Multi-criteria Optimization of Wireless Connectivity over Sparse Networks

Cristian-Octavian Ojog[a], Radu-Corneliu Marin[a], Radu-Ioan Ciobanu[a], Ciprian Dobre[a,*]

[a]*Faculty of Automatic Control and Computers*
University Politehnica of Bucharest
313 Splaiul Independentei, Bucharest, Romania

Abstract

Opportunistic networking is at the basis of cyber-physical Mobile Networks in Proximity (MNP), through an unique perspective over mobility and the incorporation of socio-inspired networking algorithms. However, results in the field are mostly theoretical, proven to account for stricter hit rate and latency requirements in specific environments. They generally assume that two devices being in proximity automatically see one-another, an assumption which might not stand under real-world conditions (Bluetooth assumes a peering session and close-proximity, WiFi Direct implementations are different between manufacturers, etc.).

Our previous studies in the area show that WiFi is still the most feasible media for opportunistic contacts. WiFi-enabled devices, with out-of-the-box networking capabilities, can connect in an ad-hoc opportunistic network, over wireless routers, and thus support a cyber-physical infrastructure for opportunistically spreading information.

In this article, we propose a machine learning algorithm that aims to increase the number of contacts between mobile nodes by using a smarter WiFi access point selection heuristic. The algorithm is based on properly balancing signal strength, latency, bandwidth, and, most importantly, the number of friends predicted to connect to the respective access point. We show through

*Corresponding author

Email addresses: cristian.ojog@cti.pub.ro (Cristian-Octavian Ojog), radu.marin@cti.pub.ro (Radu-Corneliu Marin), radu.ciobanu@cti.pub.ro (Radu-Ioan Ciobanu), ciprian.dobre@cs.pub.ro (Ciprian Dobre)

simulations based on real-life tracing data-sets that our proposed solution not only increases the likelihood of opportunistic contacts, but it also evenly distributes social subgraphs of users over wireless networks while improving the overall hit rate.

Keywords:
machine, learning, opportunistic, social, networking, WiFi

---

1. Introduction

Opportunistic networks (ON) are a type of infrastructure-less mobile network, in which nodes are mobile devices carried around by humans and communicate with each other when in proximity via the *store-carry-and-forward* paradigm. When not in direct proximity, the source node opportunistically forwards the message through the nodes in its wireless range. These nodes carry the message either until the destination is met or the message expires.

Recently, there has been an increase in the interest towards ONs, as they show great potential towards the realisation of Mobile Networking in Proximity (MNP) and cyber-physical systems alike[1]. This community-driven type of network offers a good platform for distributed context sharing. Given the always on-the-move nature of people, this type of network often faces many disconnections and variable delays, and can go from sparse, in secluded locations, to crowded, in public places, or in scenarios where an Internet connection might not even be available. This makes it difficult to properly route messages between the mobile nodes. In essence, MNPs represent spontaneous networks composed of mobile nodes that opportunstically connect in a peer-to-peer (P2P) manner when in proximity to one another[2]; as such, opportunistic networking clearly encompasses all concepts from MNP, as it combines the social nature of mobile nodes with techniques from mobile P2P[**?**].

Our previous studies have shown that opportunistic networks combined with social information, are more than suitable for generating and sharing large amounts of data [3, 4], and even for distributed computing [5, 6], while maintaining acceptable hit rates and latencies [7, 8]. In [9], we showed that WiFi is the most feasible media for opportunistic networking; due to its higher wireless range and broader bandwidth, it is able to support more opportunistic contacts than Bluetooth, which, on the other hand, tends to isolate users into micro-communities. Furthermore, WiFi is fully-standardized

and supported over all major mobile platforms, as opposed to near field communication (NFC) or WiFi Direct, which are not uniformly implemented by all modern mobile operating systems (OSs).

However, using WiFi as the media for opportunistic contacts, implies that the performance of ONs is dependent on the local infrastructure; nodes are discovered through multicast in the local wireless area network. Therefore, opportunistic contacts are established between nodes that are connected to the same wireless access point (AP). Currently, most mobile OSs only focus on connecting to APs that provide a more stable Internet connection and, for this reason, they only take simple metrics into account, such as choosing the best signal strength. However, such functionality is not relevant for ONs, which do not rely on the Internet for routing or forwarding messages, rather they make use of any available wireless connection between two mobile nodes in order to direct a message towards its destination. As such, we propose to design a WiFi AP selection algorithm that attempts to increase the number of opportunistic contacts between nodes that share social bonds; in our work, a social bond is considered to be a link in the social graph, namely a relationship between two nodes in an online social network [10]. By doing so, we believe that we are able to significantly increase the chances of messages reaching their destination, thus maximizing the hit rate and minimizing the latency of ONs. Moreover, we attempt to alleviate congestion, by distributing nodes grouped by social links towards the same wireless APs. The novelty of the solution stems from the decoupled nature of the algorithm: mobile nodes act independently of one another and, through exchanging messages during opportunistic contacts, they learn the behaviour of other nodes socially connected to them so that they connect to the same APs. This is achieved without any broadband Internet connections, thus being able to meet the low-latency requirements from MEC.

The remainder of this article is organized as follows. Section 2 presents a comparison of existing algorithms. In Section 3 we propose a new algorithm for AP selection, with an extended focus on improving ONs. Section 4 details the implementation of the simulation platform; it also provides an in-depth performance analysis of our proposed solution in comparison with existing algorithms. Finally, Section 5 gives a breakdown of the article and of the obtained results.

2. Related Work

As WiFi has become nearly ubiquitous, AP selection algorithms have been the main focus point of both the mobile computing research community, as well as of the mobile industry. Most modern mobile OSs use signal strength as the main criteria for choosing the most appropriate AP. According to the official support guide [11], iOS 8 selects target Basic Service Set Identifiers (BSSIDs) based on the difference in signal strength against the current BSSID's Received Signal Strength Indicator (RSSI). Furthermore, it selects target BSSIDs whose reported RSSI is $8dB$ or greater than the current BSSID's RSSI if the client is transmitting or receiving data, and $12dB$ for idle clients. Android is also focused on choosing the AP with best signal strength, but it is more customizable [12] by using a *wpa_supplicant* configuration file which allows choosing other network criteria, such as security policy. However, this behaviour is used by both OSs only when interacting with unknown APs. Whenever the mobile node is near a series of APs in which at least one of them is known, it uses its caching capabilities to connect to the last connected AP from that list.

Most existing algorithms for selecting APs aim to improve bandwidth allocation or maximize throughput for broadband connections. While these criteria should still be met by our solution, they are secondary objectives in ONs. This is due to the fact that the main purpose in such networks is to accurately route messages between devices without the necessity of an Internet connection. Therefore, the main criteria to improve is the number of meaningful connections between similar users. There are several attempts at improving bandwidth fairness, in which all nodes connecting to the same AP share the provided network resources in an equitable manner, either by probing [13], by introducing notions from game theory [14, 15], or even by creating a system in which APs communicate between themselves in order to gain better understanding of the overall state of the system in order to apply a modified version of the max-flow algorithm [16]. As for improving network throughput, authors in [14] introduce an algorithm in which throughput is maximized by calculating regret for a selected connection as the difference in payoff between possible and actual AP choices, and improving the selection process based on learning historical values.

Tabrizi *et al.* [17] focus on maximizing Quality of Service (QoS) through a reinforcement learning algorithm which calculates a reward for each encountered AP based on probed throughput. Our solution improves on this

approach, as we estimate the selection criteria by keeping detailed history and using the predictability of the average human schedule to our advantage in offering more accurate data, as opposed to probing for momentary data. Also based on the predictability of human mobility, Pang *et al.* [18] propose building a generalized system based on user-submitted reports which are used to predict future events.

Another interesting approach at solving the AP Selection problem is Virgil [19], a selection algorithm which aims to satisfy varied QoS constraints. On each discovered AP, Virgil quickly connects the mobile device to it, and runs a battery of simple tests, designed to probe the AP's availability for use. It uses reference servers and estimates bandwidth and round trip time (RTT) while also testing its open ports in order to calculate a score for each AP. The results are stored in a local database for future estimates. Although an interesting solution, Virgil represents a complex probing mechanism, which only allows logging for future predictions which can lead to improving the accuracy of the results. The main criteria it aims to improve is the average percent of actually finding a connectible AP. It manages to achieve results of 20 to 100% better than those obtained by the regular maximum signal strength approach. However, the authors highlight that the number of connectible APs found with a scan is low. Over 60% of all scans find less than 3 APs that are connectible. This strengthens the motivation for our solution which attempts to find the AP which maximizes our potential for social interactions in ONs.

Fukuda *et al.* [20] propose a decentralized solution for AP selection which attempts to solve the problem of imbalanced traffic load on APs. It consists of probing the available APs from a scan and using one of two selection algorithms: maximizing local throughput (MLT) or avoiding APs with larger packet error rates (AALP). However, the waiting time required for probing each AP, especially in the case of timeouts, is not viable when the user wants to connect to other nodes as quickly as possible. Our proposed solution uses local historical values to predict viable candidates, thus removing the need for probing and making a seamless transition between the current AP and the better choice.

As an overview, our solution provides a smart decision of choosing APs in order to increase the number of opportunistic contacts between peers that are socially connected. This is achieved through a machine learning algorithm that is able to record the behaviour of other nodes, thus enabling a decentralized AP selection decision. Moreover, the algorithm seamlessly complements

existing algorithms implemented in mobile OSs by also balancing the signal strength, the available bandwidth and the connection latency. Our solution takes energy efficiency into account through removing the need for probing, and evenly distributing social subgraphs of users towards the same APs in order to reduce network congestion.

3. Algorithms and Heuristics for Access Point Selection

In this Section we offer an in-depth analysis of our proposed algorithm for AP selection. First, we briefly present our motivation by highlighting the main pitfalls in existing implementations on mobile devices. Further on, we introduce a machine learning approach based on a Q-learning algorithm; we present a framework for the Markov decision process, we detail the equations for the Q-learning algorithm, we provide our approach for the cost function, and we provide the methods of normalizing the scanned parameters received from the mobile device in order to correctly calculate the aforementioned cost function.

3.1. Discussion

After having compared existing approaches to AP selection in Section 2, we consider that the built-in signal strength maximizing algorithm is insufficient for ONs, as it only attempts to guarantee a stable Internet connection, while ignoring the untapped potential of opportunistic contacts in the local area network. In this sense we rule in favour of an approach that would enable users to opportunistically exchanges messages with a higher probability of reaching their destination with lower latency in the case of poor broadband connections (a common case in edge networks), but not limited to this scenario, while properly handling mobility and context changes.

By analysing real-life traces gathered using the HYCCUPS Tracer, we have proved that we are able to predict meaningful connections between users by using their online social friendship status (i.e. Facebook) [9]. With this in mind, we propose an algorithm that learns the number of friends connected to any AP through opportunistic interactions and uses it as a primary metric in the AP selection process. This metric is the key factor in the reward function of our reinforcement learning algorithm. But first, we explain the reasoning behind choosing a reinforcement learning.

In the mobile OS built-in approach, it is fairly simple to retrieve the required signal strength metric, just by executing a WiFi scan. The scan

returns a list of all APs visible by the mobile device, along with an identifier (BSSID) and the corresponding signal strength, making it easy to directly choose based on the gathered information. In the case of more advanced criteria, such as number of friends or even latency, acquisition is conditioned by establishing an active connection with each AP, which leaves two possibilities: either probing each available AP for that information, or attempting to predict those metrics based on historical values.

The first approach has several weak points. First of all, the time required for a probe is unpredictable. Although theoretically the duration of a probe should be less than a second[21] based on the 802.11 protocol specification, we have empirically observed such probes taking up to 3-5 seconds per AP in the case in which the distance between the AP and the mobile device is quite large and when package collisions are the norm. Such high intervals imply noticeable overhead and significant down-time periods, i.e. when the mobile device disconnects in order to find a better AP. Such interruptions would be extremely inconvenient or even unacceptable for users. A possible solution would be to avoid probing when the device is actively using the WiFi connection, however there is little to no support for determining a user's direct intent to use Internet services. Secondly, constantly probing every visible AP would take a heavy toll on battery life; we consider that a feasible solution should take energy-efficiency into consideration.

A better alternative is to predict the needed metrics, thus removing the need for probing. In this case, the problem is reduced to determining a common AP for two distinct nodes, who are connected in an either online fashion (via pre-fetched list of friends for each node gathered from social networks) or offline (having met before by connecting to the same AP and sharing messages), without knowing each other's decision. Furthermore, our approach is focused on facilitating connections between all nodes in the same subgraph of friends, and not between two given nodes. We rule in favor of a reinforcement learning approach that is able to learn the patterns and habbits of user interactions in order to more accurately predict the desired metrics, towards increasing the number of connections between socially-connected peerds.

3.2. A Reinforcement Learning Approach

The first step needed towards a solution is defining the main aspects for the Markov decision process illustrated in Table 1. Given that the States, Model, Actions and Reward are straightforward, we need to define a policy

States:	Define the state in which the agent is in a particular moment in time. In our case, the agent is the mobile device, and the states can be Disconnected, Connected to AP_i (where $AP_1 ... AP_n$ are the connectible APs returned by a mobile WiFi scan)
Model:	Also known as Transition model, represents a function with three parameters: State, Action and State after executing that action. This function produces the probability of ending in the final state, given the initial state and chosen action. $T(s, a, s') \sim Pr(s'|s, a)$
Actions:	Define the possible actions to execute and depend on the current state. In our case, the actions are either disconnect from current AP or connect to AP_i.
Reward:	Consists of the feedback given by the environment. It is a function of a heuristic nature that is the crux of the RL algorithm. Given that fact, we have made it easy to tweak in order to get the best results and will be covered in more detail below.

Table 1: Markov Decision Process

function $\pi(s)$, which determines the action that will maximize the reward of taking an action between two in our model. To do this, we need to calculate the long term reward, U, for each possible state. Therefore, we obtain the following two equations:

$$U(s) = R(s) + \gamma \cdot \max_a \cdot \sum_{s'} T(s, a, s')U(s') \tag{1}$$

$$\pi(s) = \underset{a}{argmax} \cdot \sum_{s'} T(s, a, s')U(s') \tag{2}$$

Unfortunately, due to the $max(a)$ function, U is non-linear and thus difficult to calculate. In order to simplify, Q-learning introduces the function appropriately named $Q(s, a)$, which improves upon the long term reward equation by making it linear:

$$Q(s, a) = R(s) + \gamma \cdot \sum_{s'} T(s, a, s')max_{a'}Q(s', a') \tag{3}$$

Therefore the value for arriving in state s, leaving via a, proceeding optimally thereafter becomes:

$$U(s) = \max_a Q(s, a) \tag{4}$$

$$\pi(s) = \underset{a}{argmax} Q(s, a) \tag{5}$$

Our proposed reward function must be able to satisfy the following criteria:

- Promote APs for which the number of connected friend nodes is maximum;
- Encourage undiscovered nodes in order to make better decisions in the future;
- Maximize the potential of predicting the right criteria by making use of the predictable nature of our daily routine;
- Use multiple criteria to easily solve tie-breakers.

Let A be a vector of all the criteria used to determine the reward for a particular state,

$$A = \begin{bmatrix} latency \\ linkSpeed \\ signalStrength \\ numFriends \end{bmatrix} \tag{6}$$

and F, the vector of factors which normalize each criteria and gives them an equal importance in the overall evaluation of the reward,

$$F = \begin{bmatrix} f_{latency} \\ f_{linkSpeed} \\ f_{signalStrength} \\ f_{numFriends} \end{bmatrix} \tag{7}$$

However, in this current form, it is unclear how the algorithm reacts in an unknown world, where no APs have yet to be discovered. Given that the criteria vector A must be learned to be used in predicting future values, it cannot be calculated beforehand. Therefore, whichever AP is chosen first in a certain location, will most likely be chosen every time. In order to solve this issue, we have added a bias to the reward function. This bias should provide a better reward for new APs and it will be reduced over time.

Let $Bias \epsilon [0, 1]$ that determines whether the reward should lean more towards discovering a new AP or basing the decision on the known A vector.

By doing so, we encourage the mobile device to want to know the APs before making a more educated decision about which one to connect to next. Since the number of connectible APs is reduced (1-3 on average) by an average WiFi scan, this method converges quickly, as will be seen in Section 4. Thus our equation for $R(s)$ becomes:

$$R(s) = A^T(s) \cdot F \cdot (1 - Bias) \tag{8}$$

The metrics measured in the criteria vector A are not constant throughout the day for any given AP; we cannot assume that an AP always has poor bandwidth, only based on the fact that it has reported such values over a crowded weekend. However, we consider it safe to assume that people usually have a predictable routine during the weekdays, and some even in the weekend. With this in mind, we decided to split the vector A into different versions of what will be learned every day. Not only that, but we have split the day into three equal periods, in order to have a more accurate representation of the state of each known AP. Therefore $Bias$ becomes $Bias(t)$, and t becomes $t(day\ of\ week, hour\ of\ day)$. Furthermore, the value of $Bias$ increases linearly over time, incentivising the discovery of new APs. In our experiments, $Bias$ reaches its maximum value after a week.

This leads to a slow convergence which can be overcome by sharing and aggregating the criteria vector A as context data during an opportunistic contact, similar to our work in [8]. We have found that this solution drastically improves the learning rate of the algorithm.

3.3. Normalizing mobile network parameters

We consider that the metrics measured in the criteria vector A should have equal weights in the output of the reward function. However, normalizing said criteria is rather cumbersome, as some criteria are not linear functions, e.g. latency. Furthermore, the number of friends also provides some issues since there is no maximum number of friends to compare to, or, if we were to set a maximum number, the fact that there's generally a low number of friends per AP at a moment in time makes it such that the influence of that criteria in the reward function would be swept aside by all the others. Therefore, A is normalized as follows:

- Signal strength is technically bound between $-100dbm$ and $0dbm$. Even so, the values encountered in real life never reach these limits. On average, signal strength below $-80dbm$ renders an unusable connection,

and, on Apple devices, even values below $-60dbm$ are generally not recommended; a good quality signal strength is considered to be any value that's more than or equal to $-50dbm$. Therefore, for better results we have to first bind the value between $-80dbm$ and $-50dbm$ and then convert it to percentage:

$$signal\ strength\% = \frac{min(-50,\ max(-80,\ signal\ strength))}{100} + 1 \quad (9)$$

- When it comes to latency, there is no maximum bound, and setting one does not help due to its non-linear behaviour. Therefore, to convert its output towards $[0, 1]$ more accurately, we use a non-linear conversion function:

$$latency\% = \frac{1}{scanned\ latency^2} \quad (10)$$

- Link speed (or bandwidth) is the most straight-forward, as the value in Mbps can easily be converted:

$$link\ speed\% = \frac{max(0,\ min(100,\ scanned\ link\ speed))}{100} \quad (11)$$

- The number of friends provides no maximum bound, and is, in general, a small number, typically below 10. Therefore we need to compare the scanned number of friends to the total number of friends that we can connect to, based on the AP selection:

$$num\ friends\% = \frac{scanned\ num\ friends}{total\ num\ friends} \quad (12)$$

with the note that the total number of friends can be 0, case in which the whole criteria becomes irrelevant.

4. Experimental Results

4.1. Algorithm and Simulator Implementation

In order to test the validity of the proposed algorithm, we have used real-life models based on traces collected using the HYCCUPS Tracer [9];

the tracer is an Android application designed to run in the background and collect contextual information, such as: CPU usage, memory usage, WiFi usage, etc. It also implements an opportunistic networking engine which allows tracing opportunistic interactions between peers over WiFi networks. In our analysis, we have used a dataset collected using the HYCCUPS Tracer in an academic environment at the University Politehnica of Bucharest; the tracing experiment lasted for 64 days, in 2012, having 66 participants.

The collected traces were analyzed using MobEmu [3], an opportunistic network emulator that is able to replay a trace and apply any routing / forwarding algorithms when two nodes meet. In MobEmu, each node corresponds to a participant to one of the tracing experiments, and each node has an unique ID, along with other statistics and tracing information from the mobile device itself. Another important feature is finding how many users are in contact, which in real life would translate to how many other known nodes are connected to the same AP.

Our proposed algorithm was implemented in a simulator based on MobEmu, which can also manage virtual wireless APs based on traces obtains with the HYCCUPS Tracer. Figure 1, illustrates the hierarchy of data structures that hold relevant information for the algorithm implementation. At the base we have the *Environment*, which holds all the metadata needed about every AP a mobile node has encountered. The *Environment* creates an *APInfo* entry for each simulated AP which holds all the methods for fetching and updating data and acts like a wrapper for each individual time fragment, in an *APInfoFrag*. Since the AP metrics change on an hourly, or even weekly basis, we retain all metadata for each timeslot in an *APInfoFrag*. This container also implements the cost calculation method, the cornerstone of the entire algorithm.

The *Environment* is also the basis of the simulation model, as it contains all of the simulated *Rooms*, APs and mobile nodes. Each *Room*, or *Place*, contains several connectible APs, each with a different configuration and different capabilities. The simulation assigns a position for every mobile node for every hour in a day. The actual assignment is problematic, as the simulator must replicate human behaviour as closely as possible. Initially, we used a random position model, but we discovered that this approach did not give accurate simulation results, because people tend to have more repetitive schedules, especially during a weekday [9]. For this reason, when a node is created, it has a specific set of 2 to 4 places where it is expected to be found during a day, with a slight chance (10 - 15 %) of choosing a different *Place*.

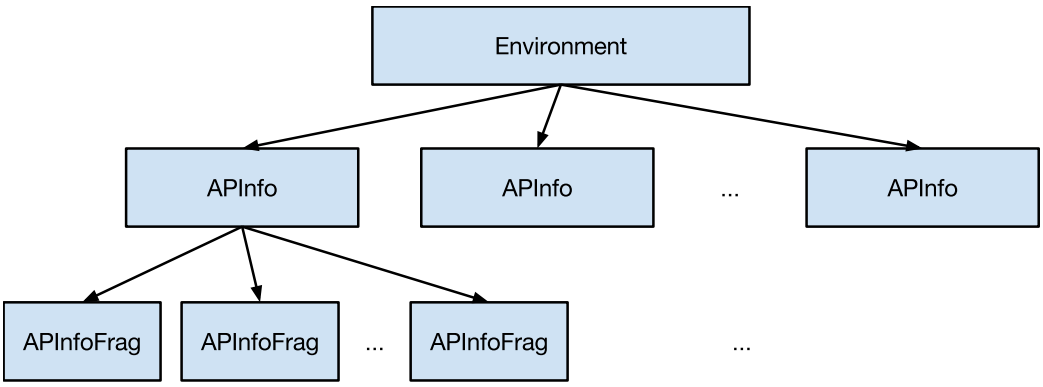

Figure 1: Algorithm implementation skeleton.

When a mobile node connects to an AP, it starts gathering information about it, such as latency, bandwidth and number of friends. It also starts sending out messages to other mobile devices in order to gather more information about APs it can possibly encounter in the future, or refresh information about the already known APs. In the latter case, it compares the timestamps from the latest scans on both devices, for a particular AP, and keeps the most recent one.

Out of the four measured criteria, computing the number of friends is, by far, the most challenging as we needed to find a different type of metric calculation so that the normalized value still carries some importance in the overall rating / reward of the AP. Therefore, the number of friends is computed as the number of friends connected to the particular AP divided by the total number of friends available in all the APs found by a WiFi scan. Even though this value cannot be discovered without connecting to the AP beforehand, we can predict it over time either by previous encounters or by encounters made and shared by other mobile devices.

The base component of the simulator's *Environment* is the *Simulation World*, as illustrated in Figure 2. It holds all the information about nodes, APs, as well as about all of the simulated locations. The *Simulation World* also dictates the behaviour of all nodes; each node has a predictable schedule, implying that, in a simulated day, it will change its location to predetermined locations which are assigned randomly at simulation start. To better simulate human behaviour, nodes use a small chance of changing their schedule to a new location, in order to cover unexpected events, and to encourage the nodes to meet new peers.

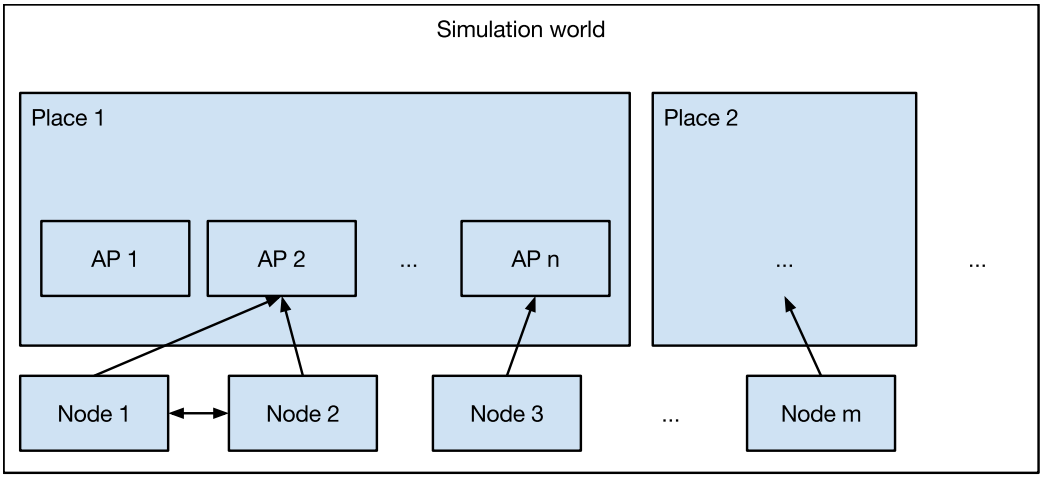

Figure 2: Simulator structure.

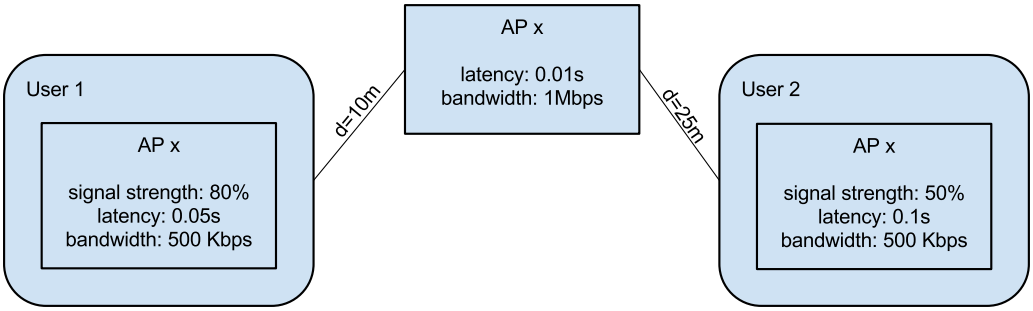

Figure 3: The same access point parameters seen by different mobile nodes.

Every AP simulates the following parameters: signal strength, bandwidth, and latency. Furthermore, each node in a place sees each AP differently, based on its distance from the respective AP, and other miscellaneous factors. Therefore, each node must have its own version of the environment. For instance, in Figure 3, $User1$ and $User2$ record different parameters for APx. $User2$ has lower latency than $User1$, because the distance between them and the AP is different, and with a worse signal strength, more packages are potentially lost. Furthermore, the bandwidth has to be shared between them.

The simulation advances in a granular fashion, splitting a day into multiple shards. Typically, simulations advance hour by hour, in order to get the

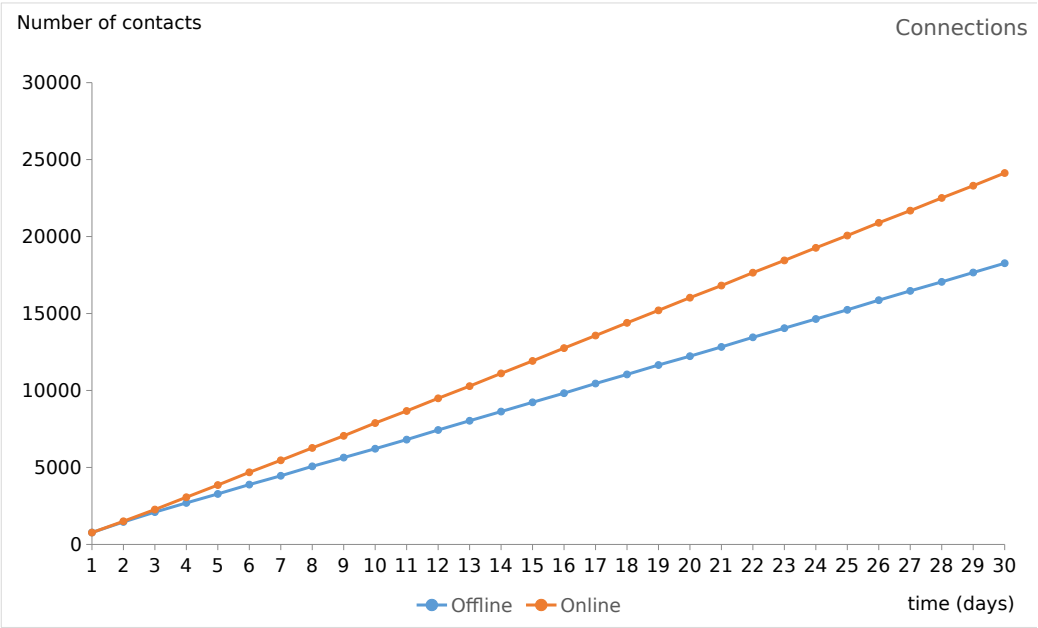

Figure 4: Online vs offline friend learning.

most accurate results. Every hour, a node changes position inside a *Place*, thus changing the signal strength, or traveles to another *Place*. Afterwards, the node chooses an AP based on the current information.

When a node chooses a new AP and connects to it, it exchanges all the information stored in its *Environment* with its peers. Peers involved in such an exchanged merge the metrics for latency, bandwidth and number of friends connected to the AP. Signal strength is not necessary, since it can be learned during the WiFi scan, and depends on the specific device's position.

As mentioned earlier, the method of computing the number of friends has provided more difficulties than the others. For this reason, we have chosen two approaches for creating a social network of mobile devices: the online network consisting of fetched social network information directly from the user's social account (i.e. Facebook), and the offline network which assumes nothing about the initial relationship between nodes and builds its social network based on the regular meetings between nodes. Using the latter approach, if two nodes meet for more than a minimum period of time, they become connected and exchange information the next time they come into contact.

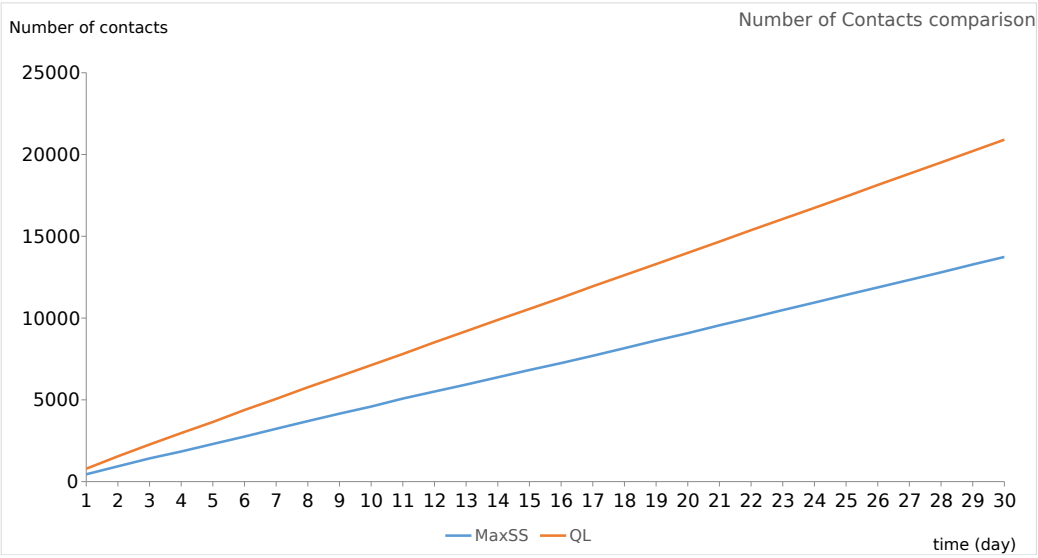

Figure 5: Comparison between *QL* and *MaxSS*, regarding number of contacts.

Figure 4 shows a direct comparison between the two approaches, when calculating the number of connections made between the mobile nodes during a 30 day simulation, with the legacy algorithm. By connection we understand a contact between nodes who are connected to the same AP, that lasts more than 20 minutes. Further on, we will only be focusing on results for the online network, as it has been proven experimentally that knowing the social network of the nodes beforehand significantly increases the number of connections [3].

4.2. Simulation results

In assessing the performance of our proposed solution, namely *QL*, we compare against the current algorithm for choosing APs currently implemented in most modern mobile OSs, respectively maximizing signal strength, or *MaxSS*.

Figure 5 shows that *QL* has a significat increase in the number of connections for the same batch of generated users and *Environment*. The simulation was executed on the same initial social graph, and the users take the same choices in moving in-between *Places* during a day. *QL* achieves an improvement of 55.9% on average, for a simulation batch of 30 days. Furthermore, *QL* improves the distribution of users per AP, as illustrated in Figure 6; although

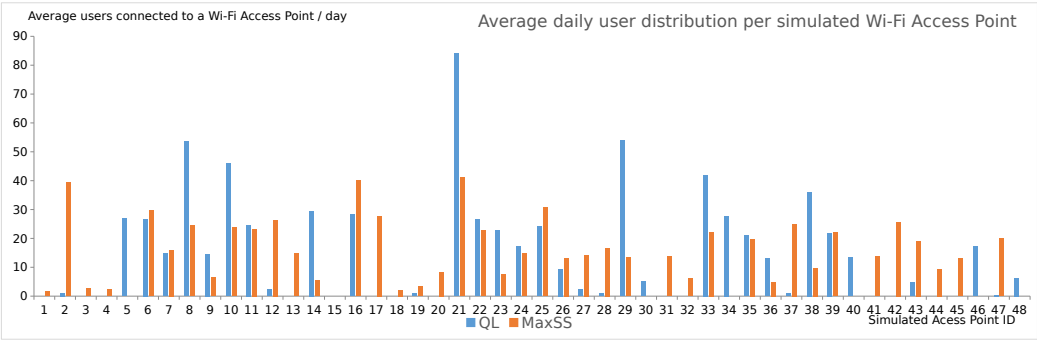

Figure 6: Comparison between *QL* and *MaxSS*, regarding the distribution of users per simulated AP.

link speed units (%)	latency (%)	signal strength (%)	number of friends (%)	contacts
0	25	25	50	20228
0	25	0	75	18931
0	50	0	50	21082
25	25	25	25	19504

Table 2: Comparison between the different configuration options for the F vector

users choose APs without being influenced by their peers, the distribution remains relatively uniform. The figure shows that users are connecting to APs in a more even distributiong thus leading to less congestioned wireless networks, which, in turn, render better QoS to all connected peers. Furthermore, the Figure also shows that the proposed AP selection heurestic has also increased the number of connected peers, thus facilitating a higher coverage than the *MaxSS* heurestic. Moreover, users belonging to one social subgraph will naturally prefer the less crowded APs, while each node connects to the same AP as its user's friends.

The simulator was designed to be configurable in order to account for the ever-changing nature of the requirements for WiFi access for different users. Table 2 lists the configurations that have been tested and also highlights the influence each metric has on the overall results – the number of friends is the most influential criteria in the cost function (50%), while latency and signal strength account for the rest (25% each). Out of the four criteria, latency is responsible for the uniform distribution of users per AP; if latency

is high, then a user who doesn't know anyone who is connected to the APs in range will choose the one AP that's less crowded (which might offer better latency), thus uniforming the distribution of users. Signal strength is the least influential factor and is only used to filter out poor networks.

Last, but not least, Figure 7 shows that *QL* renders a hit rate curve which converges faster than *MaxSS*. Moreover, the faster convergence is even more pronounced for larger tracing data-sets which further proves that our algorithm is a good candidate for improving ONs to the point that delays can be met in MEC. Based on our previous experience on conducting tracing experiments[22], we feel obliged to describe the timeline of such endeavours. A tracing experiment can be split up into three phases: adoption, regular usage, and discontinuance. Both the adoption and discontinuance represent the extremities of a tracing data-set and do not reflect the usage patterns of day-to-day mobile users; however, the central phase encompasses the interval in which most (if not all) users are continuously interacting in a real-life scenario. This pattern can also be observed in Figure 7, which further proves that our proposed selection heuristic is able to steadily increase the likelihood of messages reaching their rightful destinations as more and more users are adhering to the opportunistic network.

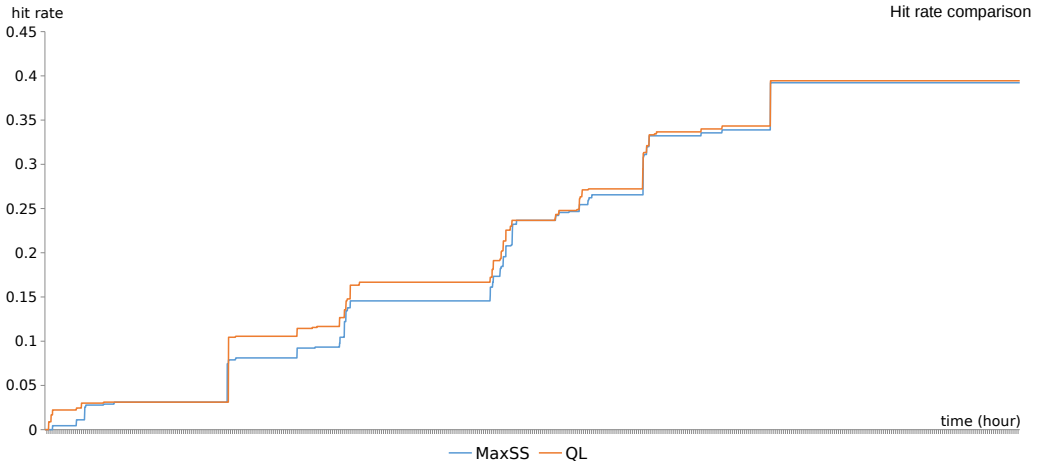

Figure 7: Comparison between *QL* and *MaxSS* regarding the message hit rate.

4.3. Performance Review

As highlighted by Khan et al.[23], learning user patterns in mobile networking is a costly endeavor. However, as we previously mentioned in Section 3, our approach is much more battery-efficient than the existing probing mechanisms; our solution is a reactive algorithm that adapts to the mobile OS's intrinsic WiFi scanning heuristic, and does not trigger any additional scans similar to the probing approaches. The spatial and temporal dimensions of the algorithm are minimalistic, as the heuristic is based on simple mathematical computations that evaluate the feasibility of connecting to an AP and stores such values locally; given that the user will only connect to a fairly limited set of WiFi APs, the spatial dimensions is quite scarce ($\approx$10K of storage) and can easily be satisfied by modern mobile devices.

The only remaining encumberance of the Q-learning approach is its ability to reach an optimal solution. As mentioned in Section 3, the AP selection heuristic can have a slow convergence due to the highly dynamic environment that it must adapt to. However, based on our previous work that proved that mobile users tend to interact amongst themselves with a high degree of predictability[22, 4], coupled with disseminating the learned state to peers through context-dissemination[7], we have managed to drastically speed up the convergence. As future work, we will attempt to treat the worst-case scenarios in which the proposed heuristic is unable to converge to an optimal value and instead leads towards a state with low QoS; in such cases, we rule in favor of balancing between the *QL* heuristic and the default AP selection heuristic, namely *MaxSS*.

5. Conclusions

In this article, we have presented the process and motivation for building a new AP selection algorithm for mobile devices aimed at improving node connectivity in opportunistic networks. Our previous studies [3, 7, 8, 9] show that ONs are more than feasible for solving most challenges due to the methods through which context changes and mobility are treated and viewed as opportunities, instead of faults, and they make for an interesting infrastructure for socio-aware MNP.

We have provided an in-depth survey about current approaches, not only in the modern mobile OSs, but from the research community as well, and we have highlighted both their advantages and pitfalls.

We have proposed and have provided step-by-step explanations about the logic behind a machine learning approach to solving the aforementioned problem. Furthermore, we have presented the simulation platform and we have proven that the algorithm is feasible even on large sets of nodes and for longer periods of time. Also, the algorithm successfully maximizes any criteria required by the user, thus further proving its flexibility in complex and ever-changing situations.

Our results show a significant improvement over existing algorithms, proving that a machine learning approach not only increases the number of connections between nodes, but also improves the convergence of the network's hit rate. Moreover, our algorithm avoids overloading WiFi routers, by automatically distributing social subgraphs of nodes to similar APs, which further proves its feasibility for mobile edge networks.

Acknowledgement

The research is supported by project MobiWay, "Mobility beyond Individualism" (PN-II-PT-PCCA-2013-4-0321). The work has been co-funded by the project DataWay, "Real-time Data Processing Platform for Smart Cities: Making sense of Big Data" (PN-II-RU-TE-2014-4-2731).

References

[1] F. Hu, Y. Lu, A. V. Vasilakos, Q. Hao, R. Ma, Y. Patil, T. Zhang, J. Lu, X. Li, N. N. Xiong, Robust cyberphysical systems: Concept, models, and implementation, Future Generation Computer Systems 56 (2016) 449 – 475. doi:http://dx.doi.org/10.1016/j.future.2015.06.006. URL http://www.sciencedirect.com/science/article/pii/S0167739X15002071

[2] Y. Wang, A. V. Vasilakos, Q. Jin, J. Ma, Survey on mobile social networking in proximity (msnp): Approaches, challenges and architecture, Wirel. Netw. 20 (6) (2014) 1295–1311. doi:10.1007/s11276-013-0677-7. URL http://dx.doi.org/10.1007/s11276-013-0677-7

[3] R. I. Ciobanu, C. Dobre, V. Cristea, Social aspects to support opportunistic networks in an academic environment, in: Ad-hoc, Mobile, and Wireless Networks, Springer, 2012, pp. 69–82.

[4] R.-I. Ciobanu, R.-C. Marin, C. Dobre, Interaction predictability of opportunistic networks in academic environments, Transactions on Emerging Telecommunications Technologies 25 (8) (2014) 852–864.

[5] R.-C. Marin, Hybrid contextual cloud in ubiquitous platforms comprising of smartphones, International Journal of Intelligent Systems Technologies and Applications 12 (1) (2013) 4–17.

[6] R.-C. Marin, C. Dobre, Reaching for the clouds: contextually enhancing smartphones for energy efficiency, in: Proceedings of the 2nd ACM workshop on High performance mobile opportunistic systems, ACM, 2013, pp. 31–38.

[7] R.-I. Ciobanu, R.-C. Marin, C. Dobre, V. Cristea, C. X. Mavromoustakis, Onside: Socially-aware and interest-based dissemination in opportunistic networks, in: Network Operations and Management Symposium (NOMS), 2014 IEEE, IEEE, 2014, pp. 1–6.

[8] R.-I. Ciobanu, R.-C. Marin, C. Dobre, V. Cristea, C. X. Mavromoustakis, G. Mastorakis, Opportunistic dissemination using context-based data aggregation over interest spaces, in: Communications (ICC), 2015 IEEE International Conference on, IEEE, 2015, pp. 1219–1225.

[9] R.-C. Marin, C. Dobre, F. Xhafa, Exploring predictability in mobile interaction, in: Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on, IEEE, 2012, pp. 133–139.

[10] F. Xia, L. Liu, J. Li, J. Ma, A. V. Vasilakos, Socially aware networking: A survey, IEEE Systems Journal 9 (3) (2015) 904–921. doi:10.1109/JSYST.2013.2281262.

[11] Apple, ap selection in apple ios. from https://support.apple.com/en-lb/ht203068, accessed on may 2, 2015.

[12] J. Niu, Y. Gao, S. Guo, C. Tong, G. Du, An Adaptive WiFi Rate Selection Algorithm for Moving Vehicles with Motion Prediction, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 85–92.

[13] H. Gong, J. Kim, Dynamic load balancing through association control of mobile users in wifi networks, Consumer Electronics, IEEE Transactions on 54 (2) (2008) 342–348.

[14] L.-H. Yen, J.-J. Li, C.-M. Lin, Stability and fairness of ap selection games in ieee 802.11 access networks, Vehicular Technology, IEEE Transactions on 60 (3) (2011) 1150–1160.

[15] C. Tekin, M. Liu, R. Southwell, J. Huang, S. H. A. Ahmad, Atomic congestion games on graphs and their applications in networking, Networking, IEEE/ACM Transactions on 20 (5) (2012) 1541–1552.

[16] S. K. Dandapat, B. Mitra, R. R. Choudhury, N. Ganguly, Smart association control in wireless mobile environment using max-flow, Network and Service Management, IEEE Transactions on 9 (1) (2012) 73–86.

[17] H. Tabrizi, G. Farhadi, J. Cioffi, A learning-based network selection method in heterogeneous wireless systems, in: Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, IEEE, 2011, pp. 1–5.

[18] J. Pang, B. Greenstein, M. Kaminsky, D. McCoy, S. Seshan, Wifi-reports: Improving wireless network selection with collaboration, Mobile Computing, IEEE Transactions on 9 (12) (2010) 1713–1731.

[19] A. J. Nicholson, Y. Chawathe, M. Y. Chen, B. D. Noble, D. Wetherall, Improved access point selection, in: Proceedings of the 4th international conference on Mobile systems, applications and services, ACM, 2006, pp. 233–245.

[20] Y. Fukuda, T. Abe, Y. Oie, Decentralized access point selection architecture for wireless lans, in: Wireless Telecommunications Symposium, 2004, IEEE, 2004, pp. 137–145.

[21] M. S. Gast, 802.11 Wireless Networks: The Definitive Guide, Second Edition, O'Reilly Media, Inc., 2005.

[22] R.-C. Marin, C. Dobre, F. Xhafa, Exploring Predictability in Mobile Interaction, in: Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on, IEEE, 2012, pp. 133–139. doi:10.1109/EIDWT.2012.29.
URL http://dx.doi.org/10.1109/EIDWT.2012.29

[23] M. A. Khan, H. Tembine, A. V. Vasilakos, Game dynamics and cost of learning in heterogeneous 4g networks, IEEE Journal on Selected Areas in Communications 30 (1) (2012) 198–213. doi:10.1109/JSAC.2012.120118.