A Feedback-based Adaptive Online Algorithm for Multi-Gateway Load-Balancing in Wireless Mesh Networks

Juan J. Gálvez
DIIC, Computer Science Faculty
University of Murcia, Spain
Email: jjgalvez@um.es

Pedro M. Ruiz
DIIC, Computer Science Faculty
University of Murcia, Spain
Email: pedrom@um.es

Antonio F. G. Skarmeta
DIIC, Computer Science Faculty
University of Murcia, Spain
Email: skarmeta@um.es

Abstract—We propose an adaptive online load-balancing protocol for Multi-Gateway Wireless Mesh Networks (WMNs) which, based on the current network conditions, balances load between gateways. Our protocol (GWLB) achieves two goals: (i) alleviating congestion in affected domains and (ii) balancing load to improve flow fairness across domains. As a result of applying GWLB the throughput and fairness of flows improves. The proposed scheme effectively takes into account the elastic nature of TCP traffic, and intra-flow and inter-flow interference when switching flows between domains. Through simulations, we analyze performance and compare with a number of proposed strategies, showing that GWLB outperforms them. In particular, we have observed average flow throughput gains of 104% over the nearest gateway strategy.

I. INTRODUCTION AND MOTIVATION

Wireless Mesh Networks (WMNs) have recently attracted much attention. They are comprised of mesh routers and mesh clients. Mesh routers are generally static (or quasi-static) in nature and are interconnected by wireless links. They serve as an infrastructure wireless backbone, providing connectivity to mesh clients. Typically, a subset of routers have direct connectivity to a fixed infrastructure (e.g. a wired network such as the Internet) and serve as *gateways* to the mesh nodes. WMNs provide a cost-effective way to deploy a wide-area network and offer services such as Internet connectivity.

Gateway nodes are a key component of WMNs. In many applications of WMNs most traffic will be directed to/from gateways. Thus, traffic aggregation occurs in the paths leading to a gateway, which can lead to congestion. When the network uses multiple gateways, one important consideration is the strategy employed to associate nodes with a particular gateway, i.e. through which gateway does a node send/receive traffic? We refer to the set of nodes served by a gateway as its *domain*.

Past work has focused mainly on finding high throughput paths in wireless multi-hop networks. To this end, various routing metrics such as ETX [1], ETT [2] and MIC [3] have been proposed. Associating nodes to the nearest gateway in terms of a routing metric (hop-count or any of the above) can lead to load imbalance. Because these metrics are load-agnostic, shortest path routing can lead to situations where gateways are subject to different load, and in a worst case where a few gateways are overloaded and others are under-utilized. Several factors can lead to this situation. For example, heterogeneous traffic demands, uneven number of nodes served by gateways (motivated for example by unplanned gateway placement, or user behavior), or gateway failure. Load imbalance can lead to inefficient use of network capacity, throughput degradation and unfairness between flows in different domains.

Because the traffic volume in a WMN is expected to be very high, performing load-balancing in a multi-gateway WMN is important. Adding to the complexity of the problem are several facts: (a) the capacity and achievable throughput of a multi-hop wireless network and its domains is very difficult to predict and varies over time; (b) the majority of Internet traffic today is TCP, meaning that a solution must take into account the elastic nature of flows. We cannot assume a specific demand for each flow. In fact, due to limited capacity of WMNs, flows will most likely use all the available bandwidth; (c) in a wireless multi-hop network, interference (namely inter-flow and intra-flow interference) has a major role in network performance, which complicates the design of a load balancing strategy. Association of nodes to gateways must be chosen carefully, as it is important to maintain traffic locality and avoid flow interactions.

We address the above problems by designing a self-corrective online adaptive algorithm called Gateway Load-Balancer (GWLB) which, given the current network conditions and using a number of heuristics, calculates an association of nodes to gateways. This solution has two objectives: (a) to alleviate congestion by re-routing traffic from congested to uncongested domains; (b) to improve inter-domain flow fairness by balancing load between domains when beneficial. Both goals are necessary and complement each other. The algorithm executes periodically, adapting to network conditions. It takes into account the elastic nature of TCP flows, as well as flow interactions when switching nodes between domains, in order to prevent severe interference. As a result of applying GWLB, flow throughput and fairness is improved.

The rest of the paper is organized as follows. In section II we review related work. In section III we describe the network model, formulate the problem, and explain the challenges and nature of the proposed solution. Section IV explains our gateway load-balancing protocol. Section V shows and

analyzes protocol performance based on simulations with *ns-2*. Finally, in Section VI we summarize our conclusions.

II. RELATED WORK

There are a number of works concerning the problem of load-balancing in single or multi-gateway WMNs.

Bejerano et al. [4] study the problem of load-balanced routing in multi-gateway mesh networks. The algorithm is executed in a centralized point outside the mesh network, and employs models which do not apply to our study: a interference-free graph-theoretic model, where each wireless link has a known capacity. Several studies use balanced trees rooted at the gateways and route traffic along the tree paths. Hsiao et al. [5] propose a distributed algorithm to find a fully top load-balanced tree, using a interference-free model (they do not consider multiple gateways). Raniwala et al. propose the *Hyacinth* architecture for multi-radio multi-channel WMNs in [6]. Routing and channel assignment is done distributively and dynamically. The performance results are mainly concerned with the increase in throughput due to the use of multiple channels and different channel assignment strategies. Mhatre et al. [7] seek a delay-optimal routing forest, where a tree is rooted at each gateway. The cost function, however, is not load-dependent.

For the research most directly related to our problem, we broadly classify existing solutions in three groups: (i) those which seek to even the load of gateways; (ii) those which seek to alleviate congestion in a gateway by re-routing nodes of the affected domain to another gateway; (iii) those in which nodes utilize multiple gateways simultaneously to send and/or receive traffic.

Examples of solutions in the first group can be found in [8]–[10]. These solutions assume a specific demand for each flow and don't take into account the elastic nature of TCP. The main difference between them is choosing which nodes to switch between gateways. Many don't take contention into account and can perform poorly in practice. Also, in this paper we contend that a solution which merely seeks to even the load of gateways is not always beneficial, e.g. it can result in congested domains due to heterogeneous and varying domain capacity.

Proposals in the second group include the protocols WCETT-LB [11] and AODV-ST [12], which extend existing routing solutions to allow nodes to switch to alternate gateways when their default gateway is congested. These can suffer from route flapping. Another example is the work by Nandiraju et al. [13], in which congested gateways notify nodes which should switch to another domain. This solution doesn't take into account the effects of interference when switching nodes, and can suffer from route flapping. Maurina et al. [14] attempt to improve the proposal in [13] by more consciously choosing nodes to switch between domains (i.e. nodes with greater traffic and farther from the gateway). One drawback of these proposals is that they don't balance load in the absence of congestion. With TCP, domains may not be congested but flow unfairness between domains can occur.

Solutions in the third group appear in [8], and the work in [15], [16]. For these solutions we argue that, by breaking the locality of traffic in the network, they greatly increase contention and can perform worse than single nearest gateway association. For example, in [17] we showed how this occurs with the FP protocol from [8], in which nodes send to all gateways in proportion to gateway capacity.

In this paper we propose a solution which attempts to achieve the benefits of both the first two groups, while effectively taking into account interference in the network, and compatible with TCP traffic.

III. THE GATEWAY LOAD-BALANCING PROBLEM

A. Network and Traffic Model

We consider WMNs which comprise of wireless static or quasi-static mesh *routers*, also called *nodes*. These nodes form a wireless multi-hop network. Mesh *clients*, also called *users*, connect to the mesh routers. A subset of nodes, referred to as *gateways*, are directly connected to a fixed infrastructure, which we will assume is the Internet for the rest of this paper. Each router has one radio interface (e.g. 802.11b or g) for communication with other routers, using a single common channel. Communication between nodes and users is done via a separate interface and channel (wired or wireless).

Although intra-WMN communication is possible, we assume that most of the traffic will be received from the Internet. Users are randomly located in the network. A user accesses the Internet through one or more links leading from its router to a gateway. To model Internet traffic realistically, we assume that all traffic entering the WMN is elastic, and that in any instant a user can initiate a connection to the Internet generating a download flow of any number of bytes. This is safe to assume, because the majority of Internet traffic today uses TCP.

B. Problem statement

Let $\mathbb{G}$ be the set of gateways in the WMN. A *sink* is a wireless mesh router that receives Internet traffic. $\mathbb{S}$ is the set of sinks in the network. Traffic directed to a sink will be served by gateway nodes; the traffic is routed from the gateway to the sink using a minimum cost path (in terms of the routing metric used). Routes *inside* the WMN are given by the routing protocol, and we don't impose any particular protocol. Users initiate connections to the Internet, which generate download flows. The gateway load-balancing problem requires choosing the serving gateway and rate allocation for every download flow. The problem concerns only download traffic, because it accounts for most of the network load.

Note that the *Nearest Gateway* (NGW) solution, which assigns each sink to its nearest gateway in terms of the routing metric, can easily lead to load imbalance. As an example, consider the network in Fig. 1, which shows a possible NGW solution using hop-count. Domain B has more flows and this would suggest that the load is greater and the bandwidth share of each flow is less than in domain A.

The gateway load-balancing problem **P** can be formulated as an optimization problem. Let $\mathbb{F}$ be the set of download
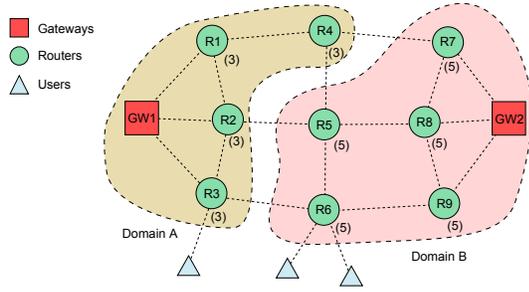
Fig. 1. Example WMN. There are two gateways GW_1 and GW_2, with their corresponding domains A and B. The number of flows directed to each router is shown in parentheses.

flows, and r_i the rate of flow f_i. Let $f = |\mathbb{F}|$ and $g = |\mathbb{G}|$. A reasonable objective is proportional fairness of flows, which can be achieved by maximizing the aggregate utility of flows, when the utility of a flow is given by $\ln(r_i)$ [18].

A flow can be routed through $|\mathbb{G}|$ different paths, corresponding to the paths from each gateway to the sink. A solution of $\mathbf{P}$ must choose the serving gateway of each flow or, in other words, the route the flow uses. To represent this, we define g flows (called *routed flows*) for every original flow, each served by a different gateway. The number of routed flows is thus $f \times g$. Let t_{ij} be the routed flow of flow i assigned to gateway j, with rate x_{ij}. We assign a binary integer variable b_{ij} to each routed flow, indicating whether it is active or not. Only one routed flow of a flow can be active: $\sum_{j=0}^{g} b_{ij} = 1, \forall i \in f$.

We thus have $r_i = \sum_{j=0}^{g} b_{ij} x_{ij}$. The network model determines the remaining problem constraints. A model which has been frequently used defines contention regions of the network as maximal cliques of the contention graph [19]. In the contention graph, each vertex is a link of the network graph, and there is a link between vertices when they contend. Interference is determined by the Protocol Model [20].

A solution to $\mathbf{P}$ is given by the vector $\boldsymbol{x} = (b_{ij} x_{ij})$, $\forall i \in f, \forall j \in g$. The clique-flow matrix [19] $\mathbf{R} = \{R_{qt}\}$ represents the number of links the routed flow t is using in clique q. Let the vector $\mathbf{C} = (C_q)$ be the vector of achievable channel capacities in each of the cliques. Given the above considerations, the problem can be formulated as:

$$\mathbf{P} : maximize \quad \sum_{i=0}^{f} \ln(\sum_{j=0}^{g} b_{ij} x_{ij}) \quad (1)$$

$$subject \ to$$

$$\mathbf{R}\boldsymbol{x} \leq \mathbf{C} \quad (2)$$

$$\sum_{j=0}^{g} b_{ij} = 1 \quad \forall i \in f \quad (3)$$

This is a Mixed-Integer Nonlinear Programming (MINLP) problem, which is NP-hard. There are important drawbacks to solving the problem via this approach:

- The optimization problem is NP-hard. Recalculating the solution to adapt to dynamic traffic is intractable.
- Determining capacity of each clique isn't trivial. It depends on a number of factors, such as MAC inefficiencies.
- Protocol model of interference cannot be considered realistic.
- Controlling the rate of flows adds complexity.

C. Alternative problem formulation

Due to the drawbacks of obtaining an optimal solution, in this paper we propose an online approach which, based on measurable network conditions and a number of heuristics, dynamically calculates an association of sinks to gateways, such that the total network throughput and inter-domain flow fairness increase with respect to the NGW solution. Rate control is performed by TCP.

A solution to the problem is a partition $\mathbb{P}$ of $\mathbb{S}$ formed by $|\mathbb{G}|$ blocks, called *domains*, each containing the set of sinks served by a gateway. In the example of Fig. 1, $\mathbb{P} = \{A, B\}$.

Because users are randomly located and can generate connections at any time, network conditions can constantly change. It is therefore important that the algorithm adapt to the current conditions, and the solution must converge, while avoiding route flapping.

In addition, the absence of a model which can accurately capture the network characteristics makes it very difficult to predict the capacity and achievable throughput of the network and its domains. The interference factor alone plays a major role in multi-hop wireless networks. The capacity of a chain of nodes and the locality of traffic are key factors influencing network capacity [20], [21]. This complicates the decision process, because the actual results of applying a solution are not known a priori.

D. Proposed solution

Due to the constraints mentioned above, the proposed online load-balancing algorithm works as follows:

It is a feedback based adaptive algorithm which continually monitors network conditions, and based on the current conditions, knowledge of the network topology, and a number of heuristics, calculates a solution $\mathbb{P}$. The algorithm is self-corrective and executes periodically to adapt to current conditions. GWLB monitors:

- Congestion by reading the state of wireless interface queues in selected nodes
- Number of active flows directed to each sink

The main idea of the algorithm consists in switching sinks from congested to uncongested domains, and in attempting to balance the number of flows between domains to increase inter-domain flow fairness, when not considered detrimental.

We will now explain the heuristics and mechanisms GWLB uses to achieve these goals. First we give some definitions:

Definition 1. A sink's *native domain* is the domain of its nearest gateway, according to the routing metric used.

Definition 2. A *foreign sink* is a sink whose native domain is different from its current domain.

Definition 3. A *foreign domain* is used to refer to the native domain of a foreign sink.

The most important rule for GWLB is taking contention into account. NGW is the base solution from which to build upon because, from the point of view of maintaining traffic locality and shorter paths (and thus reducing contention) it is best. Because contention plays a major role, load cannot be balanced arbitrarily between gateways. If flows are routed along very long paths, or interference between flows is not taken into account, throughput can be much lower than expected. For example, in Fig. 1 it is obvious that simultaneously assigning R8 to GW1 and R5 to GW2 is not a good decision, because it will increase contention in most of the network.

The algorithm thus starts with the NGW solution. When switching a node from domain A to B, the path to the new gateway must not severely interfere with nodes of domains other than domain B (including nodes of A). How interference is estimated will be explained later. When taking corrective measures, GWLB will progressively revert to NGW. For example, if a domain is congested and has foreign sinks, the algorithm will start sending foreign sinks one by one to their native domains until congestion ends. Also, if all domains are congested, the algorithm will default to NGW, because less contention leads to less congestion.

When choosing how many flows to switch between domains, the heuristic by default considers all domains of similar capacity and seeks to balance the number of flows between domains, if permitted by all other algorithm restrictions. By balancing the number of flows, inter-domain flow fairness improves. However, because domain capacity can be heterogeneous, if a domain is known to be congested, the algorithm will always attempt to switch flows to uncongested domains regardless of flow balance.

Because the results of applying a solution are not known a priori, the algorithm has to be self-corrective. If a domain is congested and has foreign sinks, this can be due to a previous wrong decision of switching too many sinks from the foreign domains to this domain. To correct this, small adjustments are done incrementally, by switching the foreign sinks one by one back to their native domain, until congestion disappears. Also, to prevent oscillations, if a domain becomes congested and has foreign sinks, switching from the foreign domains to the congested domain is temporarily suspended. This prevents the foreign domain from immediately sending sinks again to the congested domain if it becomes uncongested.

Finally, in order to simplify the protocol, the dynamic association of sinks to gateways is done only for the purpose of receiving traffic from the Internet (i.e. download traffic). Traffic sent by nodes to the Internet will always be routed to the nearest gateway. This way, all knowledge of sink-gateway association and decisions can be kept at the gateways.

TABLE I
GWLB PARAMETERS.

G_T	GWLB execution period (s)
k_θ	Read the state of wireless interface queues of k-hop neighbors of gateways for $k \leq k_\theta$. In this paper $k_\theta = 0$
Q_T	Interface queue sample period (s)
C_θ	Congestion threshold (% of queue size)
S_t	Domain switching suspension time (in GWLB cycles)
P_θ	Path distance candidate sink threshold

TABLE II
GWLB SYMBOLS.

$\mathbb{G}$	Set of gateways in the network
$\mathbb{S}$	Set of sinks in the network
$\mathbb{P}$	Sink partition of $|\mathbb{G}|$ blocks calculated by GWLB
s_f	Number of active download flows of sink s
A_f	Number of active download flows of domain A: $\sum_{s \in A} s_f$
GW_A	Gateway node of domain A
$P_{m \to n}$	Path used by the routing protocol to connect nodes m and n

IV. GATEWAY LOAD-BALANCING PROTOCOL

In this section we explain in detail the GWLB protocol and algorithm. The GWLB algorithm executes periodically in one of the gateways, called the controller. The controller can be any one of the gateways and in case of failure a new controller is elected from the remaining gateways. A simple election protocol can be used, e.g. choose the gateway with lowest ID (e.g. IP address). Based on the network state, GWLB calculates a sink partition $\mathbb{P}$, formed by $|\mathbb{G}|$ domains. Every time $\mathbb{P}$ is calculated, the controller informs all gateways in order to enforce the solution. The controller informs only of differences in $\mathbb{P}$ since the last execution of GWLB. Gateways communicate through the wired network.

Download traffic is routed based on $\mathbb{P}$, i.e. a gateway is responsible for injecting the traffic of sinks in its domain inside the WMN:

- When a gateway receives traffic not destined to a sink in its domain, the gateway forwards it to the gateway calculated in $\mathbb{P}$ (directly or via a tunnel through the wired network if the gateways are in different sub-networks).
- If the traffic is directed to a sink in its domain it routes it through the WMN. We do not impose any particular routing protocol.

Table I lists the parameters used by GWLB. Table II lists the symbols used to explain the algorithm. These are used and explained throughout the text.

A. Protocol initialization

To react to the state of gateways in the network, the following procedure is executed at the controller the first time it is elected, and every time a new gateway appears or disappears:

1) Calculate shortest paths from all nodes to gateways (only to gateways for which controller doesn't have this information)
2) $\mathbb{P} \leftarrow$ NGW

3) Calculate and store candidate sinks to switch from A to B, $\forall (A, B) \in \mathbb{P} \times \mathbb{P}$.
4) Inform gateways of current partition $\mathbb{P}$

Every time a change occurs in the number of gateways, the controller calculates the NGW solution and informs all gateways. Step 3 calculates the set of sinks which can be switched from domain A to B without generating severe interference in domains other than B. This step is explained later (see IV-C). GWLB uses this information during execution.

If the controller shuts down or fails, a new one is elected. Note that this protocol adapts to the gateways present in the network at any time. If for example a gateway fails, flows of its sinks will be re-routed through the remaining gateways.

B. Network state monitoring

To calculate a solution, every cycle GWLB obtains the following information from all gateways:

- Domain congestion, which is the maximum median queue length of a subset N of nodes in the domain: $\max_{n \in N}\{median(qlen\_samples_n)\}$ where $qlen\_samples_n$ are the queue length samples of node n obtained during the last cycle. The median is a robust estimator and provides a more representative measure of the "typical" length of the queue during the last cycle, due to the bursty nature of TCP traffic.
- Number of active download flows of sinks in the domain.

The information refers to the last cycle, and is obtained by the controller from every gateway before executing GWLB. A gateway knows the number of download flows of each sink because traffic passes through the gateway. For calculating congestion, the queue state will be read from the gateway, and from the k-hop neighbors of the gateway, where $k \le k_\theta$ (k_θ can be zero). Queue length is sampled every Q_T seconds. Because routing paths from the gateway to nodes form a tree structure where the root is the gateway, congestion will most likely occur in the gateway or children near the gateway. For this paper, and due to space restrictions, we will be assuming $k_\theta = 0$. To execute GWLB, the controller must also know the complete network topology. In a WMN nodes are typically static, so this knowledge can be obtained easily by the gateways, and may be already provided by the routing protocol (e.g. OLSR).

C. Avoiding interference

GWLB will only attempt to switch sinks from a domain A to B if they satisfy the following requirement: that the path to the new domain doesn't severely interfere with domains other than B. Because the path leads to the gateway in domain B, ideally it should only interfere with nodes in B. In this subsection we explain how this is calculated. First we give some definitions:

- A path p is a sequence of unique nodes such that there exists an edge in the graph connecting each node to the next node in the sequence.
- $|p|$ is the number of nodes in path p.

- $\delta(n, m)$ is the distance between nodes n and m, defined as the number of edges in a shortest path connecting them.

The distance of a node n to a set of domains $\mathbb{D}$ is the minimum distance from n to a node in $\mathbb{D}$, as shown in Eq. 4. The distance from a path p to a set of domains $\mathbb{D}$ is the arithmetic mean of the distance of nodes of p to $\mathbb{D}$, as per eq. 5.

$$\delta_n(n, \mathbb{D}) = \min_{m \in \bigcup \mathbb{D}}\{\delta(n, m)\} \tag{4}$$

$$\delta_p(p, \mathbb{D}) = \frac{\sum_{n \in p} \delta_n(n, \mathbb{D})}{|p|} \tag{5}$$

$$\mathrm{CS}_{A,B} = \{s \in A : \delta_p(P_{s \to GW_B}, \mathbb{P} - \{B\}) \ge P_\theta\} \tag{6}$$

The set of candidate sinks which can be switched from A to B, as shown in eq. 6, includes every sink s of A that verifies that the distance of $P_{s \to GW_B}$ to domains other than B is greater than P_θ. This forces the path from s to the new gateway to be separated from nodes not in B. In other words, the new path will not interfere with nodes of other domains. Naturally, the distance of the first nodes in the path to domains other than B will be small, and this distance will increase as the path advances to GW_B. The value of P_θ should be set according to the network topology and size of domains such that it guarantees adequate interference avoidance.

Note that distance is measured in hops, which can correlate with Euclidean distance and is suitable to measure spatial separation [22]. Increased spatial separation decreases the probability of interference.

D. GWLB: Gateway Load-Balancing algorithm

The algorithm executes every G_T seconds and consists of two phases. The first phase (shown in Algorithm 1) attempts to switch sinks from congested to uncongested domains. A domain is congested if its congestion level is above the threshold C_θ. In lines 1 and 2 the sets CD and UC are initialized, containing the congested and uncongested domains, respectively.

For each congested domain C, GWLB will attempt to alleviate congestion in one of two ways:

If the domain C has active foreign sinks, the algorithm will switch, for each foreign domain, one sink back to its native domain, and temporarily suspend switching for S_t cycles from the foreign domains to C (lines 4-10). Foreign sinks are the best candidates to switch, because they don't belong to this domain as per the NGW solution, and as explained in III-D, in the face of congestion we revert to NGW. Also, switching foreign sinks may be necessary as a corrective measure. Because the capacity of domains is unknown, the foreign domain could have previously switched too many sinks to C when it was uncongested. It is also for this reason that, to prevent oscillations, we temporarily suspend switching from

Algorithm 1 GWLB algorithm 1st phase: switch flows from congested to uncongested domains.

```
 1: CD := {D ∈ ℙ : D.congestion ≥ C_θ}
 2: UD := {D ∈ ℙ : D.congestion < C_θ}
 3: for C ∈ CD do // from most congested to least
 4:     if C has active foreign sinks then
 5:         for {F ∈ ℙ : F ≠ C} do
 6:             if C has active foreign sinks from F then
 7:                 s := foreign sink from F farthest from GW_C
 8:                 C := C − {s}
 9:                 F := F ∪ {s}
10:                 Suspend switching from F to C for S_t cycles
11:     else
12:         for U ∈ UD do // from least congested to most
13:             if switching from C to U not suspended then
14:                 validCandidateSinks := list({s ∈ CS_{C,U} :
                        s_f > 0 & s.currentDomain = C})
15:                 if |validCandidateSinks| > 0 then
16:                     flowsToSwitch := max(1, C_f − φ)
17:                     i := 0
18:                     for s ∈ validCandidateSinks do
19:                         C := C − {s}
20:                         U := U ∪ {s}
21:                         i := i + s_f
22:                         if i ≥ flowsToSwitch then
23:                             break
24:                     if i ≥ flowsToSwitch then
25:                         break
```

the foreign domain. That is, to prevent the foreign domain from switching sinks back to C if it becomes uncongested.

If C has no foreign sinks, it will attempt to switch sinks to uncongested domains U, starting with the least congested. The list of valid candidate sinks to switch is first obtained (line 14). This includes all sinks in $CS_{C,U}$ which have active flows and are currently in domain C. Given this list, C will attempt to switch the following number of flows:

$$\max(1, C_f - \phi) \tag{7}$$

The value ϕ represents the desired number of flows that will remain in C after the switch. In this paper $\phi = 0.6(C_f + U_f)$. Because this value is slightly above the average number of flows of C and U, our goal is to conservatively balance the number of flows between both domains leaving more flows in C than in U. Also note that the following cases can occur: $U_f \geq C_f$ or $U_f \approx C_f$. In these cases, one flow is switched. This is also a conservative measure. Because U_f is greater or similar to C_f, and although the domain is uncongested, the idea is to progressively switch one flow every cycle to avoid overloading U.

The second phase tries to balance the number of flows between the uncongested domains, in an attempt to increase inter-domain flow fairness. This is to prevent cases where, for example, an uncongested domain has many flows and another domain has very few flows. The flows of the last domain will

likely benefit from higher throughput. GWLB will attempt to switch sinks from domains A with an above average number of flows to domains B with a below average number of flows. If switching from $a \in A$ to $b \in B$ is not suspended, the algorithm progressively switches sinks from the valid list of candidate sinks to switch from a to b (placing any foreign sinks belonging to b in front of this list) if the switch decreases the variance of the number of flows between domains.

V. PERFORMANCE EVALUATION

We evaluate the performance of GWLB in WMN scenarios with multiple gateways and a varying number of users in each n-domain[1]. We simulate GWLB with *ns-2* [23] and compare it against three different solutions: NGW, MLI [8] and MAU [14]. To our knowledge, MLI and MAU represent the best performing solutions of two different approaches used in the literature to perform gateway load-balancing (summarized in section II). MLI seeks to even the load of gateways and MAU seeks to alleviate congestion by switching sinks from congested to uncongested gateways. We implemented an approximation of MLI in [17], and found it to be the best performing of the protocols in [8]. MLI indirectly takes contention into account, which enables it to perform better than solutions with similar goals. In a similar manner, MAU has been selected because it offers the least number of disadvantages compared to similar solutions.

A. Simulation environment

We have implemented the protocols in *ns-2* in the following manner. Wired links are 100 Mbps and wireless links are 11 Mbps. There is a server $\mathcal{A}$ outside the WMN to which users connect, and sends data to them. $\mathcal{A}$ is connected by a wired link to another machine $\mathcal{B}$, which implements and runs the different solutions. $\mathcal{B}$ is connected by wired links to every WMN gateway. When $\mathcal{B}$ receives traffic directed to the WMN, it forwards it to the appropriate gateway based on the current partition $\mathbb{P}$ (which is algorithm-dependent). $\mathcal{B}$ knows the complete topology, the number of active flows directed to each sink, and can read the state of gateway queues. Routing inside the WMN is static shortest-path based on hop-count.

The WMN topologies consist of static networks of 100 nodes placed randomly in a square 2000 m × 2000 m area. All topologies are connected, with a minimum distance of 160 m between nodes. There are 4 gateways, one in each corner of the square (this results in 4 domains).

Traffic is TCP. Users connect to $\mathcal{A}$, which generates download flows. For each user, the time between start of connections follows an exponential distribution with $\lambda = 1/10000$ ms, and the size of download flows follows an exponential distribution with $\lambda = 1/65000$ bytes.

Simulations last 150 sec, with users starting connections during the first 100 sec. The last 50 sec are used to give time for ongoing TCP connections to finish (if there is congestion the duration of flows can greatly increase).

---

[1]n-domains are the domains calculated by NGW. In a static topology these domains don't change.

The number of users per n-domain varies in $\{0, 5, 10, 15, 20, 25\}$. Given a topology, we generate scenarios using all combinations of users on every n-domain. Because there are four n-domains and six possible quantities of users per n-domain, the total number of different scenarios given a topology is $6^4 - 1 = 1295$ (we eliminate the case where all domains have 0 users). We have generated 5 topologies which gives a total of 6475 scenarios. For every unique scenario, users are randomly assigned to a node in their n-domain.

We have set the parameters of GWLB to $G_T = 2$, $k_\theta = 0$, $Q_T = 0.1$, $C_\theta = 70\%$, $S_t = 5$ and $P_\theta = 1.8$ (see Table I). These values provide good results under the scenarios tested, enabling adaptation and convergence to network conditions, congestion detection, and avoiding interference when switching sinks between domains. The parameters $Monitor\_Cycle$ and $Threshold$ of MAU can be considered equivalent to G_T and C_θ in GWLB. We set the same values for a fair comparison. The parameters for the MLI scheme are $\Delta_l = 1.3$ and $P_{MLI} = 0.7$, as in [8].

Confidence intervals are shown at the 95% level.

B. Performance metrics

We use the following metrics to study protocol performance. Flows refer to TCP download flows: (i) *Flow PDR* - the Packet Delivery Ratio of a flow, calculated as the ratio of received data packets to those transmitted by the source (every packet -including TCP retransmissions- contributes to the packet count); (ii) *Flow delay* - this is the average end-to-end delay of all delivered data packets of a flow; (iii) *Total network throughput* - total data bytes delivered by the network divided by the time elapsed since the first packet was sent and the last packet was received; (iv) *Flow duration* - time elapsed since the first packet was sent and the last packet was received; (v) *Flow throughput* - the number of bytes delivered divided by the duration of the flow; (vi) *Flow throughput fairness* - rates the fairness of the throughput achieved by flows in one scenario using Jain's fairness index.

For each scenario, the flow PDR, delay, duration and throughput is taken as the median value of all flows. The median is chosen due to the frequent presence of outliers and skewed distribution of the flow metrics. For example, there can be scenarios where one flow benefits from a much higher throughput than other flows.

C. Results and analysis

Fig. 2 shows results under varying number of users in the network. It is important to note that each value shown is the mean value taken from a large number of heterogeneous scenarios (with different topology, number of users, users per domain and location of users in each domain).

We can see in the figure that GWLB outperforms all protocols across all metrics. In general, MAU will perform similar or slightly better than NGW, and MLI will perform better than both when the number of users is not too high, but rapidly degrading afterward.

As the number of users increases, congestion increases, which leads to packet drops. The PDR of flows, shown in Fig. 2 (a), in general drops faster with NGW. MAU performs slightly better than NGW. The performance of MLI is good when the number of users is below 50 but starts to rapidly degrade afterward, indicating that MLI makes erroneous decisions. The PDR drops slower with GWLB, which indicates that it can effectively avoid congestion. When domains are congested, GWLB switches sinks (if possible) from congested to uncongested domains, alleviating load. When the number of users in the network is very large (more than 75) and all domains are congested GWLB defaults to the NGW solution, or a close-to-NGW solution.

There are two main reasons behind the erroneous decisions of MLI, which manifest when the number of users is high. These were explained in section II and are: it attempts to even the load but doesn't take into account that TCP flows are elastic, and it doesn't take into account heterogeneous and varying domain capacity. In general, the performance of MAU is very similar to NGW. We believe it is limited by one reason: the choice of sink to switch from a congested to uncongested domain is suboptimal. When a domain is congested, the sink which maximizes $ETX \times rate$ is chosen, where ETX is the routing metric from the sink to its current gateway. With TCP flows, and due to the nature of multi-hop wireless transmission, flows of sinks closer to the gateway will have higher throughput. As a result, MAU may not select sinks which are farthest from the gateway but rather halfway. This means that the path to the new gateway will be longer, leading to increased contention in the network.

We can observe similar behavior of all protocols with the flow delay and network throughput metrics. With a small number of users there is no congestion and so the improvement achieved by GWLB is small. As congestion builds up, the throughput of flows decreases but by balancing load, GWLB is able to improve flow throughput with respect to NGW and therefore total network throughput. Fig. 2 (d) shows the duration of flows, which follows the same pattern. Note that congestion has a considerable impact on flow duration.

In almost all scenarios, GWLB maintains a better median flow throughput. Finally, the last figure shows how it achieves better fairness between flows. The reason is that GWLB balances the number of flows between domains whenever possible, which increases inter-domain flow fairness.

In addition to the results shown, it is important to note that comparing the throughput achieved by flows in the same scenario with GWLB over NGW, we have that the improvement in flow throughput per scenario when using GWLB is on average 104%. It is also important to note that the throughput of GWLB is noticeably less than NGW (more than 5%) only in 4% of all the scenarios tested, which indicates that GWLB seldom makes erroneous decisions, specially by avoiding arbitrary load-balancing which can increase contention and therefore prove detrimental.

Fig. 3 shows results under varying user imbalance between domains. This is measured as the standard deviation of the
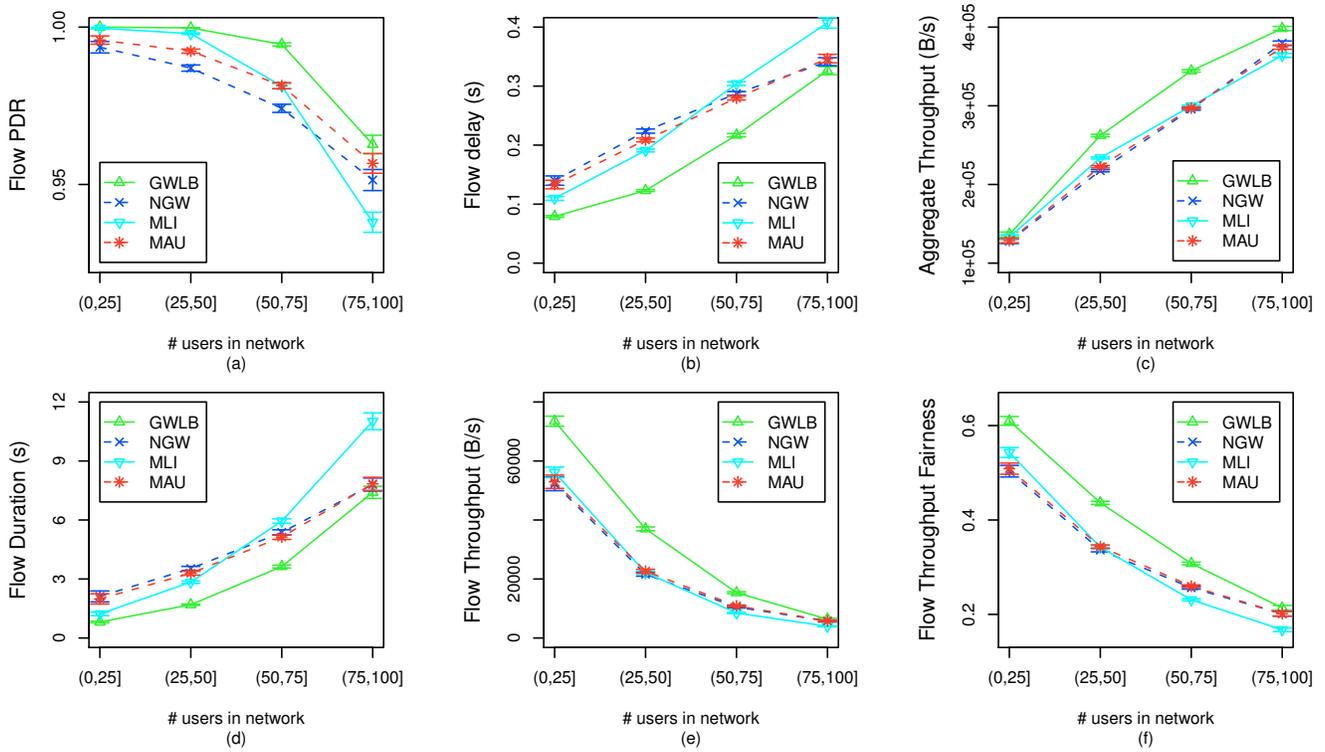
Fig. 2. Performance comparison of GWLB, NGW, MLI and MAU with varying number of users in the network.
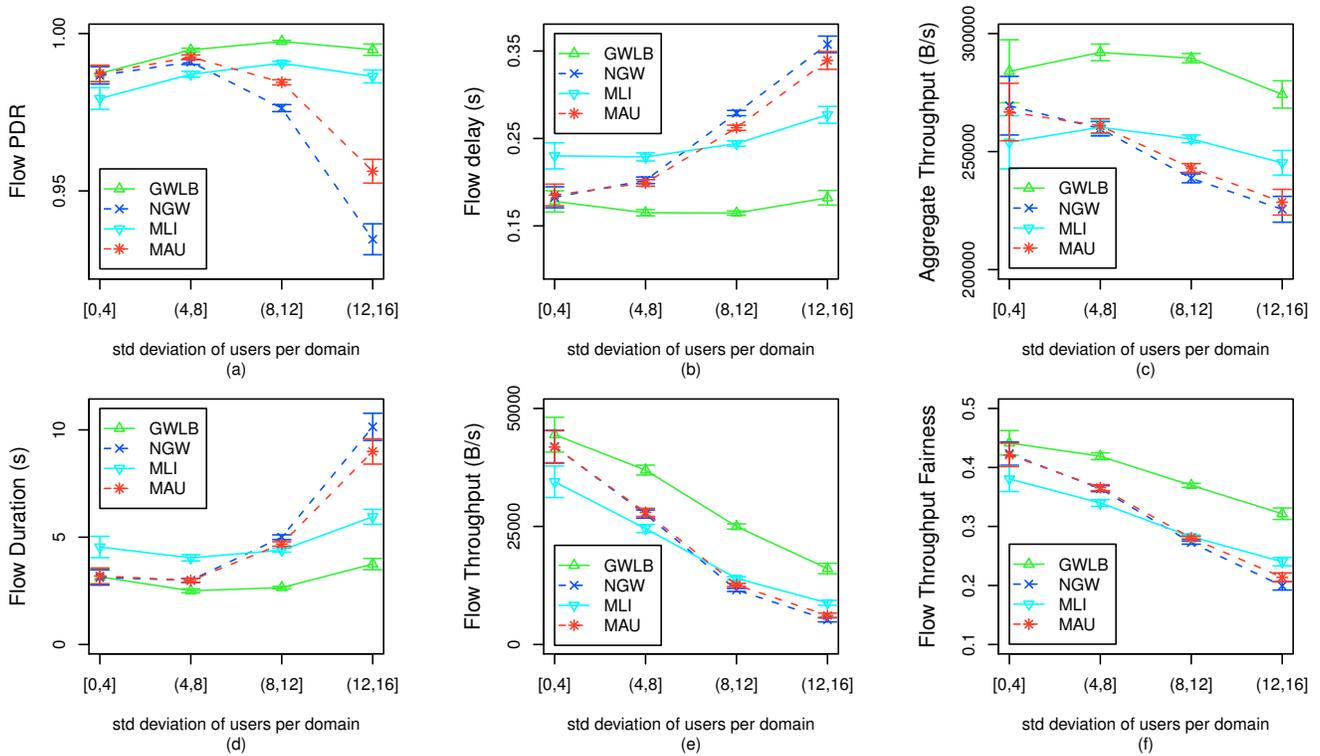

Fig. 3. Performance comparison of GWLB, NGW, MLI and MAU with varying user imbalance between n-domains.

number of users in every n-domain, which translates into load imbalance. Same as above, each value shown is the mean value taken from a large number of heterogeneous scenarios.

As in the above tests, GWLB performs best. We can observe how MLI once again makes erroneous decisions, which clearly manifest when the imbalance is low, leading to the worst performance in these cases. Only when the imbalance is high MLI starts to perform better. The performance of MAU is slightly better than NGW.

In Fig. 3 (a), we see that when imbalance increases, the PDR decreases. The reason is that increasing load imbalance progressively produces congestion in one or more domains. Note that because GWLB balances load (switching sinks from congested to uncongested domains) it is less affected by this.

With flow delay, network throughput and flow duration metrics we observe similar results. The performance of NGW is worse and decreases faster with increasing imbalance. Flow throughput is higher in GWLB as shown in Fig. 3 (e). Finally, we can see in Fig. 3 (f) that the difference in fairness between GWLB and NGW increases with load imbalance. By balancing load between domains whenever possible and by avoiding congestion, GWLB is more capable of maintaining inter-domain flow fairness.

VI. Conclusions

Load-balancing between gateways in a WMN can be of crucial importance. Imbalance between domains can easily occur due to a number of reasons, including unplanned gateway placement or heterogeneous traffic demands. Moreover, the roaming of end-systems across the network together with variable radio conditions contribute to make these demands more dynamic over time. In addition, the limited wireless link capacity aggravates the problem, turning gateways into bottlenecks, which can easily lead to congestion.

In this paper we have proposed GWLB, an online protocol which dynamically adapts to network conditions, balancing the load of gateways. It can identify congestion in domains, rerouting flows to uncongested domains, and balances the load of domains when possible to improve inter-domain flow fairness. It achieves improvements over shortest path routing in both throughput and fairness in scenarios with load imbalance. Importantly, by taking into account the effects of interference of flows when switching between domains, and the elastic nature of TCP flows, it is suitable for implementation in realistic scenarios. The simulations conducted in *ns-2* prove its effectiveness, and its advantage over previously proposed schemes. It outperforms all the alternatives tested. In particular, it achieves an average flow throughput gain of 104% over the nearest gateway strategy. GWLB specially distances itself from other solutions when congestion or load imbalance between domains increases, showing that the protocol can cope more effectively in these situations.

Acknowledgment

References

[1] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*. ACM, 2003, pp. 134–146.

[2] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *MobiCom '04: Proc. of conference on Mobile computing and networking*. ACM, 2004, pp. 114–128.

[3] Y. Yang, J. Wang, and R. Kravets, "Interference-aware Load Balancing for Multihop Wireless Networks," University of Illinois at Urbana-Champaign, Tech. Rep.

[4] Y. Bejerano, S.-J. Han, and A. Kumar, "Efficient load-balancing routing for wireless mesh networks," *Comput. Netw.*, vol. 51, no. 10, 2007.

[5] P.-H. Hsiao, A. Hwang, H. Kung, and D. Vlah, "Load-balancing routing for wireless access networks," in *INFOCOM*, vol. 2, 2001, pp. 986–995.

[6] A. Raniwala and T.-C. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, 2005, pp. 2223–2234.

[7] V. Mhatre, H. Lundgren, F. Baccelli, and C. Diot, "Joint MAC-aware routing and load balancing in mesh networks," in *Proc. of the 2007 ACM CoNEXT conference*, 2007.

[8] C.-F. Huang, H.-W. Lee, and Y.-C. Tseng, "A two-tier heterogeneous mobile ad hoc network architecture and its load-balance routing problem," *Mobile Networks and Applications*, vol. 9, no. 4, 2004.

[9] B. Xie, Y. Yu, A. Kumar, and D. P. Agrawal, "Load-balancing and Inter-domain Mobility for Wireless Mesh Networks," in *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, 2006, pp. 409–414.

[10] H. Tokito, M. Sasabe, G. Hasegawa, and H. Nakano, "Routing Method for Gateway Load Balancing in Wireless Mesh Networks," in *ICN '09: Proceedings of the 2009 Eighth International Conference on Networks*, 2009, pp. 127–132.

[11] L. Ma and M. K. Denko, "A Routing Metric for Load-Balancing in Wireless Mesh Networks," in *AINAW '07, Advanced Information Networking and Applications Workshops*, 2007, pp. 409–414.

[12] K. Ramachandran, M. Buddhikot, G. Chandranmenon, S. Miller, E. Belding-Royer, and K. Almeroth, "On the Design and Implementation of Infrastructure Mesh Networks," in *Proceedings of the IEEE Workshop on Wireless Mesh Networks (WiMesh)*, 2005.

[13] D. Nandiraju, L. Santhanam, N. Nandiraju, and D. P. Agrawal, "Achieving Load Balancing in Wireless Mesh Networks Through Multiple Gateways," in *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, 2006, pp. 807–812.

[14] S. Maurina, R. Riggio, T. Rasheed, and F. Granelli, "On Tree-Based Routing in Multi-Gateway Association based Wireless Mesh Networks," in *The 20th Personal, Indoor and Mobile Radio Communications Symposium (PIMRC'09)*, 2009.

[15] S. Lakshmanan, R. Sivakumar, and K. Sundaresan, "Multi-gateway Association in wireless mesh networks," *Ad Hoc Netw.*, vol. 7, no. 3, pp. 622–637, 2009.

[16] M. Ito, T. Shikama, and A. Watanabe, "Proposal and evaluation of multiple gateways distribution method for wireless mesh network," in *ICUIMC '09: Proc. of the 3rd International Conference on Ubiquitous Information Management and Communication*, 2009, pp. 18–25.

[17] J. J. Galvez, P. M. Ruiz, and A. F. Gomez-Skarmeta, "A Distributed Algorithm for Gateway Load-balancing in Wireless Mesh Networks," in *Wireless Days, 2008. WD '08. 1st IFIP*, 2008, pp. 1–5.

[18] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, 1997.

[19] Y. Xue, B. Li, and K. Nahrstedt, "Optimal resource allocation in wireless ad hoc networks: A price-based approach," *IEEE Transactions on Mobile Computing*, vol. 5, no. 4, pp. 347–364, 2006.

[20] P. Gupta and P. R. Kumar, "The Capacity of Wireless Networks," *IEEE Transactions on Information Theory*, pp. 388–404, 2000.

[21] J. Li, C. Blake, D. S. J. D. Couto, H. I. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *MobiCom '01: Proc. 7th ACM Int. Conf. on Mobile Computing and Networking*, 2001.

[22] J. J. Galvez, P. M. Ruiz, and A. F. Gomez-Skarmeta, "Achieving Spatial Disjointness in Multipath Routing without Location Information," in *IEEE Wireless Communications and Networking Conference (WCNC 2009)*. IEEE Computer Society, 2009, pp. 1–6.

[23] K. Fall and K. Varadham, "The ns Manual," http://www.isi.edu/nsnam/ns/ns-documentation.html.