



Responsive on-line gateway load-balancing for wireless mesh networks

Juan J. Galvez*, Pedro M. Ruiz, Antonio F.G. Skarmeta

Dept. of Information and Communications Engineering, Computer Science Faculty, University of Murcia, 30100 Murcia, Spain

ARTICLE INFO

Article history:

Received 27 September 2010
Received in revised form 20 April 2011
Accepted 8 June 2011
Available online 15 June 2011

Keywords:

Wireless mesh network
Load-balancing
Fairness
Internet gateway
Single-source unsplitable flow

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. Traffic is balanced at the TCP flow level and, as a result, the aggregate throughput, average flow throughput and fairness of flows improves. The proposed scheme (referred to as Gateway Load-Balancing, GWLB) is highly responsive, thanks to fast gateway selection and the fact that current traffic conditions are maintained up-to-date at all times without any overhead. It also effectively takes into account intra-flow and inter-flow interference when switching flows between gateway domains. We have found the performance achievable by routes used after gateway selection to be very close to the performance of optimal routes found by solving a MINLP formulation under the protocol model of interference. 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 128% over the nearest gateway strategy.

© 2011 Elsevier B.V. All rights reserved.

1. 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. 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 this

as the gateway selection problem, and the set of nodes served by a gateway as its *domain*.

The selection of gateways influences the traffic distribution and load pattern inside the WMN. It is important to balance load between gateways to avoid overutilized and underutilized regions. Load imbalance can easily occur due to a number of factors, such as heterogeneous traffic demands, time-varying traffic and uneven number of nodes served by gateways. This can lead to inefficient use of network capacity, throughput degradation and unfairness between flows in different domains.

On the other hand, arbitrary load-balancing can hurt performance. In a wireless multi-hop network, interference (namely inter-flow and intra-flow interference) has a major role in network performance. Association of nodes to gateways must be chosen carefully, as it is important to maintain traffic locality and avoid flow interactions. Solving the problem therefore requires knowledge of the WMN to ensure high performance.

In addition, because traffic is dynamic and can vary frequently and unpredictably, gateway selection must be able to adapt to the current traffic conditions.

There are two approaches to solve the problem: centralized and distributed. The distributed approach generally

* Corresponding author.

E-mail addresses: jjgalvez@um.es (J.J. Galvez), pedrom@um.es (P.M. Ruiz), skarmeta@um.es (A.F.G. Skarmeta).

involves each router choosing the best gateway based on a routing protocol and metric used in the WMN.¹ To balance traffic, the routing metric must reflect current load. Examples of metrics which directly or indirectly take load into account include ETX [2], ETT [3] and MIC [4]. Distributed approaches must deal with several issues. One is gateway flapping, where a node continually changes its gateway selection. This occurs, for example, when multiple nodes discover an underutilized gateway at the same time, and switch simultaneously to this gateway, overloading it. Convergence time is also an issue because the time required to converge to a stable gateway selection may be longer than the time between traffic variations. Another issue is the overhead required to broadcast up-to-date load information through the network. For these reasons, in a distributed approach merely relying on a load-aware routing metric is not sufficient. In the next section we examine existing centralized and distributed approaches.

In this paper we study the benefits of centralized gateway selection, where the problem is solved outside the WMN. We develop an online algorithm called Gateway Load-Balancing (GWLB) which, given the current network conditions, efficiently selects gateways for every node or flow. We assume that most traffic is directed to/from the Internet. As such, there is no added overhead involved to determine current network load (this information is collected by gateways as traffic passes through them). The solution can be recalculated periodically with a very high frequency to ensure responsiveness. Because it is calculated centrally, it will not suffer from convergence issues. Furthermore, gateway flapping does not occur, and a node can be prevented from switching gateways until a specified time elapses. To prevent harmful load-balancing due to severe interference, the potential interference generated by gateway selection is taken into account based on knowledge of the network topology. Finally, we balance traffic at the TCP flow level, thus improving the throughput and fairness of flows. For these reasons, the proposed solution is suitable for implementation in practical and dynamic WMN scenarios.

The rest of the paper is organized as follows. In Section 2 we review related work. In Section 3 we describe the network model, define and formulate the gateway load-balancing problem. Section 4 describes the Gateway selection algorithm. Section 5 explains the GWLB protocol. In Section 6 we study the performance obtained by GWLB against optimal selection found by solving a MINLP formulation of the problem. Section 7 shows and analyzes protocol performance based on simulations with *ns-3*. Finally, in Section 8 we summarize our conclusions.

2. Related work

2.1. Centralized gateway selection

Many works [5–8] study the general problem of centralized routing in WMNs, which involves calculating routes

¹ When using this strategy, there must be a mechanism to ensure that traffic from the Internet to a mesh router enters the network through the gateway chosen by the router (e.g. as in [1]).

between every active source-destination pair. These solutions can be applied to the gateway selection problem, by assuming the existence of a “virtual” node (representing the Internet) connected to every gateway. This node is the source and destination of all traffic from/to the Internet. The routes calculated by these solutions will therefore determine the gateway used by nodes. Most centralized routing approaches have the following important drawbacks:

- Assume knowledge of an offline traffic matrix, which specifies the demand between all source-destination pairs [5,6], or allocate rate for every source [7,8].
- Use throughput models of the network that assume that the capacity of the network (links and contention regions) is known. To achieve this, they rely on simplifying assumptions: interference-free graph model [8], synchronized time-slotted access [6], binary interference models and throughput estimation [5,7].
- Calculate splittable *flow*, where the demand between a source and destination is split among multiple paths [6,7].

There are several limitations in the above assumptions when designing a solution for practical scenarios:

Traffic matrices are frequently used for traffic engineering in ISP networks, where capacity is above utilization. In this paper, however, we assume that the traffic matrix is unknown and, furthermore, contend that it cannot be known in WMNs for two main reasons: (a) because the capacity of a WMN is limited, users will tend to use all available capacity. This means that routing affects resulting demands, and those demands do not necessarily reflect user requirements. As soon as routing changes, demands can change; (b) traffic is dynamic and can constantly change.

Furthermore, we assume that network capacity is very difficult to predict for various reasons, including dynamic wireless medium, interference and MAC inefficiencies. For this reason, it is not possible to accurately predict throughput to guarantee that a certain rate allocation is feasible.

Finally, we seek an unsplitable flow, where individual flows cannot be split. Our problem is closely related to the single-source unsplitable flow problem, which is NP-hard [9].

Tokito et al. [10] propose a centralized gateway selection algorithm with the goal of balancing the load served by gateways. It assumes the current demand of each node is known, but does not specify how this information is obtained, and does not take contention inside the WMN into account when selecting gateways.

We remark that solutions such as the above that require the demand of nodes to be known assume that their demand will remain the same after a change in gateway selection, which is not true when traffic is constituted by TCP flows.

2.2. Distributed gateway selection

Raniwala et al. propose the *Hyacinth* architecture for multi-radio multi-channel WMNs in [1]. The routing

protocol dynamically maintains trees rooted at each gateway, by periodic exchange of load information in the network. The tree to which a node joins determines its gateway selection. The metrics proposed are based on estimating residual capacity and are not suitable for balancing elastic traffic. Simulation results for a sample scenario show a convergence time of two minutes. Mhatre et al. [11] propose a distributed algorithm to form a delay-optimal routing forest, where a tree is rooted at each gateway. The algorithm, however, is not load-aware, and instead assumes that all nodes and links in the forest will be active simultaneously.

Similar to [10], authors in [12,13] propose approaches with the goal of evening the load served by gateways. These solutions assume a specific demand for each node and thus are not designed for TCP traffic. They do not take contention into account, and convergence time is not studied.

Other distributed proposals are aimed at switching nodes to alternate gateways when a gateway becomes congested. These include the protocols WCETT-LB [14] and AODV-ST [15], which extend existing routing solutions. These can suffer from gateway flapping. Another example is the work by Nandiraju et al. [16], in which congested gateways notify nodes which should switch to another domain. This solution does not take into account the effects of interference when switching nodes and can also suffer from gateway flapping. Maurina et al. [17] attempt to improve the proposal in [16] 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 do not balance load in the absence of congestion. With TCP, domains may not be congested but flow unfairness between domains can occur.

2.3. Multi-gateway association

Other solutions [12,18,19] have proposed associating nodes with multiple gateways simultaneously. 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 [20] we showed how this occurs with the FP protocol from [12], in which nodes send to all gateways in proportion to gateway capacity.

2.4. Contributions of this work

In this work we design a solution that meets the following goals:

- Efficient traffic-aware centralized gateway selection. Because the solution is calculated centrally it does not suffer from convergence issues and gateway flapping is easily avoided. Furthermore, the solution is calculated frequently with a period of a few seconds to adapt to varying traffic conditions.
- Does not introduce any overhead to determine current traffic conditions.
- Takes into account the potential interference generated by gateway selection to avoid harmful load-balancing.

- Balances traffic at the TCP flow level, improving the performance and fairness of flows.

3. The gateway load-balancing problem

3.1. 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).

We assume no accurate knowledge of the capacity and interference model. This is difficult to obtain in practice in a WMN and can be subject to frequent change.

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. Because users are randomly located and can generate connections at any time, network conditions can constantly change.

3.2. Gateway load-balancing overview

An Internet flow is a set of TCP packets exchanged between two endpoints, one in the Internet and one in the WMN. The network operator is responsible for choosing the exact properties that define an endpoint.²

Given a set of Internet flows, we want to find the best gateway for each flow, to optimize their performance (flow throughput and fairness). We call the router in the WMN participating in a flow a *sink*. In general, flows are created after a user in the WMN connects to a server on the Internet.

We assume that a routing protocol is executed *inside* the WMN, which establishes routes and performs routing between every pair of nodes in the WMN, including the gateways. Note that this protocol is *not* responsible for gateway selection, but will however route traffic between the chosen gateways and routers. We do not impose any such protocol or routing metric.

We require that, at any one time, the traffic belonging to a flow be served by only one gateway. This is necessary to simplify routing and to reduce the probability of out of order delivery of packets and Round-trip time variations.

² In this paper an endpoint is a pair (address, port).

This relates our problem to the single-source unsplitable flow problem, which is known to be NP-hard [9,21].

Because routes inside the WMN are already given, the gateway load-balancing problem consists in choosing the serving gateway for each flow. Let \mathbb{G} be the set of gateways in the WMN, \mathbb{F} the set of current flows and \mathbb{S} the set of sinks. Let $G = |\mathbb{G}|$, $F = |\mathbb{F}|$ and $S = |\mathbb{S}|$. A gateway assignment is a function $\alpha_f : \mathbb{F} \rightarrow \mathbb{G}$. We denote $\alpha_f(f_i) = g$ if flow f_i is assigned to gateway g . The actual path leading from the gateway g to the flow’s sink d is given by $P_{g \rightarrow d}$, which denotes the path used by the WMN routing protocol to connect nodes g and d .

Definition 1. A gateway’s native domain is the set of nodes closest to the gateway, per the WMN routing protocol metric.

Fig. 1 illustrates a simple gateway load-balancing example. In this example we assume the use of hop-count as routing metric and shortest path routing inside the WMN. In this case, the topology is unbalanced, i.e. the size of each native domain is different. An unbalanced topology can contribute to load imbalance, however, most frequently, traffic patterns are responsible for load imbalance (as is the case in this example where the most loaded domain is the smallest one). The colored nodes represent active sinks, each participating in one flow. The *Nearest Gateway* (NGW) solution, which uses native domains, can easily lead to load imbalance, as illustrated in Fig. 1a. In this situation, all flows are routed through GW_B , which means that all the load is concentrated on the region surrounding this gateway. However, the load can be balanced adequately by routing flows f_2 and f_4 through GW_A (Fig. 1b), thus utilizing the capacity of domain A and consequently

increasing the bandwidth share of all flows. The example also illustrates that minimizing imbalance usually comes at the expense of increased path cost (longer paths and consequently increased interference). Flows f_2 and f_4 are routed through longer paths in Fig. 1b. Also note that it would not be desirable to route flows f_1 and f_3 through gateway GW_A , because this would create very long paths which would generate high interference.

The NGW solution can be considered a minimum path cost solution but can lead to high imbalance. In contrast, long paths have a high penalty in single-channel multi-hop wireless networks, where the interference factor plays a major role. The capacity of a chain of nodes and the locality of traffic are key factors influencing network capacity [22,23]. Long paths and increased contention can severely impact performance. It is therefore important to limit path cost, and a trade-off is involved to optimize both load balance and path cost. How this optimization is performed will be explained in detail later.

Our approach to gateway load-balancing is to perform gateway selection for Internet flows in a centralized manner. The main reason is that a centralized controller located in one of the gateways or outside the WMN can easily perform load-balancing, providing many benefits in this scenario as explained below.

The information needed by the controller for gateway selection is the current set of flows \mathbb{F} and the current network topology and routes. Gateways collect information on flows at the same time as traffic passes through them and immediately send this information to the controller (a system such as Netflow [24] or IPFIX [25] can be used for this purpose). Controller and gateway communication is done through the wired network. Network connectivity

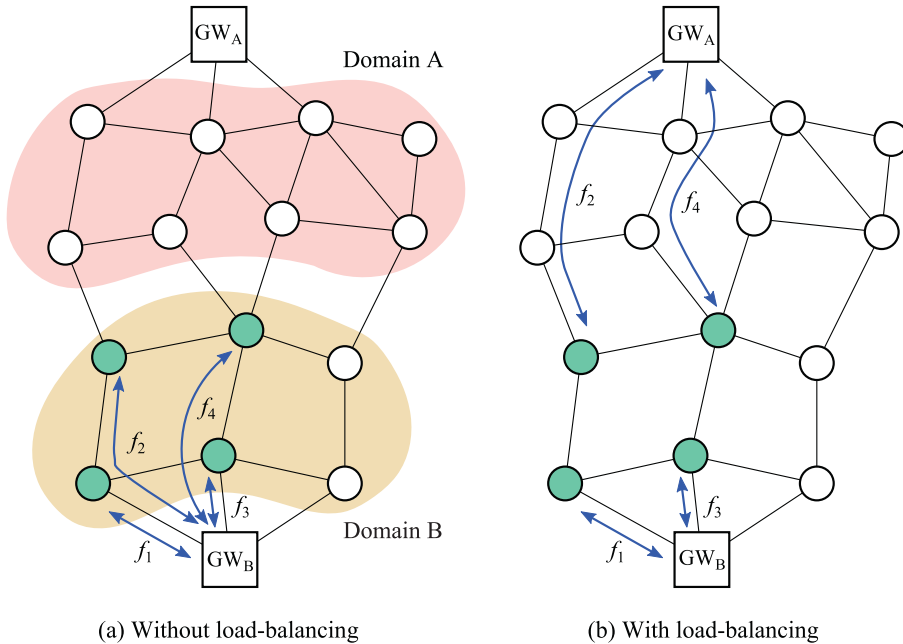


Fig. 1. Simple gateway load-balancing example. There are two gateways, GW_A and GW_B , with their corresponding native domains A and B. Colored nodes are participating in one flow (flows labeled f_1 to f_4). Arrows indicate paths followed by flows. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and routes is assumed to be obtained from the WMN routing protocol (standard link-state protocols like OLSR provide this information). With the information of the current set of flows, the controller executes a fast gateway selection algorithm, calculating α_f and informs the gateways of the new association. Download traffic of a flow will be injected into the WMN by gateways based on this association. The gateway used by download traffic will be recorded in download packets, and the sink will therefore know which gateway to send upload traffic of the flow to.

This approach has the following benefits. Given that, at any instant, the controller knows the *current* set of flows and given the low complexity of the gateway selection algorithm, it is possible to periodically recalculate α_f with a very high frequency (in our evaluations we have used an adaptation frequency as high as once per second). Therefore the controller can react quickly to the changing network conditions (new flows entering the network and flows that have ended). Because the solution is calculated centrally, there are no non-convergence problems which can arise in distributed implementations. That is, given the current state, the algorithm converges to an association α_f , and this solution remains the same until the next calculation. In addition, gateway oscillations can be easily avoided by locking the gateway association of a given flow for a specified time. Finally, the proposed solution does not introduce any control overhead in the WMN besides the routing protocol overhead.

Further details on the GWLB architecture and protocol for controller-gateway coordination are given in Section 5. In the following sections we focus on the gateway selection algorithm which is executed periodically at the controller. Table 1 lists the symbols used throughout the text to explain the algorithm.

3.3. Measuring load

In this work we consider dynamic traffic based on TCP. Flows are elastic and do not have a specific demand. Rate control and bandwidth sharing are performed by TCP. In other words, flows have the property of adapting to the available capacity and sharing bandwidth between them.

Table 1
GWLB symbols.

\mathbb{G}	Set of gateways in the network
\mathbb{F}	Set of Internet flows
\mathbb{S}	Set of sinks in the network
S_u	Set of unlocked sinks
α_f	Per-flow gateway assignment function: $\alpha_f(f_i) = g$ if flow f_i assigned to gateway g
α_s	Per-sink gateway assignment function: $\alpha_s(s_i) = g$ if sink s_i assigned to gateway g
$sink(f_i)$	Endpoint of flow f_i in the WMN
ND_g	Native domain of gateway g
$P_{m \rightarrow n}$	Path used by the WMN routing protocol to connect nodes m and n
P_0	Maximum accepted path cost
$load(s)$	Number of flows of sink s
$load(g)$	Number of flows assigned to gateway g
VG_s	Set of gateways which can be used to reach sink s
VG	$\sum_{s \in S_u} VG_s $

Taking the above into account, the load of a gateway g is measured as the number of flows served by the gateway, i.e. $load(g) = |\{f_i \mid \alpha_f(f_i) = g\}| \forall f_i \in \mathbb{F}$. Imbalance is the variance of the number of flows served by gateways. If gateways serve a similar number of flows, we can expect flows to better share all the network capacity, improving both average flow throughput and fairness. Our evaluations in Sections 6 and 7 show that this is a suitable measure to balance load with fairness in mind, better than metrics employed by previous works. The reason is that other metrics do not consider the adaptive nature of TCP flows. For example, when using TCP, congestion is usually temporary due to congestion control, and is not a reliable load indicator. In addition, flows do not have a specific demand but rather tend to use all available capacity (specially true in a WMN where capacity is limited). This means that a low number of flows can produce the same demand as a large number of flows, and therefore the demand is not a reliable indicator to balance load.

Knowledge of flows is collected by gateways as traffic passes through them, requiring no overhead in the WMN.

3.4. Measuring path cost

The WMN routing protocol establishes paths between gateways and sinks. These paths are known by the controller. The controller performs its own estimate of the cost of these paths. This estimate is used when calculating gateway selection, in order to avoid using paths that can cause harmful interference. Note that this cost is unrelated to the cost of routes measured by the routing protocol. In this paper, we will always refer to the cost of paths as calculated by the controller. We now explain how it is measured.

The controller needs the topology graph and routes used by the routing protocol to connect gateways and sinks. This information is assumed to be obtained from the routing protocol. Note that link-state protocols such as OLSR already provide this information, with no additional overhead other than the default routing overhead (more details on how this information is obtained are given in Section 5). Given the topology graph, the shortest distance (in hops) between all pairs of nodes is known. This can be calculated by Dijkstra's algorithm and need not be recalculated unless the topology changes.

The path cost metric is designed to meet the following goals:

First of all, it must only depend on the topology. This means that the cost of a particular path only needs to be recalculated if the topology changes.

Second, we desire the metric to account for increased network interference when choosing a serving gateway for a sink different from its nearest gateway.

For these reasons, the cost of a path from a gateway g to a sink s is measured as the expected interference of the path to nodes which are not likely to be served by g . That is, we desire the path to minimally interfere with nodes not served by the gateway. We define the set of nodes likely to be served by a gateway as the sinks closest to it (per the WMN routing metric). Note that this measure also implicitly accounts for path length (longer paths will have

increasingly higher cost because they will be farther away from the gateway and closer to nodes of other domains).

To derive this metric, we first define the distance of a node to a set of nodes, and the distance of a path to a set of nodes.

First we give some definitions:

- A path p is a sequence of unique nodes such that there exists an edge in the topology 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 (in terms of hops).

The distance of a node n to a set of nodes \mathbb{D} is the minimum distance from n to a node in \mathbb{D} , as shown in Eq. (1). The distance from a path p to a set of nodes \mathbb{D} is the arithmetic mean of the distance of nodes of p to \mathbb{D} , as per Eq. (2).

$$\delta_n(n, \mathbb{D}) = \min_{m \in \mathbb{D}} \{\delta(n, m)\}, \quad (1)$$

$$\delta_p(p, \mathbb{D}) = \frac{\sum_{n \in p} \delta_n(n, \mathbb{D})}{|p|}, \quad (2)$$

$$pcost(P_{g \rightarrow s}) = -\delta_p(P_{g \rightarrow s}, \mathbb{S} - \text{ND}_g). \quad (3)$$

Finally, the cost of a path from gateway g to sink s is the negative distance of the path to nodes which are not in the native domain of g . The less distance to nodes which are not in the native domain of g , the higher the cost. The maximum path cost is 0.

Note that distance is measured in hops. In [26] we proved that a distance metric based on hop-count can correlate highly with Euclidean distance between nodes, and is suitable for measuring spatial separation. Increased spatial separation decreases the probability of interference.

3.5. Gateway selection problem

The problem consists in finding a flow-gateway association α_f that balances the load of gateways and limits the path cost inside the WMN, consequently increasing flow throughput and fairness. To that end, we propose solving the following multi-objective problem:

$$\mathbf{P}_{\text{GS}} : \min_{\alpha_f} \left[\max_{g \in \mathbb{G}} \{load(g)\}, \sum_{f_i \in \mathbb{F}} pcost(P_{\alpha_f(f_i) \rightarrow sink(f_i)}) \right]. \quad (4)$$

The first objective seeks to minimize the maximum number of flows served by a gateway. The second objective seeks to minimize the cost of paths in order to avoid interference in the network.

By minimizing the utilization of a gateway, the capacity of the gateway domain can be shared among fewer flows, increasing average flow throughput. Because load is balanced across contention regions, fairness is improved (e.g. by avoiding cases where there are regions with many contending flows and regions with few). Aggregate throughput can also improve by routing through otherwise unused gateways.

Two modes of association are supported: (a) per-sink association, where all flows of the same sink use the same

gateway, and (b) per-flow association where no such restriction exists (i.e. flows of the same sink can arrive through different gateways). The advantage of per-flow association is that it can achieve better load balancing (although this benefit may not be attainable in single-channel WMNs as will be explained later), while the advantage of per-sink association is that it only needs to store sink-gateway associations and gateway selection is performed faster.

The key to solving \mathbf{P}_{GS} is determining the optimal trade-off between the objectives, establishing bounds on what is considered a valid path. This trade-off can vary from network to network, and depend on its current conditions, and as such can appear rather difficult to determine in the absence of an accurate network model.

In the next section we propose a method to solve the problem and show how with basic knowledge and understanding of the network topology and traffic, and using an approximation algorithm, high-performance gateway selection can be efficiently calculated. In Sections 6 and 7 we evaluate this.

4. Gateway selection procedure

The gateway selection algorithm (GSA) is executed centrally in the controller. To solve \mathbf{P}_{GS} , we impose an upper bound on the cost of valid paths, i.e. paths with cost higher than this bound are not considered valid. This determines the set of paths which can be used to reach a particular sink, and this in turn translates into the set of gateways which can serve the sink. We then focus on minimizing the first objective, which constitutes a load-balancing problem equivalent to the Restricted Scheduling problem [27], which in turn is an instance of Scheduling in Unrelated Parallel Processors [28]. These problems are NP-hard, and are closely related to the single-source unsplitable flow problem (restricted scheduling can be expressed as an instance of it [9]). In the restricted scheduling problem, there are n tasks and m machines. A task can only be assigned to a subset of machines. The objective is to assign tasks to machines such that the makespan is minimized. In our case, the gateways are the machines, and the flows (or alternatively sinks) are the tasks. The time to process a task is one (per-flow association) or the number of flows of the sink (per-sink association).

Because the problem is NP-hard, we propose a heuristic algorithm to quickly approximate an optimal solution. A fast algorithm is essential in order to be able to calculate flow-gateway associations periodically with a high frequency.

4.1. Generating candidate paths

The valid paths (gateways) to reach a sink s are given by:

$$VG_s = \{g \in \mathbb{G} : pcost(P_{g \rightarrow s}) \leq P_\theta\}. \quad (5)$$

This list is ordered in ascending order of cost, and updated when the cost of a path changes. The cost of a path is recalculated if the path to the sink changes, or the topology changes.

Possible values of the threshold P_θ are in the $[-X, 0]$ range, where $-X$ is the highest cost of a path in the NGW solution. This guarantees that every sink has at least one valid gateway. A higher threshold increases the number of alternative paths to reach a sink (and number of balancing options), but higher path costs will decrease performance. Finding a good value for the threshold requires some knowledge of the network characteristics, mainly, the average interference range (spatial reuse), and hop distance correlation with Euclidean distance (which essentially depends on the average link distance). A lower interference range permits a higher path cost, while a very high interference range may even prevent exploiting gateway load-balancing. The value of P_θ should be set close to the average interference range.

4.2. Gateway selection algorithm

GSA (Algorithm 1) is a fast greedy algorithm to solve P_{GS} , which focuses on evening the load of gateways, without incurring in high path costs.

As mentioned, the load-balancing problem matches the restricted scheduling problem, which is NP-hard. To our knowledge, the best approximation algorithm to solve this particular problem was recently proposed by Gairing et al. [27]. However, because our system has a requirement for high responsiveness, we propose using a faster algorithm which achieves near optimal results on average [29]. Our algorithm is an adaptation of the List Scheduling algorithm commonly used in task scheduling problems. We choose a special order for the sinks which, for this particular instance of the problem, we have found to offer better results than similar existing heuristics [29].

We only show and explain the per-sink association variant, which is the one used in the simulation tests. The per-flow variant is very similar, only eliminating the restriction that all flows of a sink use the same gateway. Although per-flow association achieves better balance at a similar path cost, the performance results observed in a single-channel WMN are almost identical to the per-sink algorithm. This is due to increased contention in the network and around the sinks.

GSA iterates over the list of currently unlocked sinks,³ in each iteration choosing the best gateway for the selected sink. The fitness of the solution depends highly on the order of sinks. A heuristic is used to order the sinks, determined by the comparison function of Algorithm 2.

Algorithm 1. GSA per-sink selection algorithm.

```

1:  $S_u := \{s_i \in \mathbb{S} : \neg \text{locked}(s_i) \wedge \text{load}(s_i) > 0\}$ 
2:  $\alpha_s(s_i) \leftarrow \emptyset \quad \forall s_i \in S_u$ 
3: Sort  $S_u$  // Comparison sort based on Algorithm 2
4: for  $s_i \in S_u$  do
5:    $\alpha_s(s_i) \leftarrow \text{argmin}_{g \in VG_{s_i}} \text{load}(g)$ 
6:    $\text{load}(\alpha_s(s_i)) \leftarrow \text{load}(\alpha_s(s_i)) + \text{load}(s_i)$ 

```

³ Unlocked sinks are sinks with no locked flows. When a flow is allocated to a gateway, GWLB locks the flow for a specified time (see Section 5).

Algorithm 2. Sink comparison function used for sorting.

```

1: function less_than(s,t) do
2:   if ( $|VG_s| = 1$ ) and ( $|VG_t| > 1$ ) then
3:     return true
4:   if ( $|VG_s| > 1$ ) and ( $|VG_t| = 1$ ) then
5:     return false
6:   if  $\frac{\text{load}(s)}{|VG_s|} > \frac{\text{load}(t)}{|VG_t|}$  then
7:     return true
8:   else
9:     return false

```

First we explain the greedy algorithm. In line 1, the list of unlocked sinks is initialized, containing sinks currently unlocked (with no locked flows). Because these sinks are now free to be routed through any gateway, their current gateway association is cleared (line 2). The list of unlocked sinks is ordered in line 3. For each unlocked sink, GSA chooses the current least loaded gateway (less number of flows) as its current gateway (lines 4–6). Note that if there is more than one least loaded gateway, the one with lowest cost to reach is chosen, because VG_s is ordered in ascending order of path cost. This contributes to generating paths with lower cost.

The comparison function of Algorithm 2 is a less-than operator which determines the order of sinks. Sinks with only one valid gateway are selected first to be assigned (lines 2–5). For all other cases (lines 6–9), a heuristic similar to Longest Processing Time (LPT) first is used. The LPT heuristic specifies that jobs with more load go first (in this case sinks with more flows). For this particular problem, however, we find that it is also important that sinks with more alternatives be picked last, because they offer more possibilities of balancing load, and these possibilities should not be used up too early. We combine both factors in the decision (line 6).

Theorem 1. *The time complexity of GSA in the worst case is $O(S + F + |S_u| \log |S_u| + VG)$.*

Proof. Line 1 requires traversing all sinks and all flows in the worst case: $O(S + F)$.

In line 3, the list of unlocked sinks is sorted by a comparison sort, requiring $O(|S_u| \log |S_u|)$.

The loop of lines 4–6: Line 5 requires traversing the valid gateways of the sink. The loop runs in $O(|S_u| \frac{VG}{|S_u|} + |S_u|) = O(VG + |S_u|) = O(VG)$. \square

5. Gateway load-balancing architecture and protocol

This section contains details on the GWLB architecture and protocol. This mainly refers to the process by which the controller coordinates with gateways to obtain the necessary network state, communicates the flow-gateway association to gateways and how they apply the solution.

The controller can be co-located with any one of the gateways or be an independent device. Controller and gateways communicate through the wired network. 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). Gateways collect network state and send it to the controller. Based on the current state, the controller calculates the serving gateway of each flow. The gateway selection algorithm (explained in Section 4) executes periodically in the controller every G_T seconds (in our evaluations $G_T = 1$). Each time the flow-gateway association α_f is calculated, the controller sends α_f to gateways in order to enforce the solution.

5.1. Routing download traffic

Download traffic is routed based on the current association α_f , i.e. a gateway is responsible for injecting the traffic of its assigned flows inside the WMN:

- If a gateway g receives traffic of flow f_i and the flow is not assigned to g , i.e. $\alpha_f(f_i) \neq g$, the gateway forwards the traffic to its serving gateway $\alpha_f(f_i)$ (directly or via a tunnel through the wired network if the gateways are in different sub-networks).
- If the traffic belongs to one of its served flows, i.e. $\alpha_f(f_i) = g$, it routes it through the WMN using the configured routing protocol. The gateway records its address in the packets' header.

5.2. Routing upload traffic

When a sink has to send upload traffic of a flow, it sends it to the gateway last used by download traffic of the flow (this information is recorded in download packets). If the gateway is currently unknown, it sends it to the nearest gateway (per the WMN routing protocol metric).

5.3. Network state monitoring

To calculate a flow-gateway association α_f , the controller needs the following information:

- WMN topology graph.
- Routes from every gateway to every sink: $P_{g-s} \forall g \in \mathbb{G}, \forall s \in \mathbb{S}$.
- Knowledge of active flows \mathbb{F} .

Topological information and routes are obtained by gateways from the WMN routing protocol and this information is sent to the controller. The frequency used by the routing protocol to send this information is independent of G_T .

TCP traffic is classified into flows. Traffic classification is performed by the gateways when traffic passes through them, and this information is sent to the controller. A system such as IPFIX [25] can be used for this. Note that these systems employ two basic mechanisms to determine if a flow has finished: flow aging (no packets have been observed for a specified time) and detection of TCP session termination messages.

The controller will have knowledge of all flows but, obviously, flows with a duration shorter than G_T are not

guaranteed to be taken into account by the GSA. This does not pose a problem when G_T is very small, because the flows which are not taken into account will be very small-sized flows, which have negligible repercussions on the imbalance of network traffic.

5.4. Gateway maintenance

The first time a controller is elected, and every time a new gateway appears or disappears, the controller obtains the current network state, recalculates α_f and informs all gateways. This is done to react to the presence of gateways in the network, adapting to the gateways present at any time. If a gateway fails, its flows will be re-routed through the remaining gateways. If the controller shuts down or fails, a new one is elected.

5.5. Load-balanced gateway selection

The load-balancing algorithm executes every G_T seconds, choosing the serving gateway of every flow. The algorithm was explained in Section 4.

When a serving gateway is chosen for a flow, the flow is locked for $Lock_t$ seconds. The algorithm will not re-route locked flows. A sink is considered locked if one of its flows is locked.

5.6. Benefits of GWLB strategy

There are important benefits to the proposed strategy:

- The only required overhead is the normal routing protocol overhead, and the recording of the gateway address in download packets. Flow classification and book-keeping is done at the controller and gateways. Distributed protocols require periodic advertisement of load (e.g. gateway load) through the whole network. Frequent advertisements can introduce too much overhead, degrading performance, while a low frequency will limit adaptation to changing conditions.
- Low-complexity load-balancing algorithm which can be executed periodically with a high frequency, adapting promptly to changing conditions.
- Prevents gateway flapping and non-convergent behavior, because the solution is calculated centrally. Gateway flapping can occur when multiple nodes discover an underutilized gateway at the same time. The nodes then switch simultaneously to this gateway, overloading it. In a distributed protocol (e.g. Hyacinth), a mechanism is needed to prevent this situation. However, this mechanism normally results in arbitrary decisions as to which nodes switch and which do not, leading to sub-optimal solutions.
- Frequent gateway oscillation of a given flow is also prevented by locking a flow for a specified time when a gateway is chosen.

6. Analysis of GSA routes

Gateway selection calculated by GSA determines the set of routes which will be used to transfer Internet traffic (we

call these GSA routes). In this section we study how these routes approximate routes found by optimally solving a rate allocation and routing problem which assumes perfect knowledge of the capacity and interference model.

6.1. Gateway load-balancing MINLP formulation

This formulation involves routing and rate allocation in order to optimize a throughput and fairness criteria.

Let \mathbb{F} be the set of flows, and r_i the rate of flow i (we assume that upload traffic is negligible and optimization involves the download direction). Let $F = |\mathbb{F}|$ and $G = |\mathbb{G}|$. A reasonable objective for WMNs 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 $\ln(r_i)$ [30].

A flow can be routed through G different paths, corresponding to the paths from each gateway to the sink. A solution 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: $\sum_{j=0}^G b_{ij} = 1, \forall i \in \mathbb{F}$.

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 [31]. 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 [22]. In this model, a transmission is successfully received only if the distance between sender and receiver is less or equal to the transmission range, and if no other node within interference range of the sender or receiver is transmitting at the same time (this assumes that the sender performs physical carrier sensing). Transmission and interference range are fixed global quantities.

A solution is given by the vector $\mathbf{x} = (b_{ij} x_{ij}), \forall i \in \mathbb{F}, \forall j \in \mathbb{G}$. The clique-flow matrix [31] $\mathbf{R} = \{R_{qt}\}$ 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:

$$\mathbf{P}_{\text{opt}} : \text{maximize } \sum_{i=0}^F \ln \left(\sum_{j=0}^G b_{ij} x_{ij} \right), \quad (6)$$

$$\text{subject to : } \mathbf{R}\mathbf{x} \leq \mathbf{C}, \quad (7)$$

$$\sum_{j=0}^G b_{ij} = 1 \quad \forall i \in \mathbb{F}, \quad (8)$$

$$b_{aj} - b_{bj} = 0 \quad \forall (a, b) \in \mathbb{F} \times \mathbb{F} \\ : \text{sink}(a) = \text{sink}(b) \quad \forall j \in \mathbb{G}. \quad (9)$$

Constraint (9) is optional, used to enforce per-sink routing. The constraint forces every pair of flows of the same sink to

have the same value of binary variables, thus guaranteeing that they are routed through the same gateway.

6.2. Routes comparison of \mathbf{P}_{GS} and \mathbf{P}_{opt}

To study how GSA routes can approximate routes found by \mathbf{P}_{opt} we use the following methodology:

- Solve random traffic scenarios in a WMN topology with \mathbf{P}_{opt} . Given a set of flows, this obtains the optimal routes and rate allocation which maximize the aggregate utility of the flows.
- Solve the same scenarios with GSA (Algorithm 1). This determines the GSA routes for the given flows. To evaluate the quality of these routes and compare with \mathbf{P}_{opt} , we determine the maximum aggregate utility of flows which can be achieved using these routes, and compare with the aggregate utility calculated by \mathbf{P}_{opt} . To this end, we input the paths determined by GSA into the NLP version of \mathbf{P}_{opt} , which only performs rate allocation. In this version, there are no integer variables because routes are fixed.

Because routes found by \mathbf{P}_{opt} depend on a particular network and interference model (Eq. (7)) (based on the protocol model of interference), this methodology allows us to compare both solutions under the same model.

The topology used consists of a static network of 100 nodes placed randomly in a square 2000 m \times 2000 m area. The topology is connected, with a minimum distance of 160 m between nodes, and maximum distance of 250 m. There are four gateways, one in each corner of the square. The protocol model of interference is used to generate the conflict graph and cliques, which determine constraints of the optimization problems. Maximum transmission range is 250 m and interference range is 550 m. Clique capacity is set to 280 KB/s. GWLB is configured with $P_o = -2.5$ (represents a cost value close to interference range).

To generate random traffic with and without imbalance, we vary the number of flows in each gateway's native domain. The number of flows per native domain is a value in $\{0, 5, 10, 15, 20\}$. We generate scenarios using all possible combinations of flows. Because there are four domains and five possible quantities of flows per domain, the total number of different scenarios is $5^4 - 1 = 624$ (eliminating the case with zero flows). For every unique scenario, flows are randomly assigned to a sink in the native domain.

To solve the MINLP problems, we used the Couenne solver [32,33], with the following parameters: a time limit of three hours, absolute termination tolerance of $1e - 9$ and relative termination tolerance of 0.005. All problems were solved before the time limit. Note that the level of approximation to the optimal solution depends on the tolerance thresholds, i.e. the solver may not reach the optimal solution, but rather a solution close to the optimal.

For comparison, we include routes used by NGW solution. In addition, to show the importance of taking path cost into account when selecting gateways, we define two gateway selection solutions which perform load-balancing in the same way as GWLB, but with little or no regard to path cost. They are called *arb_lb1* and *arb_lb2*,

respectively. In both solutions, all gateways can be used to reach every sink, i.e. $|VG_s| = G, \forall s \in \mathcal{S}$. Both solutions use Algorithm 1, with the following differences:

- In *arb_lb1*, if there is more than one least loaded gateway in one iteration, one is chosen at random (line 5 of Algorithm 1).
- In *arb_lb2*, if there is more than one least loaded gateway in one iteration, the least cost gateway is chosen (default behavior of Algorithm 1).

Fig. 2 shows the results. Each point is the average result of the scenarios that fall in that category. Confidence intervals are shown at the 95% level. Fig. 2a–c present the results under varying total number of flows in the network, while Fig. 2d–f present the results under varying flow imbalance between native domains. Fig. 2a and d show the objective maximized by the MINLP and NLP problems (aggregate utility of flows). Fig. 2b and e show the minimum rate of flows, and Fig. 2c and f show the average rate of flows. As we can see, the performance of GSA routes is very close to the performance of routes found by P_{opt} . The performance of *arb_lb1* is noticeably worse than NGW in most cases. The *arb_lb2* solution is also seen to perform worse than NGW in some cases, and rarely performs better.

The solutions found in this section seek the optimal rate allocation to achieve proportional fairness, given a protocol model of interference. In the next section, we evaluate GWLB by simulation using TCP flows.

7. Performance evaluation

We evaluate the performance of GWLB in WMN scenarios with multiple gateways and a varying number of users in each native domain. We simulate GWLB with *ns-3* [34] and compare it against three different gateway selection solutions: NGW, MLI [12] and MAU [17]. We choose to compare with MLI and MAU because, from our understanding of different proposed load-balancing protocols, they represent one of the best performing solutions of two different approaches used in the literature to execute load-balancing (explained in Section 2). MLI seeks to even the load of gateways and MAU seeks to alleviate congestion by switching sinks from congested to uncongested gateways. In [20], we found MLI to be the best performing of the protocols in [12]. MLI indirectly takes contention into account, which enables it to perform better than solutions with similar goals.

MLI and MAU are distributed protocols, but our implementation of them is centralized with no control overhead. Therefore, our implementation achieves an upper bound on their real performance.

For MLI, decisions are made centrally with complete knowledge of the topology, in each execution switching border sinks (sinks which neighbor nodes assigned to a different gateway) to a less loaded gateway. Border sinks are the first to receive advertisements from gateways different from their current gateway. In addition, the results of MLI illustrate the performance of its load metric, which is based on the current residual load of a gateway. Hyacinth

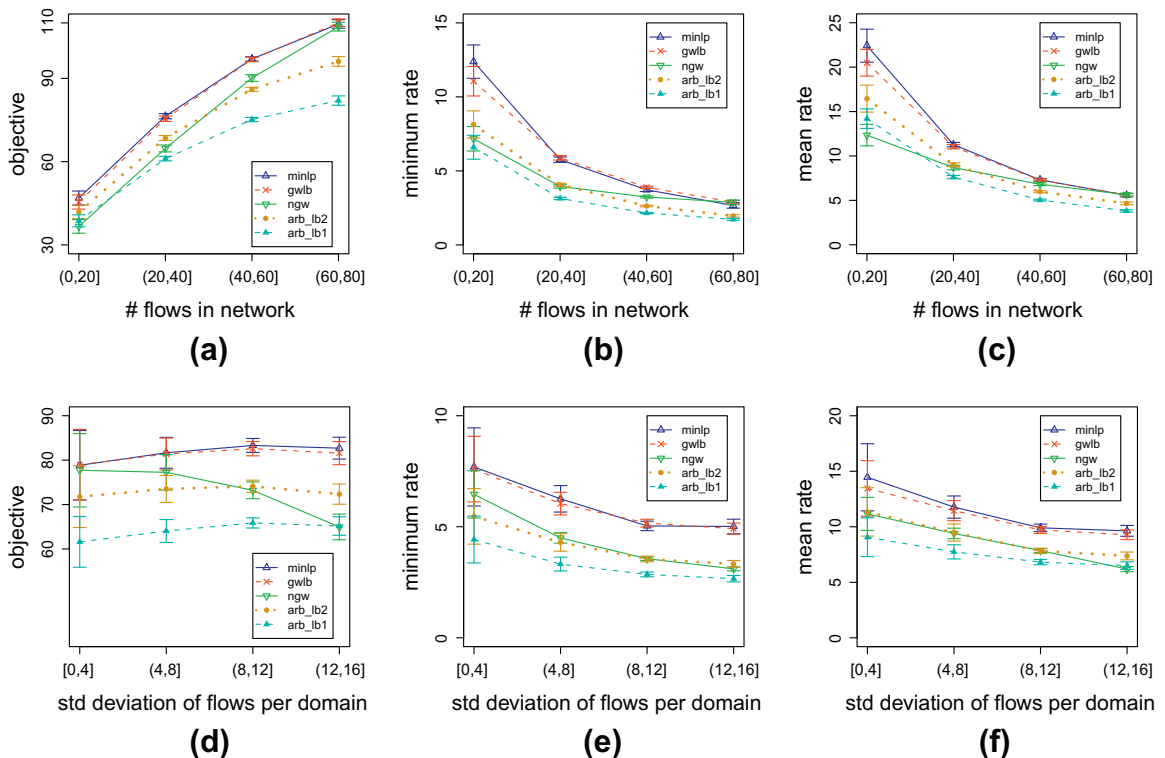


Fig. 2. Performance comparison between MINLP, GWLB, NGW and arbitrary load-balancing solutions.

[1], which uses a routing solution similar to MLI, also uses this metric. As we will see, this metric is not suitable for scenarios with TCP flows, where the flows generally use all available gateway capacity, i.e. the residual capacity of the gateways is always close to zero, independent of flow imbalance between gateways.

In our implementation of MAU, a decision is made centrally to switch the best candidate sink of a congested gateway to its nearest uncongested gateway. Load is measured as the average queue length of the gateway during a time period. It is measured at the gateway because they are the likely bottlenecks. As we will show, congestion is not a reliable measure for balancing TCP flows. Flows perform congestion control, therefore congestion is usually temporary and does not correlate with the number of flows served by a gateway.

7.1. Simulation environment

We have implemented the protocols in *ns-3* in the following manner (illustrated in Fig. 3). 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. \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 gateway selection (which is algorithm-dependent). \mathcal{B} knows the complete topology, active flows, and can read the state of gateway queues. Routing inside the WMN is static shortest-path based on hop-count.

The parameters used to configure wireless interfaces and propagation in *ns-3* are shown in Table 2. The transmit power chosen is the minimum power that guarantees a Packet Error Rate (PER) of at most 5% for all links in the network (with no concurrent transmissions). The energy detection threshold is the received power at the maximum link distance. Carrier sense threshold is derived based on a ratio of $CStH/EDth$ of -9 dB. We choose this ratio because it produced maximum aggregate throughput in numerous tests.

The WMN topologies consist of static networks of 100 nodes placed randomly in a square $2000\text{ m} \times 2000\text{ m}$ area. All topologies are connected, with a minimum distance of 160 m between nodes. There are four gateways, one in each corner of the square, resulting in four domains.

Traffic is TCP. Users connect to \mathcal{A} , generating flows. For each user, the time between start of connections follows an exponential distribution with $\lambda = 1/10,000$ ms, and the size

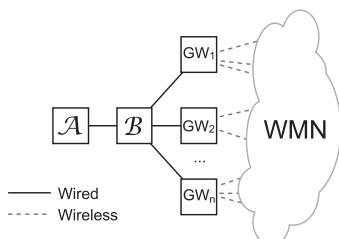


Fig. 3. Simulation architecture implemented in *ns-3*.

Table 2
ns-3 wireless parameters.

Wifi standard	802.11b @ 11mbps constant rate
Path loss model	Log-distance, exponent = 2.7
RTS/CTS	Disabled
Transmit power	$\min\{p:PER(p,l) \leq 5\% \} \forall l \in L$
Energy detection threshold (EDth) (W)	RxPower(max neighbor distance)
Carrier sense threshold (CStH) (W)	$EDth \times 10^{-9/10}$

of the download traffic of a flow follows an exponential distribution with $\lambda = 1/65,000$ byte. Simulations last until all flows finish, with users starting connections during the first 100 s.

The number of users per native domain varies in $\{0, 5, 10, 15, 20, 25\}$. Given a topology, we generate scenarios using all combinations of users on every native domain. Because there are four domains and six possible quantities of users per domain, the total number of different scenarios given a topology is $6^4 - 1 = 1295$ (excluding the case where all domains have zero 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 native domain.

We have set the parameters of GWLB to $G_T = 1s$, $Lock_t = 1s$ and $P_\theta = -2.5$. A value of $G_T = 1s$ provides high responsiveness to changing conditions. For P_θ , we have chosen a value close to the average interference range. Both MLI and MAU are also applied every second. The congestion threshold value of MAU is set to 50%. The parameters for the MLI scheme are $\Delta_l = 1.3$ and $P_{MLI} = 0.7$, as in [12].

Confidence intervals are shown at the 95% level.

7.2. Performance metrics

We use the following metrics to study protocol performance. Flows refer to TCP download flows: (i) *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; (ii) *Flow duration* – time elapsed since the first packet of a flow was sent and the last packet was received; (iii) *Flow throughput* – the number of bytes delivered divided by the duration of the flow; (iv) *Flow throughput fairness* – rates the fairness of the throughput achieved by flows in one scenario using Jain's fairness index.

The flow duration and throughput per scenario 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.

7.3. Results and analysis

Fig. 4 shows results under varying number of users in the network. Each point is the average result of the scenarios which fall in that category. It therefore includes results

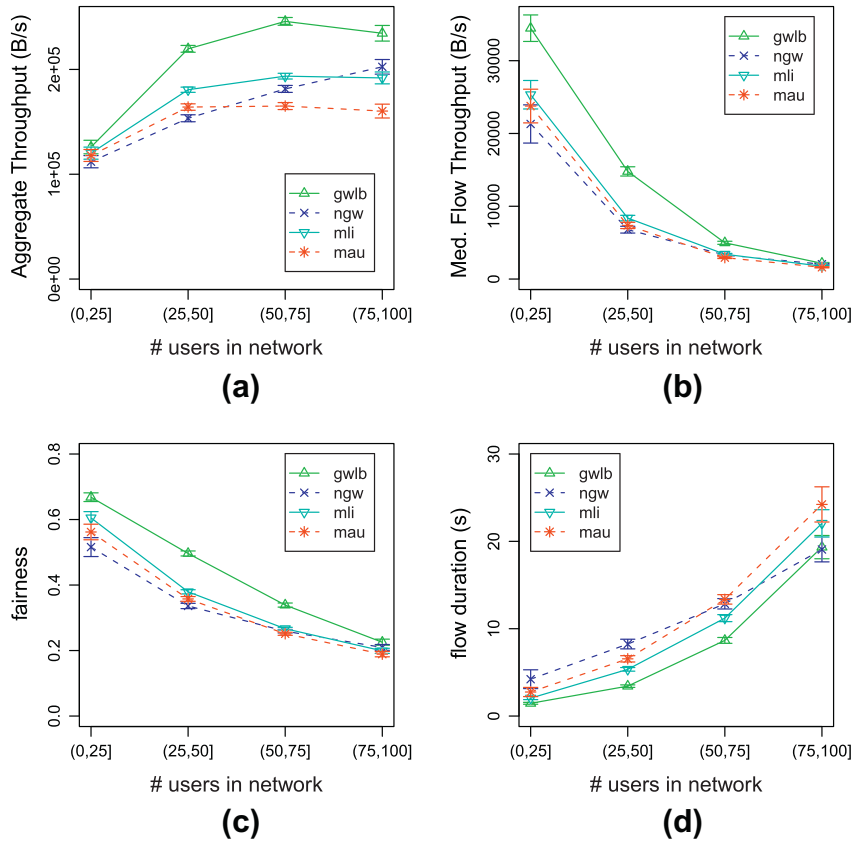


Fig. 4. Performance comparison of GWLB, NGW, MLI and MAU with varying number of users in the network.

from a 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. On average, MAU will perform similar to NGW, and MLI will perform better than both when the number of users is not too high, but degrading afterward.

As the number of users increases, load and number of flows increases, which leads to higher aggregate throughput (as long as there is sufficient capacity), and to lower average flow throughput. By balancing load, GWLB is able to maintain higher flow throughput, which also translates to higher aggregate throughput. When the number of users in the network is very large (more than 75) the native domains have a similar number of users, and GWLB defaults to the NGW solution, or a close-to-NGW solution. This also explains why the different protocols tend to converge on high load. The duration of flows follows a similar pattern. Note that congestion has a considerable impact on flow duration. Finally, we can also see that GWLB achieves better fairness between flows, by balancing load between gateways.

The results also show that MLI and MAU have inconsistent behavior. At best, MAU performs slightly better than NGW, and with high number of users performs worse. There are two main reasons for this: as explained earlier,

MAU uses congestion as a load indicator, which is not reliable because it is usually temporary when using TCP. In addition, the choice of sink to switch when congestion is detected 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.

As for MLI, its performance is good when the number of users is below 50 but starts to degrade afterward, to the point of performing worse than NGW. This indicates that MLI makes erroneous decisions. The main reason behind this is that it attempts to even the load but does not take into account that TCP flows are elastic, and as such, differences in load do not necessarily correlate with differences in number of flows.

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 128%. 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 (high path cost) which can increase contention and therefore prove detrimental.

Fig. 5 shows results under varying user imbalance between domains. This is measured as the standard deviation of the number of users in every native domain. 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 manifest when the imbalance is low. Only when the imbalance is high MLI distances itself from the other solutions. The performance of MAU is at best slightly better than NGW.

Increasing load imbalance progressively produces congestion in one or more domains, which leads to lower flow throughput and fairness and increased flow duration. Because GWLB balances load it is less affected by this. The performance of NGW is worse and decreases faster with increasing imbalance. Also, we can see that the difference in fairness between GWLB and NGW increases with load imbalance. By balancing load between gateways whenever possible and avoiding congestion, GWLB is more capable of maintaining inter-domain flow fairness.

7.4. GWLB adaptation frequency

The responsiveness of the protocol depends on the frequency with which it adapts to changes in network conditions. Because at any instant the controller has knowledge of the current set of flows, responsiveness depends on the frequency with which it recalculates flow-gateway associations, as explained in Section 3. In this subsection we evaluate the responsiveness of the protocol under highly dynamic traffic scenarios.

Tests are performed in one network topology with the same architecture and characteristics as described in Section 7.1. We generate scenarios with fixed number of active users in $\{5, 10, 15, 20, 25, 30\}$. Given a quantity of users, we generate 100 random scenarios, totaling 600 unique scenarios. In each of these unique scenarios with x active users, the set of x users which are active at a time changes every 10 seconds and is chosen at random. A user is randomly assigned to a sink and generates connections in the same manner as described in Section 7.1, with the only difference that the time between start of connections follows an exponential distribution with $\lambda = 1/5$ s (this is to generate connections within the user's 10 s interval). Note that this produces scenarios which are highly dynamic. Connections start at second 25 and end at 125. A lower

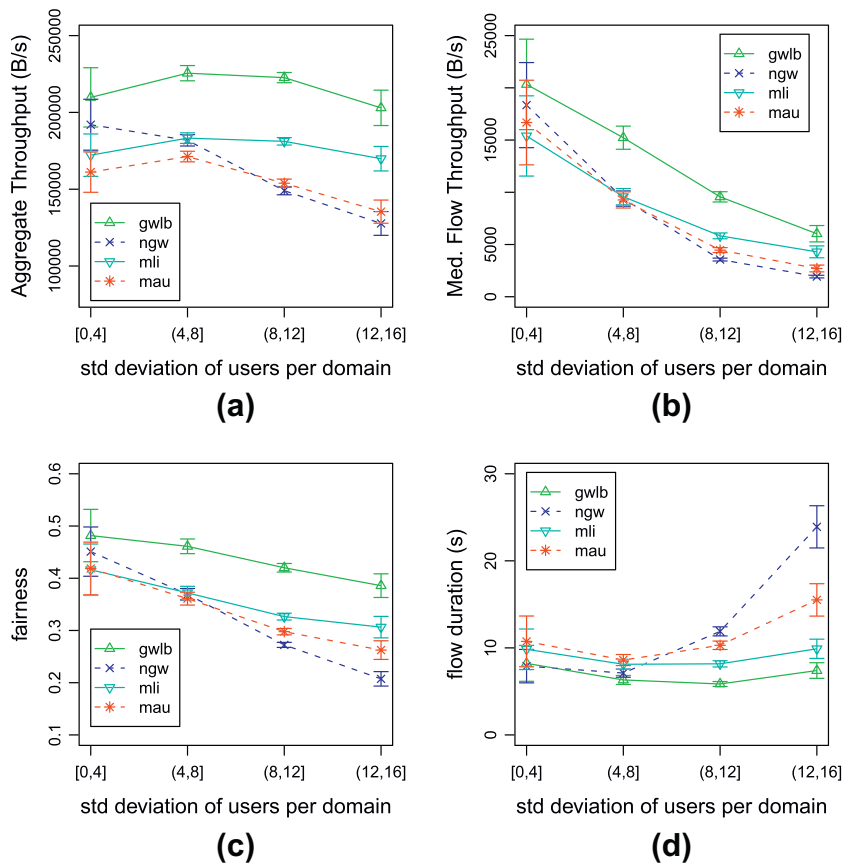


Fig. 5. Performance comparison of GWLB, NGW, MLI and MAU with varying user imbalance between native domains.

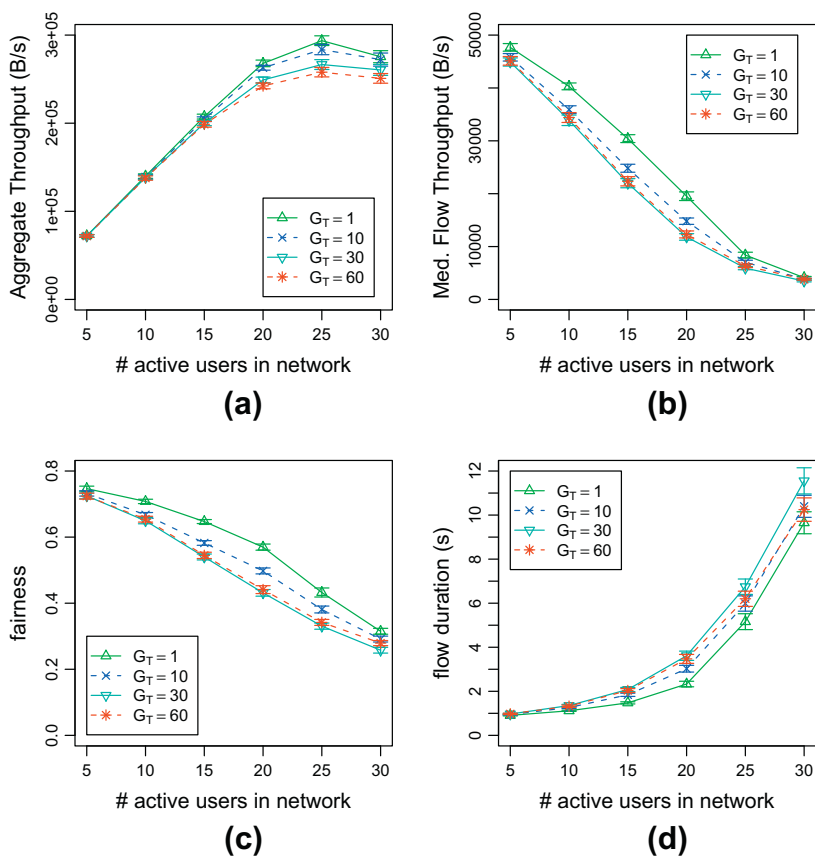


Fig. 6. Performance of GWLB in dynamic scenarios with varying adaptation frequency (determined by G_T).

period G_T enables the protocol to adapt more quickly to changing conditions. We vary G_T in $\{1, 10, 30, 60\}$ s. The other GWLB parameters remain as before.

The results are shown in Fig. 6. As we can see, the best performance is obtained with the highest frequency of adaptation, and the difference tends to increase with the number of users. When G_T is very low (one second), GWLB can respond quickly to changing conditions and is suitable for use in highly dynamic scenarios. Note that G_T can be as low as the time required to execute GSA.

8. 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. The limited wireless link capacity aggravates the problem, turning gateways into bottlenecks, which can easily lead to congestion. Moreover, the roaming of end-systems across the network together with variable radio conditions contribute to make demands more dynamic over time.

In this paper we have proposed GWLB, a highly responsive online protocol which dynamically adapts to network conditions, balancing the load of gateways. It achieves improvements over shortest path routing in both

throughput and fairness in scenarios with load imbalance. Importantly, because it is TCP flow-aware, balances traffic at the TCP flow level, and takes into account the effects of interference of flows when switching between domains, it is suitable for implementation in realistic scenarios. The simulations conducted in *ns-3* 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 128% 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.

References

- [1] A. Raniwala, 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.
- [2] D.S.J.D. Couto, D. Aguayo, J. Bicket, 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. doi:http://doi.acm.org/10.1145/938985.939000.
- [3] R. Draves, J. Padhye, B. Zill, Routing in multi-radio, multi-hop wireless mesh networks, in: MobiCom '04: Proceedings of Conference on Mobile Computing and Networking, ACM, 2004, pp. 114–128. doi:http://doi.acm.org/10.1145/1023720.1023732.

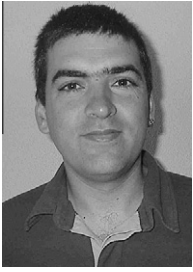
- [4] Y. Yang, J. Wang, R. Kravets, Interference-aware Load Balancing for Multihop Wireless Networks, Tech. rep., University of Illinois at Urbana-Champaign, 2005.
- [5] A. Raniwala, K. Gopalan, T.-c. Chiueh, Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks, SIGMOBILE Mob. Comput. Commun. Rev. 8 (2) (2004) 50–65. doi:<http://doi.acm.org/10.1145/997122.997130>.
- [6] M. Alicherry, R. Bhatia, L.E. Li, Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks, in: MobiCom '05: Proceedings of the 11th Annual International Conference on Mobile Computing and Networking, ACM, New York, NY, USA, 2005, pp. 58–72. doi:<http://doi.acm.org/10.1145/1080829.1080836>.
- [7] J. Tang, G. Xue, W. Zhang, Maximum throughput and fair bandwidth allocation in multi-channel wireless mesh networks, in: INFOCOM 2006: 25th IEEE International Conference on Computer Communications, IEEE Computer Society, 2006, doi:[10.1109/INFOCOM.2006.249](http://doi.acm.org/10.1109/INFOCOM.2006.249).
- [8] Y. Bejerano, S.-J. Han, A. Kumar, Efficient load-balancing routing for wireless mesh networks, Comput. Netw. 51 (10) (2007).
- [9] J.M. Kleinberg, Single-source unsplitable flow, in: FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, USA, 1996, p. 68.
- [10] H. Tokito, M. Sasabe, G. Hasegawa, 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] V. Mhatre, H. Lundgren, F. Baccelli, C. Diot, Joint MAC-aware routing and load balancing in mesh networks, in: Proceedings of the 2007 ACM CoNEXT Conference, 2007.
- [12] C.-F. Huang, H.-W. Lee, Y.-C. Tseng, A two-tier heterogeneous mobile ad hoc network architecture and its load-balance routing problem, Mob. Netw. Appl. 9 (4) (2004).
- [13] B. Xie, Y. Yu, A. Kumar, 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.
- [14] L. Ma, 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.
- [15] K. Ramachandran, M. Buddhikot, G. Chandranmenon, S. Miller, E. Belding-Royer, K. Almeroth, On the design and implementation of infrastructure mesh networks, in: Proceedings of the IEEE Workshop on Wireless Mesh Networks (WiMesh), 2005.
- [16] D. Nandiraju, L. Santhanam, N. Nandiraju, 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.
- [17] S. Maurina, R. Riggio, T. Rasheed, 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.
- [18] S. Lakshmanan, R. Sivakumar, K. Sundaresan, Multi-gateway association in wireless mesh networks, Ad Hoc Netw. 7 (3) (2009) 622–637.
- [19] M. Ito, T. Shikama, 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.
- [20] J.J. Galvez, P.M. Ruiz, 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.
- [21] J. Kleinberg, E. Tardos, Y. Rabani, Fairness in routing and load balancing, in: FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, USA, 1999, p. 568.
- [22] P. Gupta, P.R. Kumar, The capacity of wireless networks, IEEE Trans. Inform. Theory (2000) 388–404.
- [23] J. Li, C. Blake, D.S.J.D. Couto, H.I. Lee, R. Morris, Capacity of ad hoc wireless networks, in: MobiCom '01: Proceedings of 7th ACM International Conference on Mobile Computing and Networking, 2001.
- [24] B. Claise, Cisco Systems NetFlow Services Export Version 9, RFC 3954, October 2004. <<http://www.ietf.org/rfc/rfc3954.txt>>.
- [25] B. Claise, Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information, RFC 5101, January 2008. <<http://www.ietf.org/rfc/rfc5101.txt>>.
- [26] J.J. Galvez, P.M. Ruiz, A.F. Gomez-Skarmeta, Multipath routing with spatial separation in wireless multi-hop networks without location information, Comput. Netw. 55 (3) (2011) 583–599. doi:<http://dx.doi.org/10.1016/j.comnet.2010.09.016>.
- [27] M. Gairing, T. Lücking, M. Mavronicolas, B. Monien, Computing nash equilibria for scheduling on restricted parallel links, Theor. Comput. Syst. 47 (2) (2010) 405–432. doi:<http://dx.doi.org/10.1007/s00224-009-9191-9>.
- [28] O.H. Ibarra, C.E. Kim, Heuristic algorithms for scheduling independent tasks on nonidentical processors, J. ACM 24 (2) (1977) 280–289. <<http://doi.acm.org/10.1145/322003.322011>>.
- [29] J.J. Galvez, P.M. Ruiz, A.F. Gomez-Skarmeta, Heuristics for Scheduling on Restricted Identical Machines, Tech. rep., University of Murcia, Spain, TR-DIIC 1/2010, 2010. <<http://ants.dif.um.es/staff/pedrom/papers/rs-v1-1.pdf>>.
- [30] F. Kelly, Charging and rate control for elastic traffic, Eur. Trans. Telecommun. 8 (1997) 33–37.
- [31] Y. Xue, B. Li, K. Nahrstedt, Optimal resource allocation in wireless ad hoc networks: a price-based approach, IEEE Trans. Mob. Comput. 5 (4) (2006) 347–364. doi:<http://doi.ieeecomputersociety.org/10.1109/TMC.2006.52>.
- [32] P. Belotti, J. Lee, L. Liberti, F. Margot, A. Wächter, Branching and bounds tightening techniques for non-convex MINLP, Optimiz. Methods Software 24 (4–5) (2009) 597–634.
- [33] Couenne, an exact solver for nonconvex MINLPs, 2010. <<https://projects.coin-or.org/Couenne>>.
- [34] The ns-3 network simulator. <<http://www.nsnam.org/>>.



Juan J. Galvez received his B.Sc. (2004) and M.Sc. (2007) degrees in Computer Science from the University of Murcia, Spain. He is a Ph.D. student in Telematics at the Department of Information and Communication Engineering (DIIC) at the University of Murcia (UMU). His main research interests include mobile and ad hoc wireless networks, wireless mesh networks, distributed algorithms and traffic engineering.



Pedro M. Ruiz received his B.Sc. (1999) and M.Sc. (2001) and Ph.D. (2002) degrees in Computer Science from the University of Murcia, Spain. He works as Associate Professor in Telematics at the Department of Information and Communication Engineering (DIIC) at the University of Murcia (UMU). In 2003 he was awarded a *Ramón y Cajal* research position by the Spanish MEC. He has also held Post-doctoral research positions at ICSI in Berkeley, King's College London and University of California at Santa Cruz. During these years he has acted as Principal Investigator in a number of research projects mainly funded by the European Union, Spanish government and private companies, and has published a large number of refereed papers in international journals and conferences. He has received in 2007 an outstanding research trajectory recognition from the Spanish MEC. He is in the editorial board of the International Journal on Parallel, Emergent, and Distributed Systems, International Journal of Network Management and International Journal on Smart Home. He has organized several workshops on localized algorithms and protocols co-located with IEEE MASS, ACM MobiHoc and IEEE DCOSS. He has also served in a number of Organizing and Technical Committees of highly relevant conferences such as ACM MobiCom, IEEE MASS, ACM MobiHoc, IEEE SECON, etc. He also serves as a reviewer for major IEEE journals and conferences. His main research interests include vehicular networks (VANET), sensor networks, mobile and ad hoc wireless networks and distributed systems. He is a member of the ACM and IEEE Communications Society.



Antonio F. Gómez-Skarmeta/1965 received the M.S. degree in Computer Science from the University of Granada and B.S. (Hons.) and the Ph.D. degrees in Computer Science from the University of Murcia, Spain. Since 1993 he is Professor at the same department and University. Antonio F. Gómez-Skarmeta has worked on different research projects in the national and international area. He is associate editor of the IEEE SMC-Part B and reviewer of several international journals. He has published over 90 international papers and is member of several program committees.