GateSelect: A novel Internet gateway selection algorithm for client nodes

Khulan Batbayar*†, Roc Meseguer*, Ramin Sadre†, Suresh Subramaniam‡
* Universitat Politècnica de Catalunya, Spain
† Université Catholique de Louvain, Belgium
‡ The George Washington University, United States
Email:batbayar@ac.upc.edu, meseguer@ac.upc.edu, ramin.sadre@uclouvain.be, suresh@gwu.edu

***Abstract*—The Internet gateway selection problem is becoming very important as the number of Internet-connected devices increases and stresses the limited number of Internet gateway nodes. The gateway nodes often experience frequent performance fluctuations, and the best gateway selection candidate changes frequently with growing network dynamics. We propose GateSelect, a customized selection algorithm for each client node that not only provides the best-effort selection candidate but also ensures the global, balanced distribution of the gateway nodes. We utilize over the counter, lightweight calculations to optimize the client-side selection algorithm by combining classification, short term performance prediction, and randomized selection. We compare our algorithm with several baseline algorithms, and the experiment results show that our proposal provides better performance and balanced distribution of gateway nodes.**

***Index Terms*—Internet gateway selection, performance based classification, fair resource allocation, client side decision making**

I. Introduction

With the advent of the IoT, there are whole new types of heterogeneous devices introduced to the Internet ecosystem, including smart home devices, sensor networks, and wearable devices. The IoTAnalytics research [1] published in 2018 estimated that the number of active Internet-connected devices, including IoT devices, will reach over 20 billion in 2020. Each of these connected device tries to send or receive information through the Internet. Providing network connectivity to them becomes a challenging task. Especially, the presence of mobile devices that might join or leave the network at any time makes the planning of the network difficult.

To cope with this, connectivity through large-scale wireless networks has been proposed. Such networks typically contain a larger number of client nodes and one or more gateway nodes [2]. The latter are responsible for providing connectivity to the Internet. Therefore, all the traffic directed to/from the Internet passes through those gateway nodes, making them susceptible to congestion, failure, and overcrowding [3]. In large networks, these problems are further aggravated by the fact that the gateways might be heterogeneous with different capacities to handle client traffic. Furthermore, the performance of a gateway will depend on the number of clients that send their traffic through that gateway. As the demand for the gateway node increases, the Internet connectivity perceived at the client nodes can become excruciatingly slow. Measurements have shown that the performance of a gateway will vary dynamically depending on the number of active clients at a time, rush hours, etc. [2], [3]. Consequently, choosing the "right" gateway can improve the Quality of Service perceived at the client node.

The process of choosing a gateway for a client node among the available gateways in a network is called *gateway selection*. State-of-the-art algorithms for gateway selection are often based on monitoring the performance or state of the available gateways to help client nodes make an informed decision. To keep up with the network dynamics, many monitoring algorithms use an active monitoring approach where nodes periodically probe the available gateways. The algorithms aim to provide the clients with an accurate, up-to-date view on the gateway performance to enable them to select the best performing [3]–[6], the least-loaded [7]–[10] or the nearest gateway [11], [12].

The idea behind the above algorithms is that client nodes should aim to select the gateway node that is the best for them, be it in terms of performance, load, or distance. However, from a network point of view, this is not efficient. Selecting the best gateway means that similar or "nearly as good" gateways are ignored, potentially resulting in load imbalances and frequent changes in the network.

In this paper, we introduce *GateSelect*, a client-side distributed gateway selection method to tailor the gateway selection for each client node by utilizing the measurements collected locally at their side. The algorithm categorizes available gateway nodes down to the potential good candidates based on their ability to provide good, stable performance for the future and then selects the gateway from the candidate list with minimal selection bias. We compare the proposed method with a greedy gateway selection, random (load-balanced) gateway selection, and static gateway selection in different network scenarios. The experimental results show that our method results in long term stable Internet connections and balances the overall gateway allocation.

The remainder of the paper is organized as follows. Section II introduces the problem statement and the requirements for a good gateway selection. Section III introduces our gateway selection algorithm. Section IV presents the experimental results. Related work is reviewed in Section V, and the paper is concluded in Section VI.

II. Background

In the type of networks we consider in this paper, client nodes connect to the Internet through gateway nodes. For cost or security reasons, the number of gateway nodes is typically much smaller than the number of client nodes. Use cases for such networks are large IoT installations or open community-driven networks like guifi.net [13]. In such scenarios, it is not desired to have a fixed assignment of client nodes to specific gateways. Instead, a client node should be dynamically assigned to a gateway depending on network conditions and gateway loads. The goal of the gateway selection is to map the client nodes to those gateway nodes that provide long-term Internet connectivity with good performance. In this context, *good* performance means that the gateway provides better long-term performance than the average at a given time. Otherwise, its performance is called *bad*. Depending on the goals of the users or the network operator, the performance can be defined in different ways, for example in terms of latency, achievable throughput, etc. The instantaneous performance of a gateway as perceived by a client node can vary over time; it not only depends on the capacity and load of the gateway itself but also on external and internal factors such as the quality of the connections of the gateway to the Internet or the network condition inside the client network. A client node's gateway selection is done with the intention of choosing a good gateway, but the following problems occur when each client selects its locally optimal gateway.

1) Herd behavior: Selection algorithms where a client selects its locally optimal gateway are considered selfish since they ignore the impact the selection might have on the other clients. Several such algorithms have been proposed in the literature. Performance-oriented selection algorithms select the gateway with the currently best performance [3]–[5]. Load-oriented algorithms choose the gateway with the lowest load [7], [14], [15]. Between those two extremes, are hybrid methods [8], [16] that combine both metrics. The behavior of these three types of selection algorithms is shown in Figure 1. In this example, three gateway nodes $G1$, $G2$, $G3$ have a performance metric (e.g., latency, throughput, etc.) and a load metric (e.g. total traffic at the gateway, CPU load, etc.). These metrics are independently measured by the client nodes $C1$ through $C5$, i.e., each client maintains its own view on the state of the gateways. This is necessary because the clients might perceive the performance offered by the gateways differently, depending on their location in the network.

With performance-oriented selection, all nodes select $G1$, whereas the nearly equally well performing $G2$ is selected only by $C4$. $G3$ is not selected by any client node although it is less loaded. With load-oriented selection, the client nodes try to select the least loaded gateway node. In this case, the clients select $G3$ even though its performance is the lowest. Let's consider a hybrid gateway selection that gives equal weight to both metrics. In this case, the majority of the client nodes select $G3$. The main problem in choosing the locally optimal gateway at the time of selection is that there are other nodes

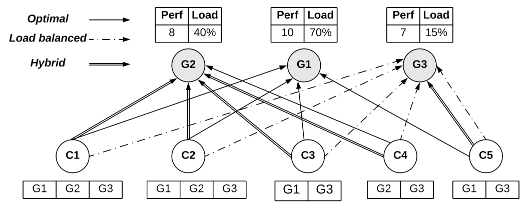


Fig. 1. Gateway selection example scenario.

which choose the same gateway node. This situation is called herd behavior. The result is that the selected gateway becomes overloaded and its performance degrades.

2) Frequent gateway selection changes: Typically, the decision of a gateway selection algorithm is not static. Many algorithms proposed in the literature are based on a continuous active monitoring of the gateways, where the selection is periodically reevaluated based on the measurement results. Such periodic reevaluations are called *rounds* in the following. In large-scale networks, the gateway performance as perceived by the clients will vary over time due to changes in network conditions, the appearance or disappearance of client nodes, or simply the activity of the client nodes. For a selection algorithm aiming at finding the locally optimal gateway, this would mean to change the selected gateway in every round. Not only would this create an imbalance in the gateway allocation, it would also force each client node to repeatedly find a new route to its selected gateway. As a result, locally, each node experiences short-term performance degradation while waiting for the new route, and globally, within the client network, additional management traffic is generated due to frequent rerouting and mass migration.

3) Instability of gateway performance: Here, stability means that the gateway selected by a client node should exhibit good performance over time. Heterogeneous networks contain gateways with different capacities, some capable of handling a large number of client nodes, others only a few. Gateways with limited capacity might show good performance at a time, but a slight increase in demand, e.g., due to new clients joining or existing clients increasing their network activity, would cause a performance drop. The current performance and load of a gateway therefore don't necessarily indicate its future performance.

In this paper, the goal of gateway selection is to select the gateway that provides a good Internet connection. We focus on the following requirements:

1) **Individual selection decision** - Clients perform their selection without coordinating the selection decision with the other nodes or gateways.
2) **Stable gateway selection** - Unless the performance of the selected gateway deteriorates too much, a change of the selection should be avoided. Client nodes should give gateways with similarly good performance a chance to get selected.

3) **Load balancing** - A node's decision should not negatively influence the overall stability of the gateway distribution as well as the performance perceived by the other client nodes.
4) **Minimum knowledge** - Clients do not have knowledge of the network topology or of the specifications of the gateways. In particular, the latter means that the clients do not know how many clients a specific gateway can handle nor can they directly query its current load.

III. The GateSelect algorithm

A. Preliminaries

	G_1	G_2	G_3	G_4	$\dots$	G_{M-1}	G_M
t_1		0.8	0.6		$\dots$	0.9	
t_2	0.7			1.3	...	0.8	0.4
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	
t_K	0.4	0.8		0.2	$\dots$		0.5

TABLE I
PARTIAL GATEWAY PERFORMANCE TABLE

We consider a large-scale network with client nodes $C = \{C_1, C_2, \dots, C_n\}$ sharing Internet connectivity through a much smaller number of Internet gateway nodes $G = \{G_1, G_2, \dots, G_m\}$, i.e., $n >> m$. Client nodes are oblivious about the underlying network topology, routing to the gateways, overall network load, etc. To monitor the state of the gateways, the client nodes periodically measure their performance through active monitoring. As a result, each client maintains a gateway performance table as shown in Table I where M_{ij} is the result of the performance measurement of G_j at round t_i. In our experiments in the next sections, M_{ij} is the time needed to download a file of a specific size from the Internet through the gateway node. Every round t, the node performs a gateway selection.

It is important to note that, as visible in Table I, measurements are not complete, i.e., a client node does not measure every gateway in every round. Doing so would result in an unacceptable overhead in the network. Instead, different strategies can be used to reduce the number of measurements, such as random sampling or the usage of dedicated measurement nodes that perform measurements and distribute the results to the other nodes. In previous papers [3], [17], we proposed a collaborative scheme where each client node would measure the performance of a randomly selected subset of gateways and share the results with its neighbor nodes, assuming that close neighbors have a similar perception of the gateway performance. As a result, the list of gateways to choose from varies in each round.

The aim of GateSelect is to find a gateway for each client node that provides good performance for a long time. We call such a gateway a *stable gateway*. The proposed algorithm's structure is shown in Figure 2. Its main advantage is that the selection is performed locally by each node, without specific knowledge of the network topology or the decisions of the other nodes. The algorithm runs periodically and consists of 3 steps:

1) Build an initial list of candidate gateways by classifying them based on their performance;

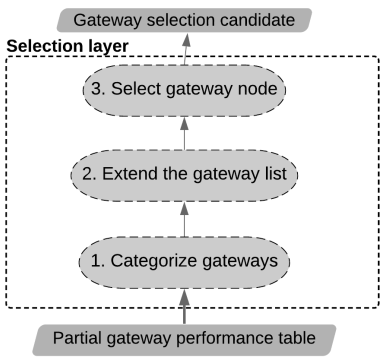

Gateway selection candidate
Selection layer
3. Select gateway node
2. Extend the gateway list
1. Categorize gateways
Partial gateway performance table

Fig. 2. Selection algorithm design.

Performance	**Deviation**	**Perf. category**
$M_{tj} \leq \mu$	$stdev(M_j, k) \leq stdev(M)$	Good
$M_{tj} \leq \mu$	$stdev(M_j, k) > stdev(M)$	Inconsistent
$M_{tj} > \mu$	$stdev(M_j, k) \leq stdev(M)$	Inconsistent
$M_{tj} > \mu$	$stdev(M_j, k) > stdev(M)$	Bad

TABLE II
GATEWAY PERFORMANCE CATEGORIES

2) Increase the number of candidates by filling the gaps in the partial performance table;
3) Select a gateway with the goal of obtaining (a) a good Internet connection for the client and (b) a balanced gateway distribution, i.e., a nearly equal number of nodes per good performing gateway.

B. Step 1: Classification

This step's goal is to identify gateways that consistently provide good performance. To this end, we extend our previous work in [17]. A client node first assigns a *performance category* to gateway node G_j based on its current performance M_{tj} and the standard deviation $stdev(M_j, k)$ of its performance in the last k rounds ($k = 10$ by default), following the rules in Table II. The thresholds μ and $stdev(M)$ are the average performance over all gateways at round t_i and the standard deviations of all gateway performances within the last k rounds, respectively. The classification is done every time a node receives a new measurement M_{ij}. For example, if the current performance of the gateway is lower than the threshold and the current gateway standard deviation is lower than the threshold standard deviation, the gateway performance is categorized as *Good*.

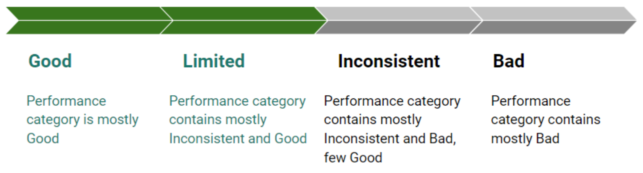

Good
Performance category is mostly Good
Limited
Performance category contains mostly Inconsistent and Good
Inconsistent
Performance category contains mostly Inconsistent and Bad, few Good
Bad
Performance category contains mostly Bad

Fig. 3. Estimated gateway capacity classification

As an extension to [17], we propose to further classify the gateways according to the evolution of their performance category over time. Again, we use a time window of k rounds. The possible classes are shown in Figure 3. If a gateway's performance is categorized as *Good* in the majority of the last k rounds, it is classified as a *Good* gateway and assumed to provide stable and good performance in the future. If it is categorized mostly as *Inconsistent* with a few cases of *Good*, then the algorithm classifies it as a gateway of *Limited* capacity. This is motivated by the idea that gateways with high capacity are less sensitive to performance fluctuations. Similar reasoning is used to classify gateways as *Inconsistent* and *Bad*. *Inconsistent* and *Bad* gateways are not considered as suitable candidates for the gateway selection.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
t-10	M(t-10,1)	M(t-10,2)	M(t-10,3)		M(t-10,5)		M(t-10,7)	M(t-10,8)	M(t-10,9)	M(t-10,10)
...	...	...	...	...	...	...	...	...	...	...
t-3		M(t-3,2)	M(t-3,3)	M(t-3,4)	M(t-3,5)	M(t-3,6)	M(t-3,7)	M(t-3,8)		
t-2	M(t-2, 1)	M(t-2, 2)	M(t-2,3)	M(t-2,4)	M(t-2,5)	M(t-2,6)	M(t-2,7)	M(t-2,8)	M(t-2,9)	M(t-2,10)
t-1	M(t-1,1)	M(t-1,2)	M(t-1,3)	M(t-1,4)	M(t-1,5)	M(t-1,6)	M(t-1,7)		M(t-1,9)	M(t-1,10)
t	M(t,1)	M(t,2)	?	?	M(t,5)	M(t,6)	M(t,7)	M(t,8)	?	M(t,10)

TABLE III
EXAMPLE GATEWAY PERFORMANCE TABLE AFTER STEP 1

Table III shows a possible outcome of the classification step. In the table, the entries with *Inconsistent* and *Bad* classification are shown with a grey background. In the latest round t, the gateways $G3$, $G4$ and $G9$ are not classified due to missing measurements.

C. Step 2: Extend the candidate list

Step 1 reduces the number of gateways to a small list of *Good* and *Limited* candidates. To extend the list of candidates, we propose a performance prediction step to fill the missing entries of the current round in the performance table. The prediction uses the last k measurements. Since we are only interested in *Good* and *Limited* gateways, we only perform the prediction for gateways with a missing measurement in the current round that were classified as *Good* or *Limited* in the previous round. We then repeat step 1 for those gateways. In our example (Table III), this is done for $G3$ and $G4$ and increases the number of candidates from 4 to 6.

Since we only perform this step on gateways that have been classified as relatively stable in the previous round, we can expect that the prediction provides reasonably good estimations of the gateways' current performance. Figure 4 shows empirical measurements for three gateways from the real large-scale community network guifi.net [13]. Measurements are performed every 2 minutes. The top plot shows the timeseries of the latency of downloading a file of 0.1MB from the Internet through the gateways. The bottom plot shows the ECDF of the standard deviation of the latency for $k = 10$. Gateways $gw1$ and $gw2$ are stable, and we consider $gw3$ an unstable gateway as its performance varies significantly over time.

We use Linear Regression (LR) with $y = \beta_0 + \beta_1 X + \epsilon$ to predict the performance y of a gateway from its past k measurements X. β_0 is the intercept term, β_1 is the slope of the line, and ϵ is the error.

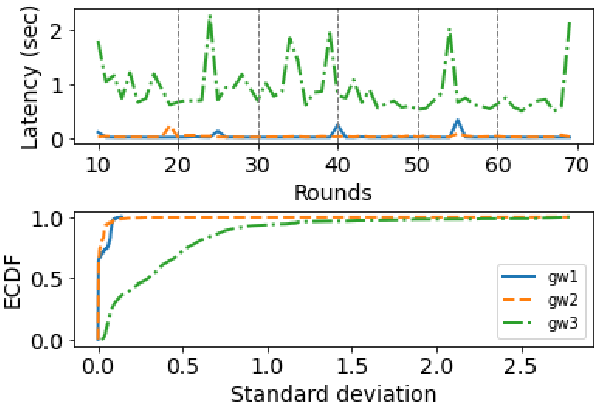


Fig. 4. Gateway latency variance.

D. Step 3: Gateway node selection

Every node wants to achieve the best Internet performance through the gateway nodes, so it's natural to select the best performing gateway. As explained in Section II, this is not always desirable. To limit this selfish behavior, we propose to restrict the node's ability to select the best gateway. After Steps 1 and 2, the gateway selection candidates have been narrowed down to *Good* and *Limited* nodes. Therefore, any of these gateways is able to provide good performance for the node.

Algorithm 1 shows how the gateway is selected from the list of candidates. The algorithm doesn't change the gateway if the currently selected gateway of the client node is among the candidates (lines 2-3). In this way, frequent changes are avoided unless the current gateway's performance deteriorates such that it is eliminated from the list of candidates.

Algorithm 1 Gateway selection

```
Require: $gateway\_candidates$
Require: $current\_gateway$
Require: $D = 2$

1: procedure SELECTGATEWAY
2:     if $current\_gateway \in gateway\_candidates$ then
3:         return $current\_gateway$
4:     $good\_gateways = \{G \in gateways\_candidates \mid G$ is $Good\}$
5:     if SIZE($good\_gateways$) $< D$ then
6:         $good\_gateways = gateway\_candidates$
7:     $candidates$ = RANDOM($good\_gateways, D$)
8:     $current\_gateway$ = BEST($candidates$)
9:     return $current\_gateway$
```

As nodes are oblivious to the decisions of their peers, it is difficult to ensure a fair distribution of the clients over the good gateways. We incorporate a randomized load balancing to reduce the selection bias. The algorithm randomly samples D candidates from the list (line 7) and selects the best performing one (line 8). By default, the random sample size is $D = 2$ by using the Power of Two Choices (PoTC) algorithm [18]. Many network load balancing [19]–[21] and job scheduling [22] algorithms use PoTC for balanced resource distribution. The *Good* gateways are given priority for the selection (line 4). However, if the number of good gateways is too small (line 5), the algorithm also considers the *Limited* gateways (line 6).

IV. Experimental validation

A. Experimental setup

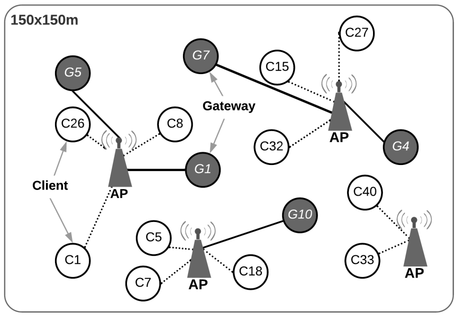


Fig. 5. Experiment network topology.

We test the GateSelect algorithm in a simulated network with 40 client nodes and 10 gateway nodes. We test three different client network configurations: wired, wireless and wireless mobile network. Gateway performance is measured every round by recording the time clients need to download a 0.1MB file through the gateways from an HTTP Server connected to the gateways through wired links. The client nodes also download a 1MB file through their selected gateways every minute to mimic client activities.

In the *wired* configuration, we use MiniNet [23] to simulate a wired client network where every 10 nodes connect to one switch.Random link delays between 1ms and 3ms (chosen at the beginning of each simulation run) are set for all links between nodes and switches and a fixed 5ms delay is set for the links between two switches. In addition, gateways add a randomly delay between 1ms and 3ms to each response.

For the *wireless* configuration, we use Mininet-Wifi [24] to simulate a wireless IEEE 802.11 network in a 150m$\times$150m area with the network topology shown in Figure 5. We place the client nodes at random positions together with 4 access points (AP) at fixed locations. The gateways are connected to the APs by a wired link with a delay between 1ms to 3ms (chosen at the beginning of the simulation run). Again, a random delay of 1ms to 3ms is added dynamically when a client downloads a file. The wireless interference model is provided by `mac80211_hwsim` in Mininet-Wifi. The transmission range is 250m.

Finally, we also simulate a *wireless mobile* configuration that is identical to the above wireless configuration but client nodes move according to a Random Walk mobility model with 1m/s and connect to the nearest access point.

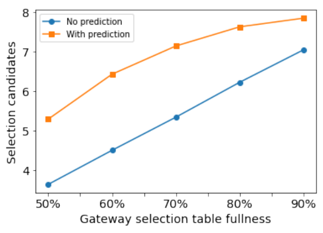
We compare four different selection algorithms:

1) GateSelect: Our algorithm;
2) Greedy (Brute-Force) selection: Clients always select the best gateway every round;
3) Random selection (Power of 2 choices): Clients choose two random gateways and select the best every round;
4) Static selection: Clients select the best gateway at the beginning of the experiment and never change the selection unless the gateway becomes unresponsive.

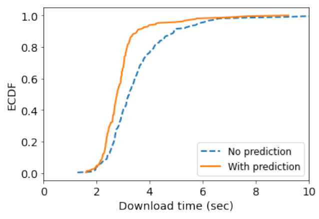
Experiments are repeated 5 times with different random positions and delays, and each experiment runs for 100 selection rounds, where a selection round has a duration of two minutes.

B. Prediction of missing values

In our first experiment, we test the impact of Step 2 of the GateSelect algorithm on the size of the candidate list. To this end, we run Step 1 with gateway selection tables filled to 50%, 60%, 70%, 80%, and 90% with gateway measurements at any selection round. Figure 6a shows the average resulting number of candidates at a node without (i.e., after Step 1) and with (i.e., after Step 2) predictions for the wireless network. Without prediction, the number of candidates shrinks down to 3-4 candidates on average for gateway performance tables filled to 50%. The number of candidates increases with the degree of filling. When the prediction step is added, the number of candidates increases by 10 to 20%, depending on the degree of filling.



(a) Prediction selection candidates



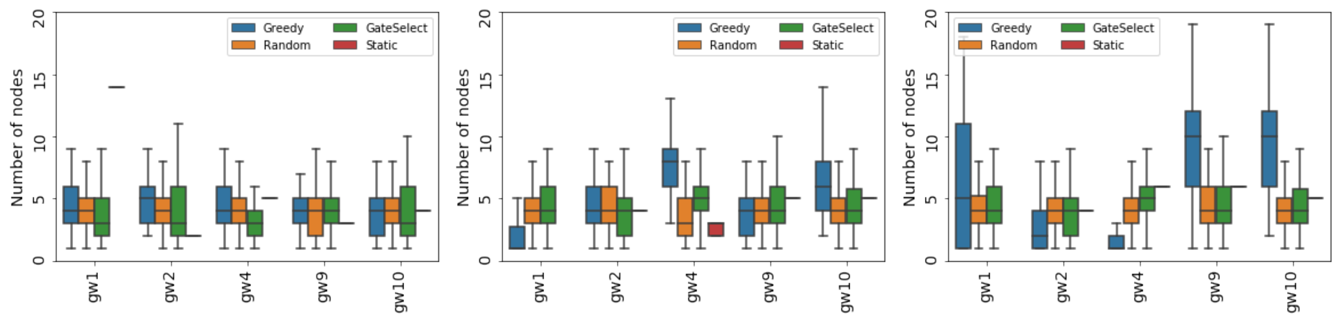
(b) Prediction download latency

Fig. 6. Wireless - Performance prediction result.

We quantify the impact of the prediction on the overall system performance by measuring how much time the clients need to download the 1MB file through their selected gateways. Figure 6b shows the results when we run the algorithm with and without the prediction step (Step 2) for the same network settings. We observe that the download time improves with prediction because the prediction step increases the number of selection candidates for each node. As a result, the traffic load is distributed more evenly over the gateways.

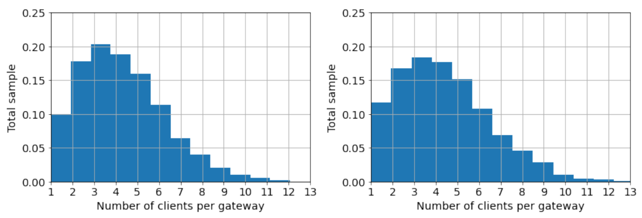
C. Gateway selection distribution

We compare the distribution of the client node over the gateways for the different selection algorithms. To this end, we calculate the average distribution of the clients after a single simulation (100 rounds). For all three network configurations,



(a) Wired network (b) Wireless network (c) Wireless mobile

Fig. 7. Client node distribution over the gateway nodes.



(a) GateSelect - Wireless (b) GateSelect - Mobile

Fig. 8. Wireless gateway distribution

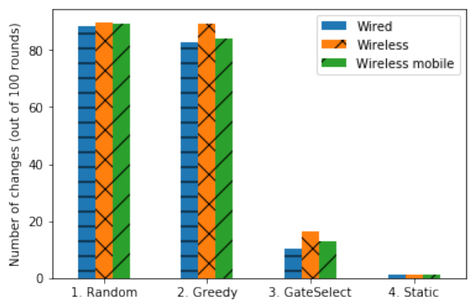


Fig. 9. Number of gateway selection changes in 100 rounds.

identical start positions were chosen for the nodes. Figure 7a shows the obtained distribution for the wired network. The gateway distribution is similar for GateSelect and the Greedy and Random selection algorithms. The 40 client nodes are more or less evenly distributed over the ten gateways, i.e., four clients per gateway. This is because in the wired network, the impact of the variances of the link delays is very small, therefore all gateways have an equal chance of being selected. Only in static selection, we see an unequal distribution due to random effects when the clients chose their gateway at the beginning of the simulation.

For the wireless network without mobility (Figure 7b), Greedy selection shows a more uneven distribution with higher variances for several gateways than the other algorithms due to the herd behavior (Section II). These variances are increased in the wireless network with mobility (Figure 7c) where the movements of the client nodes cause the Greedy algorithm to frequently change the selection. In general, GateSelect behaves similar to Random selection, which is expected since both algorithms use Power of 2 choices to achieve load balancing.

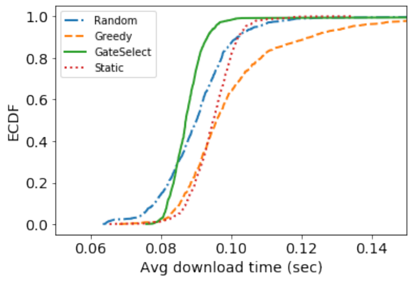
Figure 8 shows the overall distribution of client nodes per gateway per round in the wireless and wireless mobile network. The simulation runs are repeated five times with random positions of nodes. The results show that both networks behave similarly. In the majority of the rounds, the number of clients per gateway is between 2 and 5, which corresponds to a relatively balanced distribution of the clients over the gateways. At the beginning of each simulation run, network nodes only have a a limited number of gateway performance measurements, therefore, there is a small fraction of rounds with a higher number of clients for some gateways.

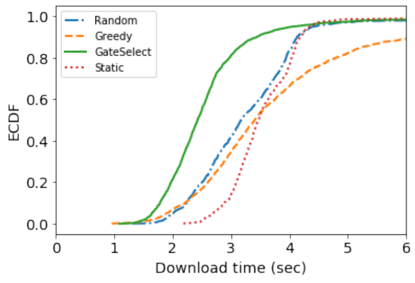
D. Gateway selection changes

Figure 9 shows the average number of gateway selection changes over 100 rounds for the different selection algorithms. The simulations were run five times for each network and selection combination with different random positions and link delays. In all three network configurations, Greedy and Random selection change the selected gateway significantly more frequently than our approach since they always try to find the best gateway among all gateways (Greedy) or among two randomly chosen gateways (Random). GateSelect reduces the number of changes by a factor of four. The static selection performs, by design, the smallest number of selections. It only changes the selected gateway if it becomes unresponsive. In the wired network, gateway changes are slightly less frequent because the links are more stable. The wireless network is noisier, thus, more selection changes occur. The wireless mobile network shows less changes because the gateways are within the access range of every node and depending on the mobility direction the performance of the selected gateway stays stable for some nodes.

E. Gateway selection performance

We measure the effectiveness of the gateway selection by how long it takes a client node on average to load content (1MB file) through the selected gateway. The results after five simulation runs are shown in Figure 10.


(a) In a wired network


(b) In a wireless network

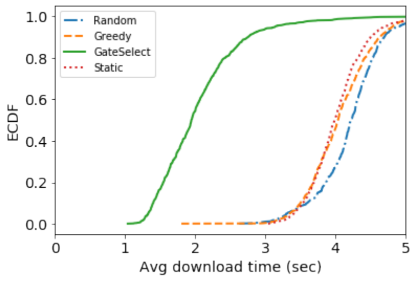

(c) In a wireless mobile network

Fig. 10. Download time through the gateway nodes.

In the wired network (Figure 10a), the download time of GateSelect is less than 0.09 seconds on average, and is better than that of the other algorithms for 90% of the downloads. However, other algorithms provide a performance close to our proposal due to the stability of the links in the wired network. In wireless networks without mobility (Figure 10b), GateSelect offers a significantly better download time for the majority of the downloads compared to the other algorithms. The Greedy algorithm's download time is the worst due to the herd behavior which results in overloaded gateways. The advantage of our approach is even clearer in wireless networks with mobility (Figure 10c). The download time stays below 2 seconds in 80% of the downloads, because of fewer selection changes and mobile nodes don't have to find a new route to the new gateway. The other algorithms show download times between 4 to 5 seconds.

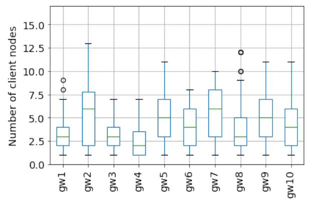
Overall, GateSelect shows better download times than the others by combining a conservative approach to selection changes and an attempt to balance the load.

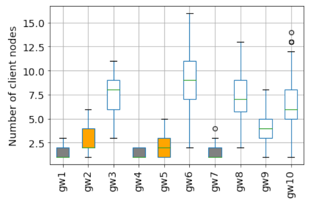
F. Sensitivity analysis

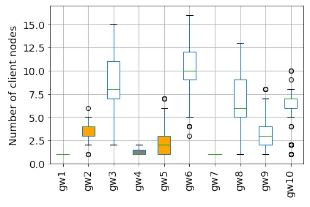
In the following experiments, we study the sensitivity of the GateSelect algorithm to its parameters and the simulated scenario. We use the wireless network configuration with client nodes equally distributed over the 150x150m area at fixed positions to better show the impact of the different parameters. We show results of single experiment runs.

1) Link performance changes: The goal of this experiment is to see if GateSelect can correctly identify the capacity of the gateways and distribute the nodes accordingly. The normal scenario is shown in Figure 11a. On average, 4-5 nodes are assigned per gateway per round.

Next, we assign a fixed 20ms delay to gateways $gw1$, $gw4$, and $gw7$ and a random 5ms to 10ms delay to $gw2$ and $gw5$. As a result, the client nodes tend to avoid selecting those gateways (Figure 11b). If we further increase the latency for $gw1$, $gw4$, and $gw7$ to 40ms, this effect is intensified (Figure 11c). Consequently, the other gateways have to serve more client nodes. However, if we increase the download frequency by a factor of two while keeping the 40ms latency, the distribution becomes again more even, as shown in Figure 11d. As download frequency increases, the performance of the normal gateways degrades as well. As a result, the high-latency gateways become more attractive again and are selected by some nodes.


(a) Normal


(b) 20ms delay added


(c) 40ms delay added

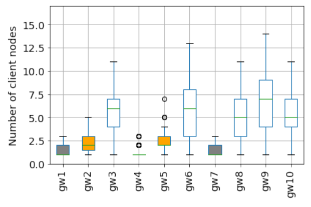

(d) 40ms delay + 2x download

Fig. 11. Wireless - Client node distribution with gateway delay.

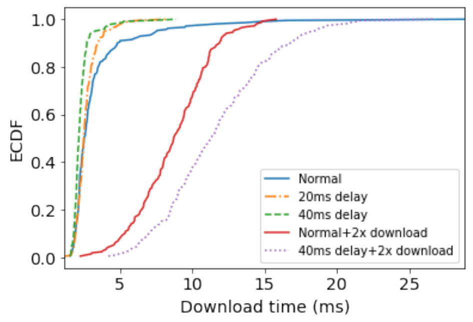
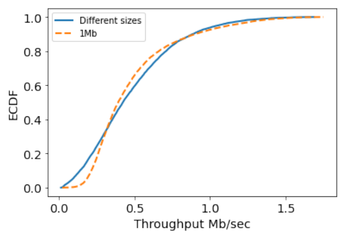
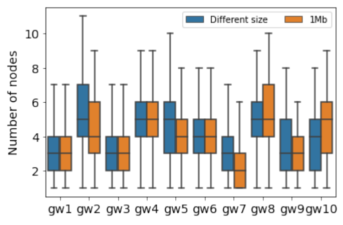


Fig. 12. Download times with different gateway delays

Figure 12 shows the download times with the different link delays. As the link delay increases, the download time increases as well because every client's gateway selection pool is narrowed down to a few good gateway nodes. When increasing the download frequency, the download time increases due to queueing delays and network congestion. We conclude that GateSelect can correctly detect the capacity of the gateways based on the measurements collected over the selection rounds and allocates the nodes evenly to the good gateways.



(a) Throughput



(b) Gateway distribution

Fig. 13. Wireless - Download size sensitivity.

2) Varying download file size: In this experiment, the nodes download files of different random sizes (0.25MB, 0.5MB, 1MB, 3MB, 5MB) instead of the 1MB file used in the other experiments. Figure 13a shows that the achieved throughput per download is slightly lower than with the 1MB file. The gateway distribution (Figure 13b) is nearly identical for both scenarios. We conclude that the gateway selection is not affected by the traffic load.

3) Power of D choices for the selection decision: By default, GateSelect selects $D = 2$ gateways randomly from the selection candidate list, but this number is configurable. We run experiments with $D = [1, 2, 3, 4]$. As shown in Figure 14, when D is lower, the gateways are distributed equally over the nodes. However, the greater D, the higher the probability that many clients will select the same gateway. For $D = 3$, this is the case for $gw3$ and $gw5$, and for $D = 4$, $gw1$, $gw5$, and $gw6$ are in high demand. We conclude that $D = 2$ is a good choice.

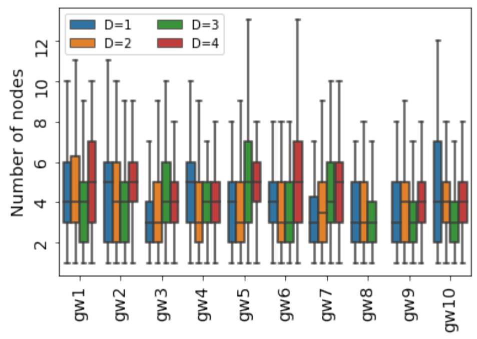


Fig. 14. Wireless - Gateway distribution.

V. Related work

The Internet gateway selection algorithms in the literature focus on performance monitoring, gateway discovery, gateway placement, or routing. In [3]–[5], [17] incorporate collaborative sensing to achieve full gateway monitoring and selects the best one at a time. In [25], the authors present a flexible gateway selection based on more than just network-based criteria (e.g., economic features or user preferences) for gateway selection. However, in the race of choosing the best Internet gateway, often algorithms eliminate the other equally good performing gateways out of the selection. This behavior enforces overall network load imbalance, congestion, and short/long term performance degradation [26]–[29]. Therefore, in a distributed gateway selection algorithm, the individual nodes have to consider the other nodes' selection impact as well. In GateSelect, we focus on the gateway selection at each client node and allow the nodes to choose from the equally good performing gateways, instead of choosing the best gateway node, thus avoiding overloading specific gateways while still receiving a good QoS.

Concerning the fair resource allocation, many algorithms take a load balancing approach that directs client nodes to the least loaded gateway [7]–[9], [14], [15]. The Load Balancing Index metric as defined in [7] uses the total traffic and the capacity of the gateway. In [14], queue length is used to calculate the gateway load. Hybrid approaches [8], [9] calculate a gateway selection score using gateway load, path quality, number of hops, etc. Learning Automata-based Load Balancing (LALB) [15] uses algorithm similar to reinforcement learning to find the perfect combination of performance and load balancing. However, load monitoring algorithms introduces an additional element in the algorithm and are often resource-intensive. Also, there is the risk that all nodes select the same least loaded gateway. GateSelect indirectly estimates each gateway's likelihood to perform well in the future using locally obtained performance measurements and by simple calculations compatible with the limited processing capacity of client nodes as found in IoT networks. We use a randomized selection to distribute the nodes to avoid overloading specific gateways.

Cluster-based resource allocation algorithms [30]–[32] reduces the complexity of the load balancing, resource selection and often used for the resource-constrained network scenario. In [30] proposes the energy-efficient resource allocation considering the cluster node's residual energy. In [31] uses two-step resource allocation where consists of resource block allocation and game based power allocation for the sensor nodes. Besides the clustering complexity, cluster-based algorithms propose a top-down approach where cluster head nodes decide the resource allocation where GateSelect offers an individual, customized resource allocation.

VI. Conclusion

This paper presents GateSelect, a distributed gateway selection algorithm tailored to fulfill each client node's Internet gateway selection using the following techniques: performance-based classification, and random sample selection. We demonstrate GateSelect's adaptability in different network scenario and compared the performance with a greedy, random, and static selection to show the improvement of the Internet gateway performance while maintaining gateway selection balance. GateSelect is simple yet effective, infrastructure- and technology-agnostic, and supports incremental implementation.

In the future, we will integrate the GateSelect algorithm with the lightweight, collaborative monitoring algorithm proposed in [3], [17] to propose the Sense-Share-Select, gateway selection framework and implement the framework in the guifi.net community network testbed.

Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and Next Generation Internet (NGI) Explorers innovation programme under the Grant Agreement No 825183 and by the EMJD-DC program and by the Spanish Government under contract PID2019-106774RB-C21.

References

[1] K. L. Lueth *et al.*, "State of the iot 2018: Number of iot devices now at 7b–market accelerating," *IoT Analytics*, vol. 8, 2018.

[2] E. Dimogerontakis, R. Meseguer, and L. Navarro, "Internet access for all: Assessing a crowdsourced web proxy service in a community network," in *Passive and Active Measurement (PAM)*, 2017, pp. 72–84.

[3] K. Batbayar, R. Meseguer, E. Dimogerontakis, L. Navarro, and R. Sadre, "Collaborative informed gateway selection in large-scale and heterogeneous networks," in *IM 2019*. Washington DC, USA: IEEE, apr 2019.

[4] E. Dimogerontakis, J. Neto, R. Meseguer, L. Navarro, and L. Veiga, "Client-side routing-agnostic gateway selection for heterogeneous wireless mesh networks," in *Integrated Network and Service Management (IM)*, 2017, pp. 377–385.

[5] B. J. Ko, S. Liu, M. Zafer, H. Y. S. Wong, and K.-W. Lee, "Gateway selection in hybrid wireless networks through cooperative probing," in *Integrated Network Management (IM)*. IEEE, 2013, pp. 352–360.

[6] H. Livingstone, H. Nakayama, T. Matsuda, X. Shen, and N. Kato, "Gateway selection in multi-hop wireless networks using route and link optimization," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*. IEEE, 2010, pp. 1–5.

[7] 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, pp. 379–391, 2004.

[8] U. Ashraf, S. Abdellatif, and G. Juanole, "Gateway selection in backbone wireless mesh networks," in *Wireless Communications and Networking Conference (WCNC)*. IEEE, 2009, pp. 1–6.

[9] Y. Yan, L. Ci, Z. Wang, and W. He, "Qos-based gateway selection in manet with internet connectivity," in *2013 15th International Conference on Advanced Communications Technology (ICACT)*. IEEE, 2013, pp. 195–199.

[10] A. Y. Delgado, M. Gadeo-Martos, J. Fernandez-Prieto, and J. Canada-Bago, "A fuzzy balancing load system to improve hybrid ad hoc networks," in *2012 IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2012, pp. 66–69.

[11] H. Xu, L. Ju, and Z. Jia, "Enhance internet access ability for ad hoc network with on-demand gateway broadcast strategy," *International Journal of Wireless Information Networks*, vol. 22, no. 4, pp. 415–427, 2015.

[12] H. Xu, Y. Zhao, L. Zhang, and J. Wang, "A bio-inspired gateway selection scheme for hybrid mobile ad hoc networks," *IEEE Access*, vol. 7, pp. 61 997–62 010, 2019.

[13] Open, free and neutral network internet for everybody. [Online]. Available: http://guifi.net/en

[14] D. Nandiraju, L. Santhanam, N. Nandiraju, and D. P. Agrawal, "Achieving load balancing in wireless mesh networks through multiple gateways," in *2006 IEEE international conference on mobile Ad Hoc and sensor systems*. IEEE, 2006, pp. 807–812.

[15] M. Kashanaki, Z. Beheshti, and M. R. Meybodi, "A distributed learning automata based gateway load balancing algorithm in wireless mesh networks," in *2012 international symposium on instrumentation & measurement, sensor network and automation (IMSNA)*, vol. 1. IEEE, 2012, pp. 90–94.

[16] A. Abujoda, D. Dietrich, P. Papadimitriou, and A. Sathiaseelan, "Software-defined wireless mesh networks for internet access sharing," *Computer Networks*, vol. 93, pp. 359–372, 2015.

[17] K. Batbayar, E. Dimogerontakis, R. Meseguer, L. Navarro, and R. Sadre, "Sense-share: A framework for resilient collaborative service performance monitoring," 2019.

[18] M.Mitzenmacher, "The power of two choices in randomized load balancing," in *Transactions on Parallel and Distributed Systems*. IEEE, 2001, pp. 1094 – 1104.

[19] H. Zhang, J. Zhang, W. Bai, K. Chen, and M. Chowdhury, "Resilient datacenter load balancing in the wild," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 253–266.

[20] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian, "Drill: Micro load balancing for low-latency data center networks," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 225–238.

[21] P. Berenbrink, P. Kling, C. Liaw, and A. Mehrabian, "Tight load balancing via randomized local search," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2017, pp. 192–201.

[22] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: distributed, low latency scheduling," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, 2013, pp. 69–84.

[23] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010, pp. 19:1–19:6. [Online]. Available: http://doi.acm.org/10.1145/1868447.1868466

[24] R. R. Fontes, S. Afzal, S. H. Brito, M. A. Santos, and C. E. Rothenberg, "Mininet-wifi: Emulating software-defined wireless networks," in *2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 384–389.

[25] X. Wang, D. Qu, K. Li, H. Cheng, S. K. Das, M. Huang, R. Wang, and S. Chen, "A flexible and generalized framework for access network selection in heterogeneous wireless networks," *Pervasive and Mobile Computing*, vol. 40, pp. 556–576, 2017.

[26] T. Roughgarden and É. Tardos, "How bad is selfish routing?" *Journal of the ACM (JACM)*, vol. 49, no. 2, pp. 236–259, 2002.

[27] M. Seshadri and R. Katz, "Dynamics of simultaneous overlay network routing, uc berkeley," CSD-03-1291, Tech. Rep., 2003.

[28] D. Acemoglu, R. Johari, and A. Ozdaglar, "Partially optimal routing," *IEEE Journal on selected areas in communications*, vol. 25, no. 6, pp. 1148–1160, 2007.

[29] R. U. Zaman, M. Waseem, A. Farokhi, A. V. Reddy *et al.*, "Traffic priority based gateway selection in integrated internet-manet," in *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*. IEEE, 2016, pp. 18–21.

[30] Sonam Palden Barfunga, P. Rai, and H. K. D. Sarma, "Energy efficient cluster based routing protocol for wireless sensor networks," in *2012 International Conference on Computer and Communication Engineering (ICCCE)*, 2012, pp. 603–607.

[31] L. Liang, W. Wang, Y. Jia, and S. Fu, "A cluster-based energy-efficient resource management scheme for ultra-dense networks," *IEEE Access*, vol. 4, pp. 6823–6832, 2016.

[32] M. Yemini and A. J. Goldsmith, "Virtual cell clustering with optimal resource allocation to maximize cellular system capacity," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–7.