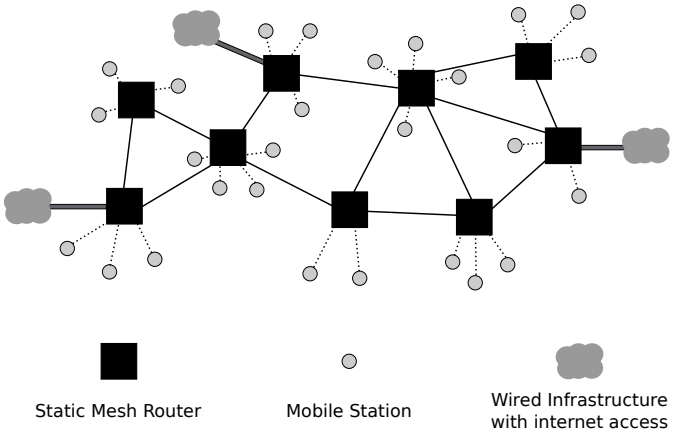


Figure 1: Wireless Mesh Network (WMN) typical architecture.

However, a completely static routing approach is not a suitable solution in this context. Static means non-optimized routing. In the wired case this is not such a big issue, since resources, specially in the core, are relatively inexpensive (in fact, most core networks are overprovisioned). On the contrary, in

*Corresponding author
Email addresses: gcapde@fing.edu.uy (Germán Capehourat), flarroca@fing.edu.uy (Federico Larroca), belza@fing.edu.uy (Pablo Belzarena)

wireless networks resources are intrinsically scarce, and "upgrading" a link's capacity is not always a possibility. Available resources must then be used at its maximum, and for this purpose a certain form of dynamism must be implemented in the network.

We present a novel approach which separates routing from forwarding, just like MPLS does in the wired context. That is to say, each ingress router has several possible paths towards the destinations, and these paths remain unchanged as long as no topological change takes place (e.g. a node failure). Please note that in the context of WMNs we may safely assume that nodes are fixed and do not change status nor position very often. Each new incoming flow will be forwarded along one of these paths, a decision that each ingress router will take depending on the current network condition We shall call this procedure dynamic load-balancing. We propose one such scheme that forwards incoming packets along several preestablished paths in order to minimize a certain objective function. If correctly designed, load-balancing will bring improved performance over static routing, without the difficult to avoid oscillations of pure dynamic routing. For more arguments in favour of load-balancing see the discussion presented in [12], where Caesar *et al.* argue for a separation of timescale between *offline* computation of multiple paths and *online* spreading of load over these paths, or the analysis by Pham *et al.* [13] where single-path and multi-path routing protocols are compared in a wireless networks scenario, showing that the latter provides better performance.

We consider a particular but very typical scenario: a planned WMN, where all bidirectional point-to-point links do not interfere with each other. This assumption means either that all backhaul links use different channels or that links in the same channel are in different collision domains. There are many scenarios where this assumption holds, for example suburban or rural area networks and even campus networks, deployed with high directional antennas with proper RF design and channel assignment. This assumption also implies that the network topology is already defined, typically at infrastructure deployment phase. This means we cannot decide which backhaul links to establish but only how to use them, i.e. which traffic route through them.

The question that remains is to what purpose should load-balancing serve and be worthwhile. That is to say, what function of the traffic distribution should be optimized (where "traffic distribution" refers to the portion of traffic sent along each path). In this paper we argue that this function should be the sum over all nodes' interfaces of the corresponding average queue length. As shall be discussed in Sec. 3, this is a very versatile and important performance indicator. The problem we address is then to find the traffic distribution that minimizes the sum over all interfaces of the average queue size. However, instead of relying in analytical expressions based on (arbitrary) models, we will strive at reflecting reality as much as possible, and design a measurement-based scheme. In this framework the relationship between the average queue length and the current traffic distribution will be learned from measurements, and the optimization shall be performed based on this learned function.

This kind of approach, using a network model developed from measurements of queue sizes and traffic loads, has already proved suitable for a wired scenario [14]. In this work, we extend the framework to the previously described wireless scenario. Furthermore, we also consider the dynamic gateway selection problem and we obtain a load balanced solution using the proposed approach. Differently to the wired case, in the considered wireless scenario the average queue size at a given interface now depends not only on the incoming traffic, but also on the activity of the interface at the other end of the link. We model each link with only one average queue (the sum of both interfaces involved) which depends on the traffic in both link directions. A method to learn this bi-variable function is presented, whereas simulations illustrate the framework.

It is important to highlight that we are considering a WMN where links performance is stable and predictable, with a strong correlation between the error rate and the received signal strength. In the context of WMN, as stated in [15], interference (and not multipath fading) is the primary cause of unpredictable performance. In the scenario of interest there is no internal interference, so we expect to have a proper model with the proposed learning technique.

In a nutshell, the contributions of this paper are the following. We propose a load-balancing framework for multipath forwarding in 802.11 WMNs and we show the advantages for this kind of networks. We compare the performance of the proposed method with static routing through shortest path and dynamic routing using 802.11s. Several simulations over canonical topologies show the advantages of the proposed scheme over the alternatives. The proposed framework also copes with the gateway selection problem, typically present in WMNs. The deployment of WMNs in recent years has grown and is expected to continue rising, so it becomes essential to find a proper routing/forwarding to provide adequate service to the also growing traffic demands. The results we present suggest that dynamic load-balancing is an excellent candidate.

The rest of the paper is structured as follows. In the next section we describe some previous work and highlight some recent papers. In Sec. 3 we introduce the network model and most of the notation used in the paper. The paper continues in Sec. 4 where we describe the procedure for learning the congestion function model from measurements, while in Sec. 5 we detail the operation of the proposed method. Finally, in Sec. 6 we present the simulation experiments and performance comparison, while conclusions and future work are discussed in Sec. 7.

2. Related work

In the context of WMNs, several previous works presented new metrics for single path routing that take into account information from lower layers [8]. The need to increase the WMNs capacity led to the use of nodes with multiple radio interfaces which was analyzed in [16, 17]. In this paper we consider a planned WMN, where all links do not interfere with each other. Even in an unplanned scenario several algorithms have been

proposed [18, 19, 20] which could be used to schedule the links so that they do not interfere with each other.

There are some recent related works that we would like to highlight. In [21] an optimization framework is presented to reach minimum average delay per packet in a single channel WMN. Starting from a Markov chain model for the medium access of a single node, they derived a closed form representation for the average system delay which is used as the objective function. The model takes into account the neighbours interference but several parameters of the Markov chain need to be calculated or defined which could difficult the implementation.

Another work that uses an analytical model in the context of single channel WMNs is [22]. In particular, the authors developed a queuing-based model which is used to estimate the network capacity and to identify network bottlenecks. Based on a load-aware routing metric they choose the corresponding path for each new incoming flow, and then based on the model a centralized entity performs admission control to guarantee network stability. They focused on per-flow performance and compare the results with shortest-path first routing algorithm.

Concerning dynamic gateway selection, in [23] an heuristic algorithm was proposed to tackle the problem. A single channel WMN is considered between routers, but operating in a different channel than links between WMN nodes and mobile hosts. They assume that a routing protocol is executed in the WMN which establishes routes between every pair of nodes, including the gateways. They seek to minimize the maximum number of flows served by a gateway and minimize the cost of paths in order to avoid interference in the network. Contention regions are modelled as the maximal cliques of the contention graph, which leads to a Mixed Integer Nonlinear Programming (MINLP) formulation of the problem. Their proposal solves gateway selection for internet flows in a centralized manner using a greedy heuristic.

To the best of our knowledge, the only work that proposes a forwarding scheme for WMNs is the recent article [24], where the authors present an MPLS-based forwarding paradigm. However, two important differences with our proposal should be highlighted. Firstly, they allow traffic splitting at every node in the network while we only allow it at ingress routers. Secondly, and most importantly, they considered the *hose* traffic model (only knowledge about maximum traffic demands) which leads to a robust routing fashion to solve the problem. The optimization cost function of a routing solution is calculated as the average over all the feasible flows allocations, where the function used is a weighted average of the total utilizations over all the collision domains. We think that in the context of WMNs, it is more appropriate to consider a dynamic load-balancing solution rather than a robust routing scheme, because it is exactly in scenarios with highly dynamic traffic like WMNs where the former takes advantage over the latter. For a deep comparison between both methods please refer to [25].

All in all, two major differences should be distinguished between our proposal and previous works. The first one is the introduction of a measurement-based model for 802.11 links, whereas most of the literature is based on (arbitrary) MAC layer models like the one presented in Bianchi's seminal paper [26]. The second important difference is the time scale at which decisions are taken. Most of routing algorithms proposed for WMNs are based on a certain metric which changes at a time scale of seconds. Our framework operates with averages taken over tens of seconds and forwarding decision is taken with flow granularity. This fact enables decoupling the link model learning phase from the forwarding decision, and ensures better stability properties avoiding route flapping problem.

3. Network Model and Problem Formulation

Firstly, let us remark that in the context of WMNs we may safely assume that nodes are fixed and do not change position very often. In addition, power supply is not a problem, so we will completely ignore energy consumption. We will then concentrate on the performance as perceived by packets in terms of delay, dropping probability and throughput. Naturally, we will limit ourselves to the WMN, which means that throughput will refer to a quantity proportional to the inverse of the time that it takes any given packet to leave the network.

Before introducing the notation, let us highlight that throughout this paper we will assume that each node has a single FIFO queue attached to each of its (possibly several) interfaces. This means that all packets at each interface will receive the same treatment, independently of its destination, number of traversed hops, etc. This is not a very problematic assumption, since the only queue management that most wireless routers implement is some form of prioritization of certain particular and few packets (e.g. ARP packets).

Let $n = 1, ..., N$ be the set of static wireless mesh routers (including gateways) which we shall call *nodes* and $l = 1, ..., L$ the backbone bidirectional links in the network. Typically, high gain directional antennas are used for backhaul links with other nodes and sector panels or omnidirectional antennas are used to provide connectivity for mobile stations. Gateways nodes have also wired links to a fixed infrastructure network with internet access. We will focus on the mesh core, so only backhaul links and aggregated traffic at mesh routers will be considered. Traffic generated at node n will refer to all traffic arriving at n from the mobile hosts attached to it. We will assume that this traffic uses different channels (e.g. 802.11b/g) than the ones used within the mesh core (e.g. 802.11a). If n is a gateway, the generated traffic also includes that coming from the internet to nodes in the WMN. As we mentioned before, we shall further assume that channels within the mesh core do not interfere with each other. Moreover, paths are assumed to be established *a priori* and how to choose them is out of the scope of the present paper. In particular, we will use the k shortest paths.

Traffic generated at a node will have as final destination a set of nodes, which may contain for instance any other node in the WMN. This defines a set of possible origin-destination (OD) pairs, which we shall index by the integer $s = 1, ..., S$. The amount of traffic corresponding to OD pair s will be noted by d_s and we further define the column vector $\boldsymbol{d} = [d_1 \ ... \ d_S]^T$. We will assume that entries in $\boldsymbol{d}$ are independent of each other. In

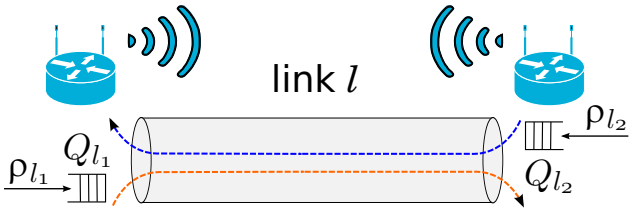


Figure 2: Wireless link queues and flows in both directions.

particular, this means that the amount of traffic sent to the internet through a particular gateway does not influence the amount of traffic that gateway generates.

Each pair will have a set of n_s fixed, established *a priori* paths, which we shall note as P_{si} for $i = 1, ..., n_s$. The amount of traffic sent along path P_{si} shall be noted as $d_{P_{si}} = \alpha_{P_{si}} d_s$, where $\alpha_{P_{si}}$ is the traffic distribution coefficient for path P_{si}. We further define $\boldsymbol{\alpha} = [\alpha_{P_{11}} \ ... \ \alpha_{P_{1n_1}} \alpha_{P_{21}} \ ... \ \alpha_{P_{S1}} \ ... \ \alpha_{P_{Sn_S}}]^T$ as the traffic distribution vector. The following two constraints should hold $\sum_{i=1}^{n_s} d_{P_{si}} = d_s \ \ \forall \ s$ and $d_{P_{si}} \geq 0 \ \ \forall \ s, i$, which implies $\sum_{i=1}^{n_s} \alpha_{P_{si}} = 1 \ \ \forall \ s$ and $\alpha_{P_{si}} \geq 0 \ \ \forall \ s, i$.

Within this context, for each link l we have two traffic loads, one for each direction of the communication, which we shall call ρ_{l_1} and ρ_{l_2} taking any arbitrary convention (see Fig. 2). Given a demand vector $\boldsymbol{d}$ and a traffic distribution vector $\boldsymbol{\alpha}$, the total traffic load on link l in one direction (e.g. ρ_{l_1}) is given by the sum over all OD pairs of the traffic forwarded along those paths P_{si} which use the link in that direction. Let D_{l_1} be the average amount of time a packet spends at the queue of link l in the direction of load ρ_{l_1}. Naturally, this non-decreasing function depends on the traffic load ρ_{l_1}, which is the queue's input traffic intensity. However, and due to the half-duplex operation of the link and the 802.11 medium access control, D_{l_1} also depends on the load in the opposite direction (ρ_{l_2}).

Let us now discuss with more detail what this delay is composed of. Once a packet enters a node interface queue, it has to wait for several things to happen. Firstly, it has to reach the head of the line of the queue. What happens after then depends on whether the node is a gateway and the packet goes to the internet, or not. In the former case, it has to wait for all its bits to be sent by the wired interface. In the latter case, it has to wait for the channel to be idle. Once this happens, the packet has to be correctly received by the destination node. This includes the transmission delay plus maybe some retransmissions. It is important to highlight then that queueing delay captures several aspects of the wireless link operation: congestion at the MAC layer, transmission errors at PHY layer and the chosen modulation rate.

Let D_P be the average end-to-end delay of path P. Note that, as mentioned above, the throughput of path P is proportional to the inverse of D_P. This fact in addition to what we discussed above suggests the use of the average end-to-end queueing delay in the network $D(\boldsymbol{d}, \boldsymbol{\alpha})$ as a total congestion measure:

$$D(\boldsymbol{d}, \boldsymbol{\alpha}) := \sum_{s=1}^{S} \sum_{i=1}^{n_s} d_{P_{si}} D_P = \sum_{s=1}^{S} \sum_{i=1}^{n_s} \alpha_{P_{si}} d_s D_P \qquad (1)$$

Notice that this measure depends, on the one hand, of the vector $\boldsymbol{d}$, defined by the OD traffic demands, which cannot be set as desired because they are given by the network usage. On the other hand, the function also depends on the traffic distribution vector $\boldsymbol{\alpha}$, which we can control and will set so as to minimize the network congestion. Then, it is easy to prove that the sum over all the paths is equal to the sum over all the links, so we have:

$$D(\boldsymbol{d}, \boldsymbol{\alpha}) = \sum_{l=1}^{L} D_{l_1}(\rho_{l_1}, \rho_{l_2}) \rho_{l_1} + D_{l_2}(\rho_{l_2}, \rho_{l_1}) \rho_{l_2}$$

Let Q_{l_1} and Q_{l_2} be the mean amount of bytes on link l queues on each direction. Then, by Little's law we obtain the following result: $Q_{l_1} = D_{l_1} \times \rho_{l_1}$ and $Q_{l_2} = D_{l_2} \times \rho_{l_2}$. Finally $D(\boldsymbol{d}, \boldsymbol{\alpha})$ is given by:

$$D(\boldsymbol{d}, \boldsymbol{\alpha}) = \sum_{l=1}^{L} Q_{l_1}(\rho_{l_1}, \rho_{l_2}) + Q_{l_2}(\rho_{l_1}, \rho_{l_2})$$

$$= \sum_{l=1}^{L} Q_l(\rho_{l_1}, \rho_{l_2})$$

where Q_l is the average sum over both link queues (i.e. $Q_{l_1} + Q_{l_2}$ in Fig. 2). In Sec. 4 we will present a measurement-based scheme to characterize $Q_l(\rho_{l_1}, \rho_{l_2})$.

All in all, the dynamic load-balancing scheme should strive at solving the following problem:

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \ \ D(\boldsymbol{d}, \boldsymbol{\alpha}) = \sum_{l=1}^{L} Q_l(\rho_{l_1}, \rho_{l_2})$$

subject to:

$$\sum_{i=1}^{n_s} \alpha_{P_{si}} = 1 \ \ \forall \ s,$$

$$\alpha_{P_{si}} \geq 0 \ \ \forall \ s, i.$$

Let us further justify our choice of the objective function. Equation 1 suggests that our objective function may be regarded as a weighted average end-to-end delay, where the weight of each path is how much traffic is being sent along it. This means that $\sum_l Q_l$ considers both delay and throughput at the same time. Concerning dropping probability, the last of the three performance indicators cited before, it should be clear that a bigger value of it will result in a bigger queue at the output air interface, resulting in a bigger $\sum_l Q_l$. The conclusion of this discussion is that $\sum_l Q_l$ is a number that is affected by the three performance indicators, and as such reflects the three of them. We referred to this when we said before that $\sum_l Q_l$ is a versatile indicator.

4. Wireless Link Average Queue

In this section we present the procedure to choose the most appropriate $Q_l(\rho_{l_1}, \rho_{l_2})$ for every 802.11 link in the network. We shall omit the subindex l since the procedure is the same for

every link. The function $Q(\rho_1, \rho_2)$ is not trivial as we are dealing with 802.11 wireless links which use CSMA/CA as medium access control mechanism. Several works since [26] have tried to find the relation between wireless link parameters and the corresponding TCP and UDP achievable throughput. We use a different approach, that has already proved suitable for wired links [14], which is learning the function from measurements. This way we avoid using an arbitrary model and reflect reality as much as possible. However, the learning procedure should be carried out with some care. For instance, differently to the wired case, the average queue length at a given link is now a bivariable function, because it depends not only on the incoming traffic, but also on the traffic in the opposite direction.

Assume we have a set of N measurements $\{Q_1, Q_2, ...Q_N\}$ for the corresponding values $\{(\rho_{1_1}, \rho_{2_1}), (\rho_{1_2}, \rho_{2_2}), \ ... \ (\rho_{1_N}, \rho_{2_N})\}$ (also called training set). Assume that the response variable Q (the average queue length measurement) is related to (ρ_1, ρ_2) (the link average traffic loads measurements) by the following equation:

$$Q = f(\rho_1, \rho_2) + \epsilon$$

where ϵ is the measurement error and is modelled as a random variable such that $\mathrm{E}\{\epsilon\} = 0$ and $\mathrm{Var}\{\epsilon\} = \sigma < \infty$. The Weighted Least Squares (WLS) problem consists in finding the function f that minimizes the weighted sum of quadratic errors, assuming that f belongs to a given family of functions $\mathcal{F}$. The weights represent the relative importance of each measurement point with respect to the rest of the measurements in the training set.

We present a method that restrict the assumptions on the family of functions $\mathcal{F}$ to the minimum. Regarding its shape, we have only two necessary assumptions: (i) $f(\rho_1, \rho_2)$ should be non-decreasing, since more load may never mean less queue length; (ii) $f(\rho_1, \rho_2)$ should be convex in order to guarantee the existence and uniqueness of the optimum demand vector (later on we will discuss on this assumption). We then consider $\mathcal{F}$ as the family of continuous, monotonous increasing and convex functions. This WLS problem with such $\mathcal{F}$ is called Convex Non-parametric Weighted Least Squares (CNWLS), a variation of the original unweighted Convex Non-parametric Least Squares (CNLS) [27]. The size of $\mathcal{F}$ makes this problem very difficult to solve in such general form, which motivates to use instead a subfamily of $\mathcal{F}$, the piecewise linear functions included in $\mathcal{F}$. This lead us to a standard finite dimensional Quadratic Programming (QP) problem in order to solve the regression, for which mature methods to solve it exist (e.g. interior point algorithms) and several solver software are available (for instance, we used MOSEK [28]). This scheme is easily adaptable to update the function in real time through online learning as new measurements are gathered from the network. This fact could be useful to react properly to physical changes that may affect the link capacity (e.g. antenna misalignment or environmental changes).

We will now discuss on the convexity assumption mentioned before. A necessary condition for the convexity of $Q(\rho_1, \rho_2)$ is that the feasible region of the link is convex (i.e. the set of $\{(\rho_1, \rho_2)\}$ such that $Q(\rho_1, \rho_2) < \infty$). Several previous works studied the feasible region for 802.11 wireless multihop networks. This region is known to be not necessarily convex, which is demonstrated in [29] with models and simulations for different topologies. This fact is also analyzed in [30], where the log-convexity of this region is established, a fact that is taken as a basis for characterising max-min fair rate allocations for 802.11 WMNs in [31]. However, the model presented in [32] approximates the feasible region by a convex polytope. The procedure is based on the computation of extreme points in order to get the polytope convex hull (boundary) and it is shown that most of the cases presented in [29] can be adequately captured by this model.

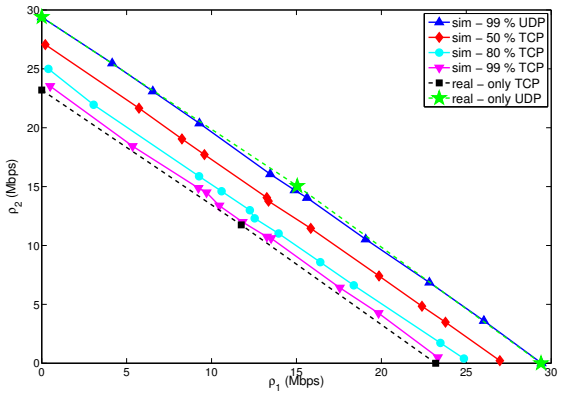


Figure 3: Feasible region analysis for a 802.11a link @54Mbps.

For the case we are considering in this paper, a planned WMN, the analysis is much simpler because we have only two nodes that can interfere with each other (i.e. the endpoints of each link). This simplifies the feasibility region analysis to the study of the behaviour of only one link as the traffic loads in both directions changes. For this purpose, first let us take a look at the well-known Bianchi model [26] to notice that the capacity for two nodes is 2.5% larger than for a single node. This fact indicates that two simultaneously transmitting nodes may support more traffic than only one, which means that feasible region of a 802.11 point to point link should be convex. We further studied the feasible region for a 802.11a link operating at 54 Mbps with simulations performed with the ns-3 simulator [33] and real data measurements. In Fig. 3 we can see the results for different traffic compositions combining TCP and UDP flows. As we can see, the feasible region increases as the proportion of UDP traffic increase, with throughput ranging from 24 to almost 30 Mbps. It is clear from the results that for all cases it is suitable to use a convex model as an approximation, as used in [32].

4.1. Average queue regression example

In order to illustrate the proposed procedure we will show an example with simulations performed with ns-3. We configured a wireless link operating in 802.11a, with a distance = 100m between nodes and fixed RSS = −65 dBm as propagation model. This implies that the link is always operating at the same modulation rate (54Mbps in this case).

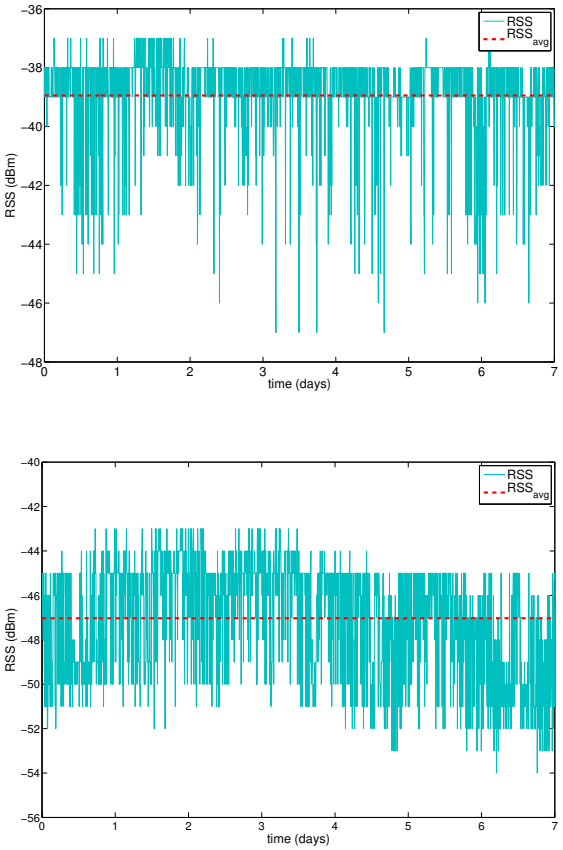


Figure 4: RSS measurements for two real 802.11 links.

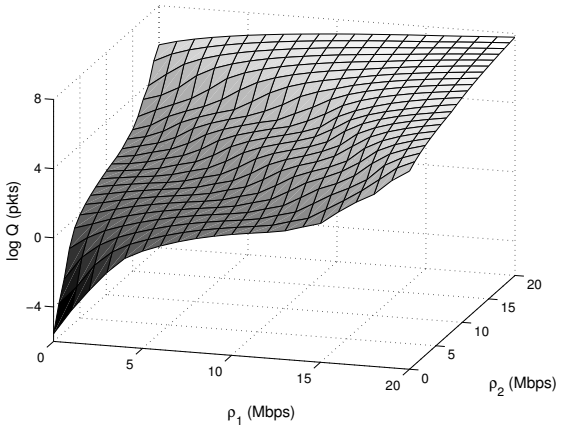


Figure 5: Learnt function (log-scale) for the average queue size.

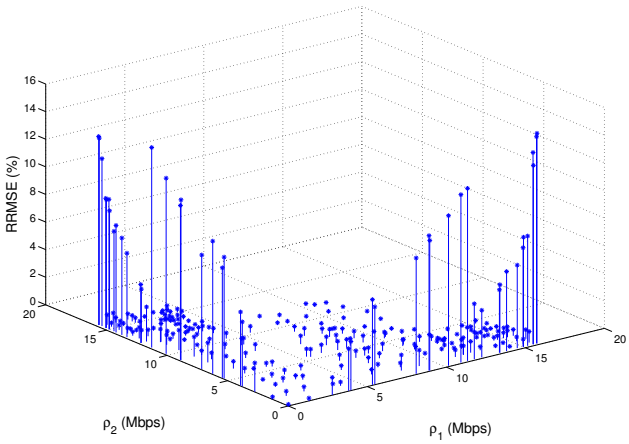


Figure 6: Relative RMSE for the example test data.

As we said before, we are considering a WMN where links performance is stable and predictable, with a strong correlation between the error rate and the received signal strength. Under this assumption, if we do not have much RSS variation for our network links, we will not have variation at all on each link modulation rate. This assumption is valid for a wide range of WMNs, not only in rural or suburban areas, but also in some urban scenarios with LOS links using directional antennas. As an example in Fig. 4 we show the RSS for one week for two urban links from Plan Ceibal network. Both of them operate with line of sight and with an approximated distance of 200m between nodes. As we can see the RSS variation is not significant and enables a stable link operation at a fixed modulation rate, as the receiver sensitivity for 54Mbps is -71dBm. This fact is consistent with the data shown in [15].

Now, we present an example for one link to illustrate the procedure followed for every link in the network in the learning phase. In this example we generated a dataset of 484 measurements, 228 used for learning the function and the remaining 256 for testing the regression performance. To generate each flow with the desired traffic load ρ, we used a combination of random TCP and UDP flows (80% and 20% respectively). TCP flows were generated with exponential file sizes with mean 500 Kbytes. UDP flows were generated with a fixed rate of 100 kbps and exponential length with mean 30 seconds. The arrival rate distribution was also exponential for both cases, with mean according with the desired traffic loads (0.8ρ and 0.2ρ respectively). Each measurement corresponds to the average traffic load in both directions (ρ_1, ρ_2) and the average queue length Q, where averages are considered over 100 seconds.

In Fig. 5 we present the resulting function after the regression in logarithmic scale (for the sake of clarity). Queue size is expressed in packets because both ns-3 simulator and typical wireless equipment use 802.11 packet-based queues [34]. The RMSE for training data was 4.3 packets, while the RMSE for test data was 5.5 packets. The relative RMSE for training data was 2.9% with a maximum of 14.2%, while for test data was 4% with a maximum of 15.2%. The RRMSE for test data is shown in Fig. 6. This results show that the function approximation is suitable.

The presented example is only to show the procedure we followed for every link in the network in the learning phase. Once we have learned the function $Q_l(\rho_{l_1}, \rho_{l_2})$ for each link, we are in position to tackle the optimization problem defined in Sec. 3. The forwarding decision will come up from the optimum traffic distribution vector α which minimizes the total network congestion.

5. Optimal forwarding proposal

In order to drive the network to the desired operation point, we have to solve the optimization problem detailed in Sec. 3:

$$\min_{\boldsymbol{\alpha}} \sum_{l=1}^{L} Q_l\left(\rho_{l_1}, \rho_{l_2}\right); \text{s.t.} \sum_{i=1}^{n_s} \alpha_{P_{si}} = 1, \alpha_{P_{si}} \geq 0.$$

For this purpose, we used a gradient descent method to iteratively update the traffic distribution vector $\boldsymbol{\alpha}$ by setting the proper load balance leading to the optimum. We can assure that there are no local minima because we are minimizing a sum of convex functions, which is also a convex function. To start the optimization algorithm we need an initialization step, so certain initial values have to be set to enable the network to begin the operation. Then, we consider a periodic update every ΔT seconds, given by:

$$\boldsymbol{\alpha}^{t+\Delta T} = \boldsymbol{\alpha}^{t} - \gamma \cdot \nabla \left(\sum_{l=1}^{L} Q_l\left(\rho_{l_1}, \rho_{l_2}\right) \right)$$

where γ is the gradient descent step size. Before updating $\boldsymbol{\alpha}$ we have a normalization step to guarantee the constraints on $\alpha_{P_{si}}$. With this procedure the demands are periodically adjusted, using the following equation for updating the traffic distribution coefficient which corresponds to the path P_{si}:

$$\hat{\alpha}_{P_{si}}^{t+\Delta T} = \left[\alpha_{P_{si}}^{t} - \gamma \sum_{l:l \in P_{si}} \frac{\partial Q_l}{\partial \rho_{l_{si}}} \left(\rho_{l_1}^{t}, \rho_{l_2}^{t} \right) \right]^{+} \tag{2}$$

$$\alpha_{P_{si}}^{t+\Delta T} = \hat{\alpha}_{P_{si}}^{t+\Delta T} / \sum_{i=1}^{n_s} \hat{\alpha}_{P_{si}}^{t+\Delta T} \tag{3}$$

Notice that the partial derivatives in the second term are with respect to $\rho_{l_{si}}$, which is the traffic load of link l in the direction that corresponds to path P_{si}. This fact implies that for updating the traffic distribution coefficients $\alpha_{P_{si}}$ we only need to know the learned functions for the links used by the path P_{si}, which means that edge routers only need information from the intermediate routers included in the pre-established paths they will use, enabling a decentralized implementation of the algorithm. All the notation used in this paper is summarized in Tab. 1.

The complete network operation is defined by the three processes: measurement-based learning of the objective function, update of traffic demands distribution via gradient descent optimization and packet forwarding on a per-flow basis. These processes operate at different time scales as shown in Fig. 7.

At the longer time scale we have the measurement-based learning of the average queue length function, which takes several hours of information to update the $Q_l\left(\rho_{l_1}, \rho_{l_2}\right)$ for every link in the network following the procedure described in the previous section.

Then we have the update of traffic demands distribution in order to lead the network to the minimum queue length load balance (i.e. for each OD pair we use Eqs. 2 and 3). In this case each iteration is performed at a smaller time scale than model learning, but a much longer time scale than packet forwarding. The optimization takes into account average values, so we need an update period long enough to take good quality average measurements. On the other hand, this period should not be excessive in order to be able to respond quickly when traffic conditions change abruptly. Typically a suitable period is some tens of seconds, which is the minimum time to get reasonable average measurements (e.g. we used 100 seconds).

Table 1: Index of key notations.

Variable	Description
$1, .., n, .., N$	Set of nodes (i.e., wireless mesh routers)
$1, .., l, .., L$	Set of bidirectional links
$1, .., s, .., S$	Set of OD pairs
d_s	Average traffic demand for OD pair s
n_s	Number of paths for OD pair s
P_{si}	i-th path for OD pair s
$d_{P_{si}}$	Average amount of traffic for path P_{si}
$\boldsymbol{d}$	Average traffic demands vector
$\alpha_{P_{si}}$	Traffic distribution coefficient for path P_{si}
$\boldsymbol{\alpha}$	Traffic distribution vector
ρ_{l_1}, ρ_{l_2}	Average traffic load on link l for each direction
D_{l_1}	Average delay at link l in the direction of load ρ_{l_1}
D_P	Average delay at path P
Q_{l_1}	Average queue size at link l in the direction of load ρ_{l_1}
Q_l	Sum of the average queues sizes at link l
$\alpha_{P_{si}}^{t}$	Traffic distribution coefficient for path P_{si} at time t
γ	Gradient descent step size

The smaller time scale corresponds to the packet forwarding, which is performed with flow granularity. This means that every new traffic flow at an ingress router corresponding to OD pair s is associated with a certain path P_{si} with probability $\alpha_{P_{si}}$. Let us recall that we have certain pre-established paths defined by the network topology. This packet forwarding scheme is very similar to the one used in wired networks with MPLS. Several paths are defined at edge routers, where incoming traffic is labelled according to the corresponding path and then packet forwarding at relay routers is based on labels. This is why we say that our proposal of separating routing from forwarding is

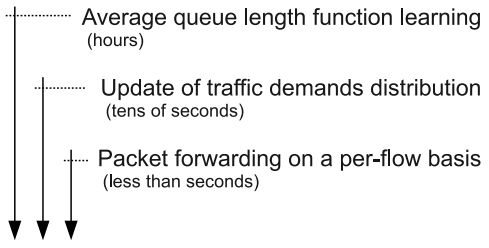


Figure 7: Processes involved in the proposed framework.

a solution *à la* MPLS.

With respect to the running time of each action, its precise value depends on the specific hardware at use (e.g. ingress router, relay nodes). However, it is clear that the more costly actions are the ones that operate at a larger timescale (i.e. function learning costs more than gradient descent and both of them more than forwarding).

5.1. Implementation Issues

The application of the proposed framework in a real-world network is relatively simple. First of all we need a routing protocol to establish the multiple routes for each OD pair defined by the wireless network topology. Once we have learnt Q_l for every link l, each ingress router receives the values ρ_l from the links used by the OD flows with origin in that ingress router. A routing protocol that supports information distribution such as OSPF-TE may be used for this purpose. With that information, each ingress router is able to update the traffic portion that has to be routed through each path. This process is repeated indefinitely every some seconds.

With respect to the flow-based multipath forwarding implementation, the idea is to use an MPLS-based solution, similar to the wired case. Although an standard of MPLS over WMNs does not exist yet, several proposals were already presented. For example in [24] the proposal considers traffic splitting at every router and optimization over the average of all possible traffic matrices. Our proposal could be implemented reusing the same splitting-based scheme, but considering splitting only at ingress routers over all the different end-to-end paths and enabling dynamic load-balancing for the average load at each moment.

Regarding the learning phase we envisage several possibilities differing in the resulting architecture. One possibility is that a central entity gathers the measurements, performs the regression and communicates the obtained parameters to all ingress routers. This option has the advantage that the required new functionalities on routers are minimal. However, as all centralized arquitectures, it may not be suitable for some network scenarios, and handling the failure of this central entity could be very complicated. An alternative is that for each wireless link only the two directly involved routers perform the regression. They should keep the average queue size measurements for themselves, perform the regression and communicate the result to the ingress routers.

Another aspect that has different possibilities is what characterization (i.e. Q_l learnt function) use at each moment and which measurements to keep for the training set. Measurements could be gathered every day, the regression performed, and its result could be used the next day or the same day the next week. In addition, it is clear that newer measurements should be given priority over older ones. A possible way to manage training data is to keep always the newer measurements and use weights in the regression to introduce temporal information (e.g. exponential decay). It may also be necessary to force keeping particular measurements to ensure a proper coverage density of the whole load value range.

Concerning the number of measurements needed for training, we now show how the considered learning algorithm (CNWLS) does not need a large number of measurements, as long as the training samples adequately covers the whole range of possible values. In Fig. 8 we show the test error analysis for training sets with different sizes, using the same data as in the example discussed in section 4.1. In particular, for each size, we randomly sampled several training sets (we used 20) and computed the corresponding average RRMSE with the test data for the resulting learned function. As we can see, the RRMSE is always below 10% with only 60 training samples and falls below 5% with more than 150 samples.

Finally, rare events like node failures or changes in propagation conditions can be taken into account in our framework as follows. If interference on a particular link changes, this is captured when the learning of the function associated with that link is repeated. As we mentioned before, this learning process is periodically repeated. However, if several new measurements differ greatly from the learned model, one could decide to trigger a new learning process. Moreover, if a node fails, the ingress routers will not receive the corresponding link load information. If no such announcements are received for a certain period of time, this should lead to the decision of disabling all paths that use the faulty router.

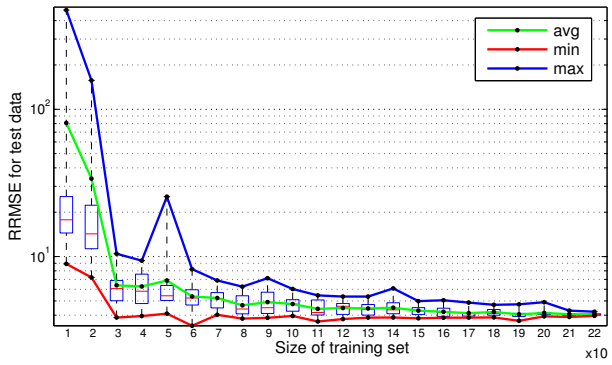


Figure 8: Training set size analysis. On each box, the central red mark is the median and the edges of the box are the 25th and 75th percentiles.

6. Simulation experiments

In order to validate the framework we tested the proposed minimum queue length load-balancing (MQLLB from now on) algorithm with simulations performed with ns-3. Most of the examples considered correspond to canonical topologies of WMNs [29] but also to typical configurations in real deployments (e.g. Plan Ceibal network [5]).

In this paper we will present four examples. The first one is a three node topology used to describe the framework operation. In the second example we illustrate the gateway selection problem which can be solved within the same proposed framework. The third example corresponds to a four node topology where we deeply analyze the advantage of the proposed model under asymmetric traffic demands, comparing the performance with

IEEE 802.11s. Finally, we present an example with a larger network, a 25 node uniform square grid, where we analyze convergence and scalability of the algorithm.

In all the examples the traffic considered is the same described before in Sec. 4 with a combination of TCP and UDP flows (80% and 20% respectively), both of them with exponential arrival rates. We also used exponential distributions for the file size (in case of TCP flows) and length (in case of UDP flows), with the same characteristics mentioned for the model learning example shown before. Wireless links were set to the standard 802.11a with a distance of 100m between nodes, while the propagation model used was fixed received signal strength (RSS = −65 dBm) which implies that links always operate at the same modulation rate (@54Mbps). The buffer size for each interface is 400 packets (ns-3 default) which is consistent with typical wireless equipment [34]. In every case, we used 235 measurements in the learning phase for each link, which is approximately 6.5 hours of training data. Then, we implemented the MQLLB method which uses the described optimization framework to iteratively update $\boldsymbol{\alpha}$, taking the forwarding decision with a flow level granularity.

For performance comparison we considered as a benchmark the IEEE 802.11s routing scheme, which uses HWMP (Hybrid Wireless Mesh Protocol) to compute paths. We think this benchmark is the most suitable one, as HWMP is the only algorithm included in an approved standard up to date and can be used by everyone to compare with. In addition, there are implementations available as the one included in the ns-3 simulator. Such protocol uses a routing metric called *airtime* metric which is designed to represent the channel resources needed for a frame to be transmitted over a wireless link and is calculated as follows:

$$airtime = \left(O_{ca} + O_p + \frac{B_t}{r}\right)\frac{1}{1 - e_{fr}}$$

where O_{ca}, O_p, and B_t are constants quantifying respectively the Channel Access Overhead, the Protocol Overhead, and the number of Bits in a probe frame. O_{ca} and O_p depend solely on the underlying modulation scheme, r is the transmission rate, and e_{fr} is the frame error rate. This routing metric is similar to ETX (Expected Transmission Count) and ETT (Expected Transmission Time) [8, 16]. However, *airtime* further accounts for channel access and protocol overheads. An implementation of 802.11s is available in the ns-3 simulator.

For performance analysis and comparison we considered three metrics: average delay and jitter of UDP flows and average goodput of TCP flows, which corresponds to the amount of data per second carried by TCP flows discarding TCP ACKs. The analysis for each flow was done using the ns-3 flow monitor [35] which enables flow level statistical analysis of the simulation. We compared the results with the 802.11s performance for the different scenarios. We also considered static routing as a different alternative, using shortest path routing with hop count as metric.

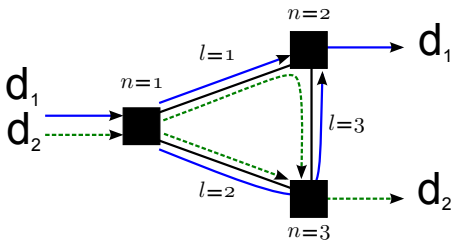


Figure 9: 3-nodes topology multipath forwarding example.

6.1. Multipath forwarding: 3-nodes topology

The first example is presented to illustrate the framework and corresponds to the topology and flows shown in Fig. 9. This topology has three links 1, 2 and 3, which implies we have also three functions Q_1, Q_2 and Q_3, each of them corresponding to the sum of the link queues in both directions. In this case we considered flows from node 1 to nodes 2 and 3 with traffic loads d_1 and d_2 respectively. When we apply the described framework to this particular topology and the considered traffic flows, we have the following function for the average end to end queueing delay in the network:

$$D(\boldsymbol{d}, \boldsymbol{\alpha}) = Q_1(\rho_{1_1}, \rho_{1_2}) + Q_2(\rho_{2_1}, \rho_{2_2}) + Q_3(\rho_{3_1}, \rho_{3_2})$$

For each OD pair we have two possible paths:

- $P_{11} = \{1, 2\}$ and $P_{12} = \{1, 3, 2\}$ for d_1.
- $P_{21} = \{1, 2, 3\}$ and $P_{22} = \{1, 3\}$ for d_2.

We will call $\alpha_{P_{11}}$ the portion of traffic d_1 that is routed through path P_{11}, which leaves $\alpha_{P_{12}} = 1 - \alpha_{P_{11}}$ through path P_{12}. We will call $\alpha_{P_{21}}$ the portion of traffic d_2 that is routed through path P_{21}, which leaves $\alpha_{P_{22}} = 1 - \alpha_{P_{21}}$ through path P_{22}. Functions Q_1, Q_2 and Q_3 are learned from previous measurements following the procedure described in Sec. 4. Then, in order to find the optimum forwarding decision for a particular combination of the considered traffic flows, we have to find the optimum values of $\alpha_{P_{si}}$ which lead us to the minimum network congestion. The proposed framework applied to this particular case leads us to the following optimization problem:

$$\min_{\alpha_{P_{11}}, \alpha_{P_{21}}, \alpha_{P_{12}}, \alpha_{P_{22}}} \quad Q_1 + Q_2 + Q_3$$

$$\text{subject to: } \alpha_{P_{11}} + \alpha_{P_{12}} = 1$$

$$\alpha_{P_{21}} + \alpha_{P_{22}} = 1$$

$$\alpha_{P_{11}}, \alpha_{P_{21}}, \alpha_{P_{12}}, \alpha_{P_{22}} \geq 0$$

Then, in order to update $\alpha_{P_{11}}$ (for $\alpha_{P_{21}}$ is analogous) we have to use the following equations:

$$\hat{\alpha}_{P_{11}}^{t+\Delta T} = \left[\alpha_{P_{11}}^{t} - \gamma\left(\frac{\partial Q_1}{d\rho_{1_1}} - \frac{\partial Q_2}{d\rho_{2_2}} - \frac{\partial Q_3}{d\rho_{3_2}}\right)\right]^{+}$$

$$\alpha_{P_{11}}^{t+\Delta T} = \min\left(\hat{\alpha}_{P_{11}}^{t+\Delta T}, 1\right)$$

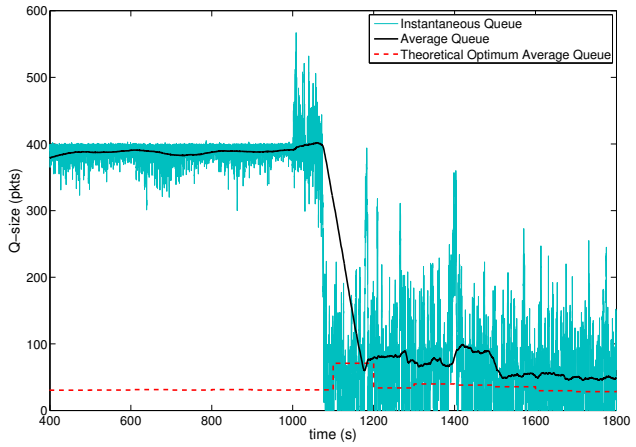


Figure 10: Total queue for the 3-nodes topology symmetric case.

Method	UDP flows Delay (ms)	UDP flows Jitter (ms)	TCP flows Goodput (Mbps)
MQLLB	14.6	6.7	15.2
802.11s	17.5	7.6	13.9
static routing	14.3	7.1	15.4

Table 2: Performance metrics for the 3-nodes topology symmetric case.

In order to choose the most appropriate function Q_l for each link we followed the measurement-based method described in Sec. 4. In the learning phase, to generate the training data we used simulations with different traffic distribution coefficients $\alpha_{P_{si}}$, uniformly covering all the possibles values. Then, to calculate the partial derivatives of each link queue Q_l we used the learned functions in order to periodically update the $\alpha_{P_{si}}$.

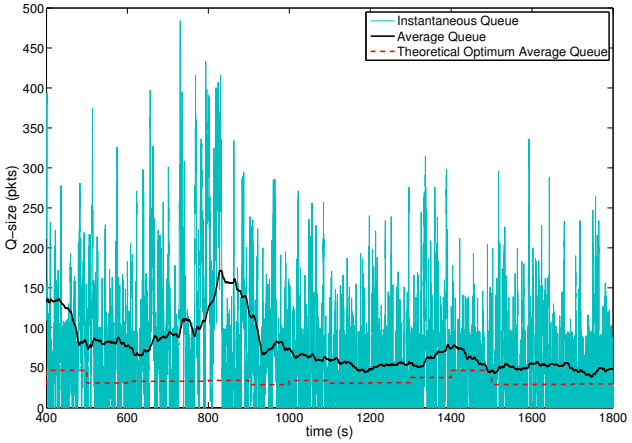
Now, we will present the simulation results using the presented framework for two different traffic loads: symmetric and asymmetric cases. First we will show a symmetric example where traffic loads were $d_1 = d_2 = 13$ Mbps. In Fig. 10 we can see the evolution during the simulation of the total queue size (expressed in packets), which corresponds to the sum of all interfaces queues in the network. We present the comparison of the instantaneous queue size and the 100-seconds average with the theoretical optimum queue length, which is calculated from the learned model and the traffic average measures. We show from time $t = 400$s, when we have already reached steady state, starting with $\alpha_{P_{11}} = 1$ and $\alpha_{P_{21}} = 1$ (i.e. both flows forwarded through link 1), which causes saturation at link 1 and we start using MQLLB at time $t = 1000$s. Concerning the performance metrics, the results are summarized in Table 2. We can see that none of the metrics show significant differences between the three alternatives. It is clear that with symmetric traffic as in this case, static routing through shortest paths is a good alternative, as the results reflect. Notice that 802.11s presents slightly worse results, something which will be deeper analyzed in the next simulations.

The other example with the three-node topology corresponds to an asymmetric case, where traffic loads were $d_1 = 20$ Mbps, $d_2 = 5$ Mbps. We started the simulation with $\alpha_{P_{11}} = 1$ and

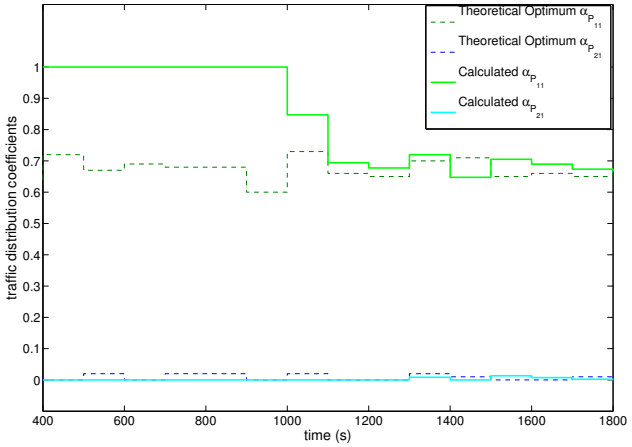
Method	UDP flows Delay (ms)	UDP flows Jitter (ms)	TCP flows Goodput (Mbps)
MQLLB	14.3	6.6	15.5
802.11s	43.1	8.9	9.6
static routing	35.1	8.6	11.2

Table 3: Performance metrics for the 3-nodes topology asymmetric case.

$\alpha_{P_{21}} = 0$ (i.e. only the one-hop path for each OD pair). In Fig. 11(a) we can see the total queue size evolution from time $t = 400$s. We started the operation of MQLLB at time $t = 1000$s and as we can see the average queue size goes down which means the network is better load balanced. Fig. 11(b) shows the traffic distribution coefficients evolution. Notice that at time $t = 1100$s, when the second update round happens, we already reached the optimum load-balancing. Looking at performance metrics shown in Table 3, we can see that the difference is clear in favour of MQLLB in this case where we have asymmetric traffic. As expected, for the asymmetric example we have an important improvement in the network performance due to the load-balancing mechanism.



(a) Total queue size evolution.



(b) Traffic distribution coefficients evolution.

Figure 11: 3-nodes topology asymmetric case simulation.

Method	UDP flows Delay (ms)	UDP flows Jitter (ms)	TCP flows Goodput (Mbps)
MQLLB	21.4	8.1	10.8
static routing	48.4	9.2	8.3

Table 4: Performance metrics for the gateway selection asymmetric case.

6.2. *Gateway selection problem*

In this subsection we will analyze an example corresponding to the gateway selection scenario shown in Fig. 12. We will show that it is possible to solve this problem under the proposed framework, treated as an equivalent multipath forwarding one. In this topology we considered downlink flows to nodes 3 and 4, with demands d_1 and d_2 respectively, which can be distributed between the two gateways GW 1 and GW 2. Notice that both gateways could be considered as the same traffic origin (internet). We can think this origin as a *super* node, connected to both gateways by links with infinite capacity (shown with dashed lines in Fig. 12). Then, the gateway selection problem turns into a multipath forwarding problem, where we have to decide which portion of traffic demands d_1 and d_2 to forward through each of the possible paths from the *super* node (internet), which is equivalent to decide which portion of traffic to route from each gateway.

In this example, we also considered inter-gateways flows from node 1 to node 2 and viceversa, with demands d_3 (from 1 to 2) and d_4 (from 2 to 1) respectively. This traffic flows may exist due to mobile hosts directly attached to one gateway that access resources allocated at servers in the other gateway. There is only one possible path for this flows, so there is no forwarding decision to take for that OD pairs. However, they affect the amount of traffic on each link, which leads the network to a different load condition than the one without inter-gateways flows. It is a desirable property of the algorithm that the existence of those inter-gateways flows do not affect the forwarding decision of the other flows.

We will analyze an asymmetric simulation example where traffic loads are $d_1 = 15$ Mbps, $d_2 = 5$ Mbps and $d_3 = d_4 = 3$ Mbps. In this case network started operating with shortest path routing with hop count as routing metric (i.e. d_1 through GW 1 and d_2 through GW 2). The heavy traffic load from GW 1 to node 3 produces congestion in that link, which is visible in Fig. 13(a) where the total average queue evolution is shown from $t = 400$s, when we have already reached steady state. The operation of MQLLB starts at $t = 1000$s and reached convergence at $t = 1200$s. The final total average queue length as we reached convergence is 79 packets, which is almost 50% smaller than before starting MQLLB where it was 154 packets (with peaks up to 235). In Fig. 13(b) we show the average packet delay analysis for UDP flows. Please note that the x-axis does not correspond to time but to the flow index. It is clear that after MQLLB starts there is an important improvement with a smaller average delay. Performance metrics are summarized in Table 4, where we compare the results of MQLLB with static routing through the nearest gateway (802.11s was not considered in this gateway selection example). It is clear the advantage of using MQLLB in this case, particularly noticeable in the UDP flows delay with an improvement of more than 50%.

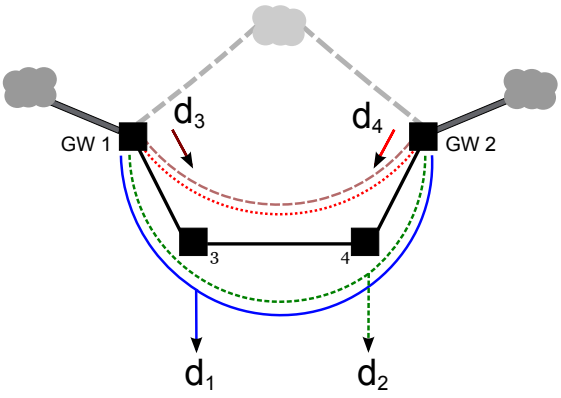
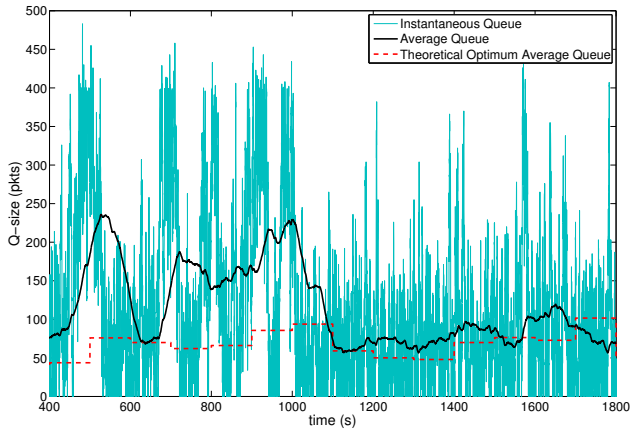
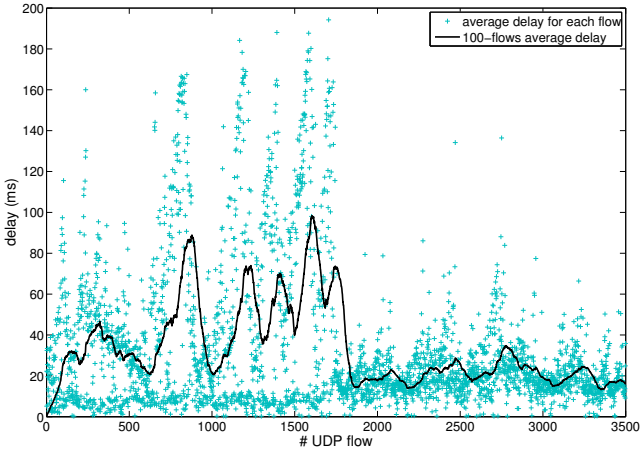


Figure 12: Gateway selection problem.



(a) Total queue size evolution.



(b) Average packet delay for UDP flows.

Figure 13: Gateway selection with asymmetric traffic loads.

For the gateway selection problem there is an important issue to solve for a real-world implementation. For the downlink case (traffic coming from the internet) we cannot perform path selec-

Method	UDP flows Delay (ms)	UDP flows Jitter (ms)	TCP flows Goodput (Mbps)
MQLLB	19.0	6.7	15.4
802.11s	141.3	8.0	4.8
static routing	104.0	8.4	6.8

Table 5: Performance metrics for 4-node topology example, situation 1.

tion at the ingress routers (i.e. the gateways) since we are distributing traffic between paths that do not share the same origin node. A simple alternative to solve this issue is to make gateway selection with client granularity. In this case, the routers which are directly connected to mobile hosts may decide the proper gateway for each client. In order to improve the performance of this approach these routers could monitor each client traffic demand. Thus, the optimization process could use a client granularity but including the client demand information, which allows a better traffic forwarding update at each step.

6.3. Multipath forwarding: 4-node topology

The next example correspond to a four nodes topology with five 802.11 links and two OD pairs (see Figs. 14 and 17). In this scenario we have three possible paths for each OD pair, each of them of distance 1, 2 and 3 links. We will consider only the two shortest paths for each one, so we have to decide for each OD pair, how much traffic to forward on each route. As we said before, the possible paths for each OD pair are defined by the network topology, but we can decide not to use any given path by configuration, because we want to simplify the network operation or just avoid the usage of a particular path. We will consider two different situations, both of them with asymmetric traffic demands, but the difference between them is how the paths share the different links.

First, we will analyze the situation shown in Fig. 14, where both flows are from left to right, so links are shared by flows in the same direction. We simulated the scenario with d_1 = 25 Mbps and d_2 = 10 Mbps and compared the performance of MQLLB with 802.11s. Both simulations have a total duration of 2500s, in one case beginning with static routing using only the single-hop paths and MQLLB starting at time 500s and in the other case using 802.11s during all the simulation. The different performance metrics analyzed show a clear advantage of MQLLB over 802.11s and static routing. The results are summarized in Table 5, where we can see an improvement of more than 70% in the average delay for UDP flows and more than 100% in the average goodput for TCP flows. In Figs. 15(a)

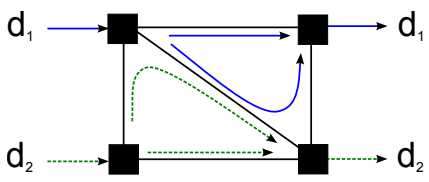
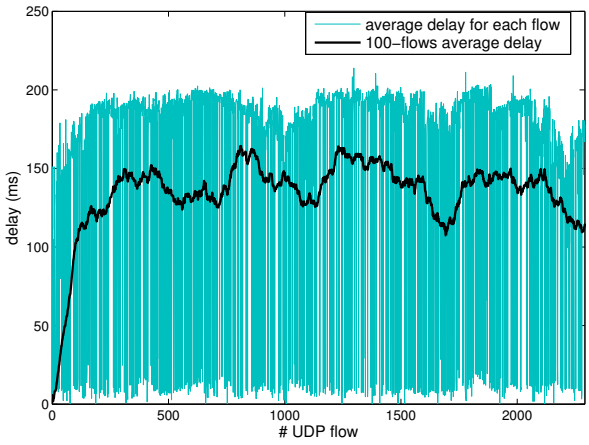
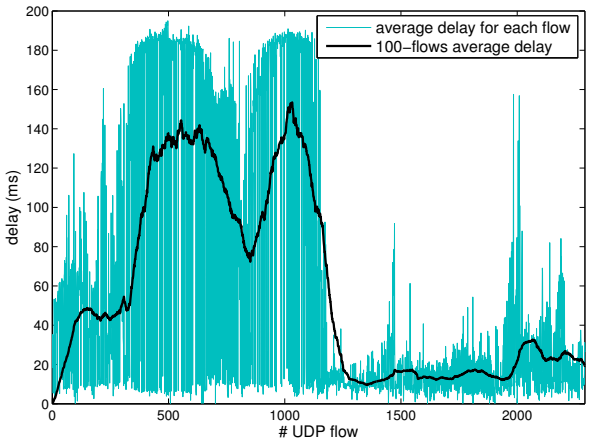


Figure 14: 4-node topology multipath forwarding example, situation 1.



(a) Simulation with 802.11s.



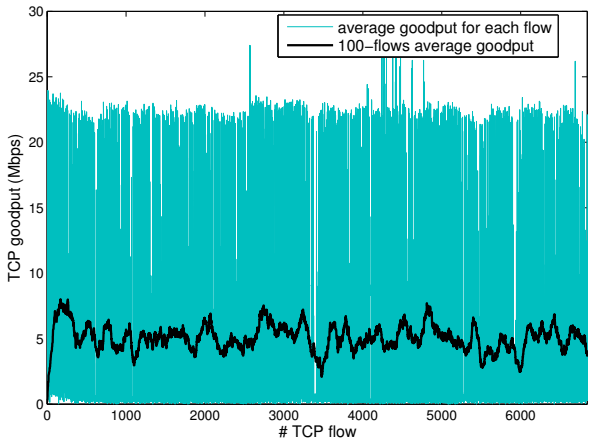
(b) Simulation with static routing and MQLLB.

Figure 15: UDP flows average delay analysis for 4-node topology example, situation 1.

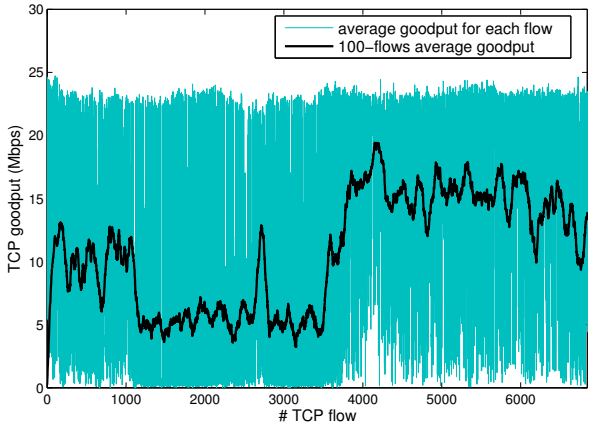
and 15(b) we show the average packet delay evolution for UDP flows in time order during the first 1000s of the simulations. Similarly, in Figs. 16(a) and 16(b) we show the average goodput evolution for TCP flows. In both cases it is clear the moment when MQLLB starts the operation (at 500s), which is reflected on the network performance with a smaller average delay for UDP flows and a larger average goodput for TCP flows.

The other considered situation is shown in Fig. 17. The traffic loads are the same than before (d_1 = 25 Mbps and d_2 = 10 Mbps), but now d_1 is from left to right and d_2 from right to left, so links are shared by flows in the opposite direction. The results, which are summarized in Table 6, are quite similar to the previous situation, with significant improvements in all the analyzed performance metrics in favour of MQLLB. The purpose of this example is to show the ability of the proposed framework to cope with different link sharing situations, with traffic demands sharing the links both in the same direction or in opposite directions.

To explain the improvements of using an scheme like MQLLB instead of 802.11s, we must first note the advantage of considering multiple paths for each origin-destination pair,



(a) Simulation with 802.11s.



(b) Simulation with static routing and MQLLB.

Figure 16: TCP flows average goodput analysis for 4-node topology example, situation 1.

which allows a better adaptation to the particular traffic conditions. This fact is particularly clear when we analyze asymmetric traffic situations like the one of the examples. Second, we must consider the problems of using a metric that reflects the dynamics of each link at each moment as the *airtime* used by 802.11s. As studied in [11] routing oscillations may happen because of the dynamics of the different links metric. When more traffic is forwarded through a link, the metric is degraded, which causes that quickly we can find an unloaded link with a better metric. This fact causes that the node will change the selected path and it will start forwarding the traffic on the other link. The new selected link will suffer the same metric degradation that the other one had before, so the node will change the

Method	UDP flows Delay (ms)	UDP flows Jitter (ms)	TCP flows Goodput (Mbps)
MQLLB	25.9	7.2	13.9
802.11s	141.6	8.4	4.7
static routing	101.9	8.2	6.8

Table 6: Performance metrics for 4-node topology example, situation 2.

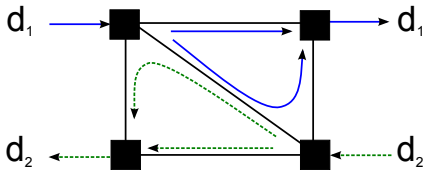


Figure 17: 4-node topology multipath forwarding example, situation 2.

selected path again. This phenomenon is repeated indefinitely generating an oscillation of the chosen path. This phenomenon was also noticed in [36] where it was called "ping-pong" effect, and the results reported in that work were similar with the ETX metric. This fact explains the bad performance of 802.11s, which is even worse than the one for static routing through one-hop paths in this examples. The proposed MQLLB uses average measurements to reflect the dynamics which allows a quick adaptation to traffic changes but ensuring an stable operation for steady state situations.

6.4. Gateway selection: 25-node topology

Finally, we present a gateway selection scenario in a 25-node topology to take a look into scalability and convergence of the proposed framework. The nodes are disposed in a 5 x 5 uniform square grid with side 500m and links are established between the closest nodes, all with a 100m distance. We call each node n_{ij} using matrix notation and we have two gateways corresponding to nodes n_{15} and n_{51} (top right and bottom left of the square respectively). We have a routing protocol (OSPF) which establishes routes between every pair of nodes, so, as we have two gateways, each node has two possibles routes to the internet. We will use the proposed method to find the proper traffic distribution between gateways for each node, which is called in this example α_{ij} ($\alpha = 1$ means all the traffic comes from n_{15}).

The traffic considered in this example is all downlink (from the gateways to the other nodes) and it was generated with the same characteristics as in previous examples. In the simulation, we started with $\alpha_{ij} = 0.5$ for all nodes, which corresponds to half of the traffic coming from each gateway for all of them. The load values used in the simulation were 5Mbps for nodes $\{n_{11}, n_{12}, n_{13}, n_{14}, n_{52}, n_{53}, n_{54}, n_{55}\}$ and 2.5Mbps for the rest of the nodes.

We enabled the operation of MQLLB at $t = 300$s. In Fig. 18 we show the evolution of the traffic distribution α_{ij} for each node, while for the gateways we show the total traffic load that comes from each of them. We can see that all the nodes which are at the same distance from each gateway (nodes n_{ii}, at 4-hop distance to gateways) remained with $\alpha_{ii} = 0.5$ during all the simulation. On the other hand, nodes which are closer to gateway n_{15} changed to $\alpha_{ij} = 1$ while the ones closer to gateway n_{51} changed to $\alpha_{ij} = 0$. This means that nodes with one gateway closer than the other, change the traffic distribution in order to receive all the traffic from the closest gateway. Taking into account the convergence, we can see that nodes which are closer to gateways converge in less optimization steps than the others. For example, looking at gateway n_{51} we can see that nodes

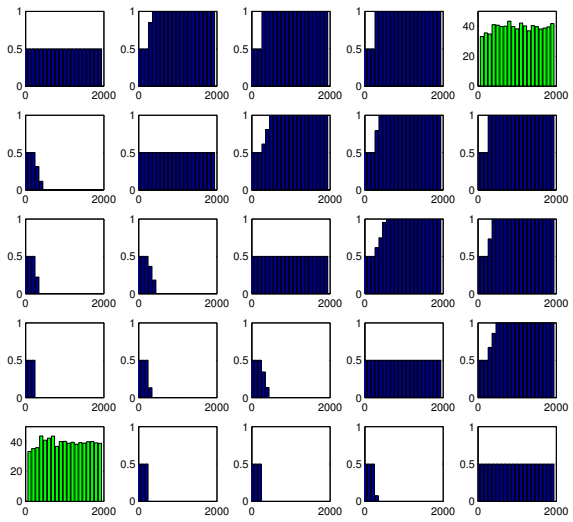


Figure 18: Traffic distribution and aggregate load at each GW as a function of time (subplot ij corresponds to node n_{ij}).

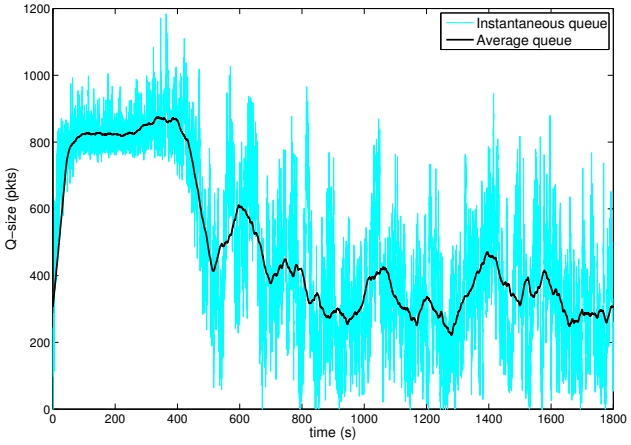


Figure 19: Total queue size evolution.

at one hop distance converge in one step, while nodes at two hop distance take two iterations to converge and finally nodes at three hop distance take three iterations to reach convergence. In Fig. 19 we show the evolution of the total average queue, where we can appreciate its steep descent when MQLLB starts the operation.

7. Conclusions and Future Work

In this paper we proposed an algorithm for dynamic multipath forwarding in a WMN. The algorithm enables load-balancing and conducts the network to operate at the minimum average congestion. The proposed framework also allows to solve the gateway selection problem in a WMN. This was achieved learning the average queue length function from measurements for each link and applying an optimization method in order to reach the minimum average queue length in the network. The proper evolution and convergence of the proposed method was verified by our packet-level simulations over several canonical topologies which served as a proof of concept.

We further analyzed the simulations taking several flow-level performance metrics as average delay and jitter for UDP traffic and average goodput for TCP traffic. With this metrics we studied the performance of the proposed MQLLB method compared with the IEEE 802.11s standard. The results show a clear advantage of MQLLB against a dynamic metric routing method like the one used by 802.11s. In all the simulations, independently of the topology size, we observed a quick adaptation of MQLLB to traffic changes and also an stable operation, avoiding the routing oscillations of 802.11s, already noticed before by [11, 36].

In our future work we will perform the learning phase with real data which includes among other issues the non-zero channel error rate, typical of a real-world wireless link. All the simulations presented in this paper are done with synthetic traffic, so we would like to extend our work using real traffic data. It would also be very interesting to perform a statistical analysis of the behaviour of the mean queue size with respect to the load. A possible analysis would be to study how often does the regression function change over time (i.e. answer the question of whether the mean queue size function changes over time, and how often it does).

Another aspect of our future work is the implementation of the proposed framework in a real-world network which was briefly discussed in this article. One possible way is to explore the adaptation of a recent MPLS-based routing scheme for WMNs [24] to our proposal. A testbed deployment would be useful for enhancing the algorithm and detecting real-world driven problems that need to be solved. An interesting point which could be more profoundly studied in the future is the optimization phase. This problem could be solved by several different methods and was not analyzed in the present paper. Finally, we would like to extend the proposed framework, which was developed for a link disjoint WMN, to scenarios that have not only point to point links but also point to multipoint links.

Acknowledgments

This work was partially supported by Centro Ceibal, CSIC (I+D project “Algoritmos de control de acceso al medio en redes inalámbricas”) and ANII (grant POS_2011_1_3525). We also thank our three anonymous reviewers for their feedback which helped us to improve the quality of the paper.

References

[1] I. F. Akyildiz, X. Wang, W. Wang, Wireless mesh networks: a survey, Comput. Netw. ISDN Syst. 47 (2005) 445–487. doi:10.1016/j.comnet.2004.12.001.

[2] Wireless Communities (2012). URL `https://personaltelco.net/wiki/WirelessCommunities`

[3] F. J. Simó Reigadas, P. Osuna García, D. Espinoza, L. Camacho, R. Quispe, Application of IEEE 802.11 technology for health isolated rural environments, in: WCIT 2006, Santiago de Chile, 2006.

[4] B. Raman, K. Chebrolu, Experiences in using wifi for rural internet in india, IEEE Commun. Mag. 45 (1) (2007) 104 –110. doi:10.1109/MCOM.2007.284545.

[5] Plan Ceibal: One Laptop per Child implementation in Uruguay. (2012). URL http://www.ceibal.org.uy/

[6] Carrier WiFi equipment (2012). URL http://www.infonetics.com/pr/2012/Carrier-WiFi-Equipment-Market-Highlights.asp

[7] G. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, B. Walke, IEEE 802.11s: The WLAN mesh standard, IEEE Wireless Commun. 17 (1) (2010) 104–111. doi:10.1109/MWC.2010.5416357.

[8] R. Draves, J. Padhye, B. Zill, Comparison of routing metrics for static multi-hop wireless networks, in: ACM SIGCOMM, 2004.

[9] A. Khanna, J. Zinky, The revised ARPANET routing metric, SIGCOMM Comput. Commun. Rev. 19 (1989) 45–56. doi:http://doi.acm.org/10.1145/75247.75252.

[10] K. Ramachandran, I. Sheriff, E. Belding, K. Almeroth, Routing stability in static wireless mesh networks, in: PAM, 2007, pp. 73–83.

[11] R. G. Garroppo, S. Giordano, L. Tavanti, A joint experimental and simulation study of the IEEE 802.11s HWMP protocol and airtime link metric, Int. J. Commun. Syst. 25 (2) (2012) 92–110. doi:10.1002/dac.1255.

[12] M. Caesar, M. Casado, T. Koponen, J. Rexford, S. Shenker, Dynamic route recomputation considered harmful, SIGCOMM Comput. Commun. Rev. 40 (2010) 66–71. doi:http://doi.acm.org/10.1145/1764873.1764885.

[13] P. P. Pham, S. Perreau, Increasing the network performance using multi-path routing mechanism with load balance, Ad Hoc Networks 2 (4) (2004) 433 – 459. doi:10.1016/j.adhoc.2003.09.003.

[14] F. Larroca, J.-L. Rougier, Minimum delay load-balancing via nonparametric regression and no-regret algorithms, Computer Networks 56 (4) (2012) 1152 – 1166. doi:10.1016/j.comnet.2011.11.015.

[15] B. Raman, K. Chebrolu, D. Gokhale, S. Sen, On the feasibility of the link abstraction in wireless mesh networks, Networking, IEEE/ACM Transactions on 17 (2) (2009) 528 –541. doi:10.1109/TNET.2009.2013706.

[16] R. Draves, J. Padhye, B. Zill, Routing in multi-radio, multi-hop wireless mesh networks, in: ACM MobiCom, 2004, pp. 114–128.

[17] A. Raniwala, T. Chiueh, Architecture and algorithms for an ieee 802.11-based multi-channel wireless mesh network, in: IEEE INFOCOM, Vol. 3, 2005, pp. 2223 – 2234 vol. 3. doi:10.1109/INFCOM.2005.1498497.

[18] V. Mhatre, H. Lundgren, F. Baccelli, C. Diot, Joint mac-aware routing and load balancing in mesh networks, in: ACM CoNEXT, 2007, pp. 19:1–19:12. doi:http://doi.acm.org/10.1145/1364654.1364679.

[19] Y. Bejerano, S.-J. Han, A. Kumar, Efficient load-balancing routing for wireless mesh networks, Comput. Netw. 51 (2007) 2450–2466. doi:10.1016/j.comnet.2006.09.018.

[20] E. Alotaibi, V. Ramamurthi, M. Batayneh, B. Mukherjee, Interference-aware routing for multi-hop wireless mesh networks, Comput. Commun. 33 (2010) 1961–1971. doi:http://dx.doi.org/10.1016/j.comcom.2010.06.025.

[21] J. Zhou, K. Mitchell, A scalable delay based analytical framework for CSMA/CA wireless mesh networks, Comput. Netw. 54 (2010) 304–318. doi:http://dx.doi.org/10.1016/j.comnet.2009.05.013.

[22] E. Ancillotti, R. Bruno, M. Conti, A. Pinizzotto, Load-aware routing in mesh networks: Models, algorithms and experimentation, Comput Commun. 34 (8) (2011) 948 – 961. doi:10.1016/j.comcom.2010.03.004.

[23] J. J. Gálvez, P. M. Ruiz, A. F. Gómez-Skarmeta, Responsive on-line gateway load-balancing for wireless mesh networks, Ad Hoc Networks 10 (1) (2012) 46–61.

[24] S. Avallone, G. Di Stasi, A new MPLS-based forwarding paradigm for multi-radio wireless mesh networks, Wireless Communications, IEEE Transactions on 12 (8) (2013) 3968–3979. doi:10.1109/TWC.2013.071113.121529.

[25] P. Casas, F. Larroca, J.-L. Rougier, S. Vaton, Taming traffic dynamics: Analysis and improvements, Comput. Commun. 35 (5) (2012) 565–578. doi:10.1016/j.comcom.2010.07.009. URL http://dx.doi.org/10.1016/j.comcom.2010.07.009

[26] G. Bianchi, Performance analysis of the IEEE 802.11 distributed coordination function, IEEE J. Sel. Areas Commun. 18 (3) (2000) 535–547. doi:10.1109/49.840210.

[27] T. Kuosmanen, Representation theorem for convex nonparametric least squares, Econometrics Journal 11 (2) (2008) 308–325.

[28] The MOSEK optimization software. URL http://www.mosek.com/

[29] A. Jindal, K. Psounis, The achievable rate region of 802.11-scheduled multihop networks, IEEE/ACM Trans. Netw. 17 (4) (2009) 1118–1131. doi:10.1109/TNET.2008.2007844.

[30] D. J. Leith, V. G. Subramanian, K. R. Duffy, Log-convexity of rate region in 802.11e wlans, Comm. Letters. 14 (1) (2010) 57–59. doi:10.1109/LCOMM.2010.01.091154.

[31] D. J. Leith, Q. Cao, V. G. Subramanian, Max-min fairness in 802.11 mesh networks, IEEE/ACM Trans. Netw. 20 (3) (2012) 756–769.

[32] T. Salonidis, G. Sotiropoulos, R. Guerin, R. Govindan, Online optimization of 802.11 mesh networks, in: Proceedings of the 5th international conference on Emerging networking experiments and technologies, CoNEXT '09, 2009, pp. 61–72. doi:10.1145/1658939.1658947.

[33] The ns-3 Network Simulator. URL http://www.nsnam.org/releases/

[34] F. Li, M. Li, R. Lu, H. Wu, M. Claypool, R. E. Kinicki, Measuring queue capacities of ieee 802.11 wireless access points, in: BROADNETS, 2007, pp. 846–853.

[35] G. Carneiro, P. Fortuna, M. Ricardo, Flowmonitor: a network monitoring framework for the network simulator 3 (ns-3), in: Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS '09, 2009, pp. 1:1–1:10. doi:10.4108/ICST.VALUETOOLS2009.7493.

[36] R. M. Abid, T. Benbrahim, S. Biaz, Ieee 802.11s wireless mesh networks for last-mile internet access: An open-source real-world indoor testbed implementation, Wireless Sensor Network 2 (10) (2010) 725–738.