Lightweight Self-organizing Reconfiguration of Opportunistic Infrastructure-mode WiFi Networks

Daniel J. Dubois[1], Yosuke Bando[1,2], Konosuke Watanabe[1,2], Henry Holtzman[1]
[1]MIT Media Lab, Cambridge, MA, USA
[2]Toshiba Corporation, Tokyo, Japan
ddubois@mit.edu, yosuke1.bando@toshiba.co.jp
konosuke.watanabe@toshiba.co.jp, holtzman@media.mit.edu

***Abstract*—The purpose of this work is to provide a method for exploiting pervasive wireless communication capabilities that are often underutilized on smart devices (e.g., phones, tables, cameras, TVs, etc.) in an opportunistic and collaborative way. This goal can be accomplished by sharing device resources using their built-in WiFi adapter. In this paper we explain why the standard ad-hoc mode for building mobile peer-to-peer networks is not always the best choice and we propose an alternative self-organizing approach in which an opportunistic infrastructure-mode WiFi network is built. The particularity of this network is that each device can either be an access point or a client and change its role and wireless channel over time. This contribution advances the state of the art by using a context-aware approach that considers actual frequency allocation to other devices and monitored traffic. We finally show that our approach increases the average speed for delivering messages to a level that in several situations outperforms previous work in the area, as well as a simple single-channel ad-hoc WiFi network.**

Keywords*—*delay-tolerant networks; peer-to-peer; self-organization; WiFi; performance

I. Introduction

In the last years we have seen an increase in the number of smart devices that are equipped with wireless communication capabilities and storage. These devices may be very different, specialized, and pervasive such as phones, tablets, cameras, TVs, printers, etc. Another recent example is the FlashAir SD Card [25], which is an SD card equipped with a tiny WiFi adapter and a tiny web server for sharing its storage via WiFi: this can virtually be integrated in any device that accepts SD card technology. We envision that tiny devices like the one described above are likely to become more common in the future and therefore there is an increasing industrial effort to find innovative ways for exploiting them. In this work we propose a self-organizing policy for exploiting the radio capabilities of such devices. The idea is to share them when they are not in use to create opportunistic wireless networks. The purpose of these networks is to provide an infrastructure for sharing data and services for situations in which a preexisting 3G/4G network or WiFi hotspot is too slow or cannot be accessed. Examples of situations are areas with bad coverage, overcrowded areas, rural areas, or areas in which other networks are too slow, expensive, or filtered.

The problem we solve is the self-organizing creation, maintenance, and optimization of these opportunistic WiFi networks. WiFi can operate in two modes[1]: (*i*) *infrastructure mode* in which the devices are divided into *access points* that can typically receive connections and *clients* that can only connect to access points; (*ii*) *ad-hoc mode* in which the devices can both connect and receive connections from other devices as long as they are in range to each other. Ad-hoc networks have been designed for a scenario that is similar to the one we consider: opportunistic mobile peer-to-peer networks; however in the real world ad-hoc networks have the following limitations: (*i*) all the devices of an ad-hoc network must use the same fixed wireless channel and therefore do not scale well [29]; (*ii*) ad-hoc mode consumes more power [20]; (*iii*) most mobile devices do not support ad-hoc mode (e.g., iOS-based and Android-based) or do not achieve full speed without user modification for their firmware [26], [30]. Due to these limitations we propose a solution that is able to exploit the higher speeds and compatibility of infrastructure-based WiFi networks. Moreover, our solution with multiple access points makes it possible to have different parts of the network on different WiFi channels, thus increasing the speed bounds.

The solution we propose will give each node the possibility to decide which WiFi channel to choose and which role between access point and client. Since an infrastructure-mode network is partitioned into different subnetworks that are managed by a single access point, at a certain moment the network may be composed of subnetworks that possibly have different frequencies. For this reason our approach consists of a policy (i.e., a set of rules) to stimulate opportunistic contacts for transferring information across nodes in different partitions of the network. The fact that the network may be temporarily partitioned has led us to model it as a delay-tolerant network. The capability for the network to be delay-tolerant is useful in a dynamic settings in which network devices are not reliable and can appear, disappear, and move in an unpredictable way. To evaluate our approach we use the ONE simulator [17]. The obtained results show that in the considered scenarios our policy is stable and more efficient than previous work in the area and, thanks to the possibility to use multiple channels, even better than a single-channel ad-hoc network.

This paper is organized as follows. Section II explains the context of this work. Section III discusses some related approaches in the literature. In Section IV we formalize our system and the problem we want to solve. Section V describes and analyzes the solution we propose. In Section VI we evaluate our solution in different scenarios. Finally, Section VII concludes the paper.

This work has been partially funded by the Fondazione Fratelli Rocca (Progetto Roberto Rocca fellowship – MIT-Italy Program).

[1]Specialized devices can operate in additional modes not cited here, such as using complex Wireless Distribution System configurations. However we do not consider them because they are typically vendor-specific and not always available in commercial smart devices.

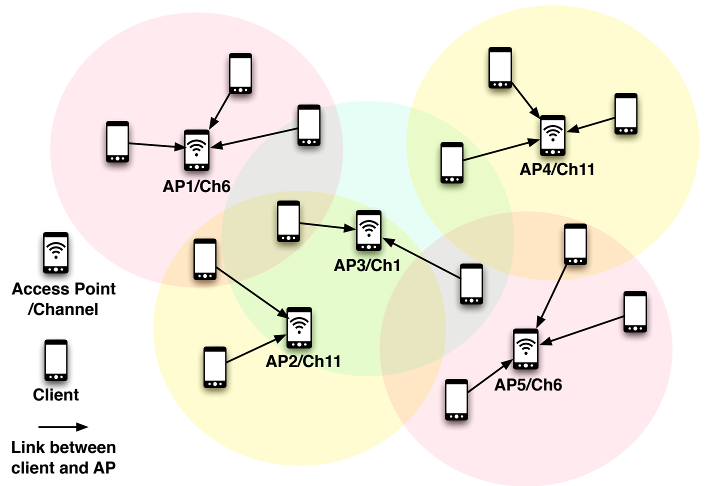

Fig. 1: Examples of opportunistic WiFi networks. White circles represent the access points coverage, shadows represent the area reached by client transmissions.

II. Context and Motivation

The purpose of this work is to support the spontaneous opportunistic creation of a wireless network infrastructure for storing and exchanging information among different devices. The type of network we consider in the context of this paper is WiFi (IEEE 802.11-family of networks [14]), in particular its *infrastructure mode* in which each radio interface may either be an access point (AP) or a client: APs can be connected to many clients, while clients may be connected to one access point only. The main reasons why we have opted for a model that resembles WiFi in infrastructure mode and not other models such as Bluetooth, WiFi-direct, or WiFi in ad-hoc mode are the following: the first is that all of them require to operate on the same frequency range, thus leading to scalability issues at frequency level in case of large number of devices; the second is that for security reasons most devices (e.g., Android-based devices, which are equipped with the most common mobile operating system [23]) either require unauthorized firmware modifications or continuous user confirmations to establish connections; last, but not least, none of the considered alternatives is able to achieve the theoretical speed of 54-600Mbps of 802.11n in infrastructure-mode. Since we want a system that has the capability to scale and optimize the use of available frequencies without user intervention, we opted for WiFi in infrastructure mode. The advantage of WiFi in infrastructure mode is that different parts of the network may use different channels (each channel corresponds to a range of frequencies), therefore it is possible to extend the network indefinitely by avoiding overlapping channels in adjacent parts, this way some channels may be simultaneously used in different parts of the network without interference. Another important motivation that is only available in infrastructure-mode is the possibility to exploit existing WiFi access points when they are open to everybody. This gives an additional opportunity to create WiFi opportunistic networks in areas that have no available frequencies or in buildings that already have open WiFi access points bridged to all the rooms (e.g., some hotels, universities, etc.). In Figure 1 we can see a simple WiFi network with 5 access points, several associated clients, each one operating on different non-overlapping channels of the 2.4GHz band (Ch1, Ch6, Ch11). The disadvantage of WiFi in infrastructure mode with multiple channels is that each part of the network has a star topology with an AP in the center that is disconnected from the other APs. To overcome this issue we rely on the possibility that devices may change role or association overtime (i.e., change from AP mode to client mode and vice-versa, or change association from an AP to another in range): this way, although the network at a certain moment may still be partitioned, its traversing information can wait on a device until new paths appear to reach its destinations. The possibility of storing and forwarding a piece of information when an end-to-end path does not exist require the network to be delay-tolerant, thus capable of using delay-tolerant protocols for exchanging information. In WiFi infrastructure-mode each device has the possibility to scan for the presence of APs in range. For each AP in range each device can detect the channel used and the signal level. It can also choose to act as a client of an existing AP (thus using the same channel), or it can choose to become a new AP on an arbitrary channel. In some devices it is also possible to decide the power used to transmit data, however we intentionally ignore this possibility since it was proven that in recent devices small negligible savings in energy result in significant drops of the actual signal [9], [21], [13]. In this paper we want to address the problem of creating opportunistic networks using the WiFi infrastructure-mode scenario described above, thus considering its limits in radio range, the possibility to scan for nearby devices, and the possibility to choose the role (AP vs client) and the channel.

III. Related Work

In the following subsections we discuss some related work in the areas that are relevant to the context of this work.

A. Distributed Frequency Allocation

The problem of allocating frequencies to a wireless network is a common case study of graph coloring problem [12], which has been proven to be NP-hard [10]. Due to the difficulty of the problem all the scalable solutions are based on heuristics. For example greedy approaches in which the solution is built step-by-step by maximizing an utility function [2], game theory approaches in which the problem is modeled as a game [6], [28], auction theory approaches in which the different components of the system may coordinate by bidding and bargaining frequencies [16], [5], [18], and approaches based on collaborative multi-agent systems [22]. One common point of these approaches is that they solve a theoretical problem that is often oversimplified with respect to the variability and complexity of a real system. In a white paper from Cisco Systems [7], the authors state that spectrum problems often come from external sources instead of from interferences generated by the network. These external interferences cannot be usually detected or predicted by normal network devices and therefore there is a need to go beyond the simple spectrum allocation problem. In our work we consider that the actual and historical link speeds can be used as estimators of external performance degradation phenomena that cannot be directly detected. For this reason we extend a greedy model based on the *weighted aggregate interference* [2] in which the utility function also considers the historical speed information measured by the nodes.

B. Resource Sharing in Distributed Opportunistic Networks

Since the frequency spectrum is a shared resource, it is worth citing some additional work whose goal is to facilitate the sharing of generic resources. P2PCS [3] is an approach for sharing resources in a distributed IaaS cloud computing system using resource partitioning. Talipov *et al.* [24] propose another approach for sharing and discovering resources in smartphone delay-tolerant networks based on mobility prediction using Markovian models. MobiShare [31] uses a different mobility profiling and prediction approach based on clustering to create an opportunistic cloud for sharing resources among mobile peers. Other approaches such as [27], [19] try to optimize non-functional properties of these systems by deciding where the resources are allocated [19] and how the load is balanced among them [27]. These approaches are all based on several techniques for optimizing resource sharing based on mobility and traffic prediction patterns and routing optimization. In our approach we are transparent with respect to how the information is routed and we focus more on optimizing how the network infrastructure is deployed and configured rather than how generic resources are assigned, therefore all the above techniques can be considered complementary to ours and some of them may be integrated as a future work.

C. Infrastructure-mode WiFi Reconfiguration

The need for augmenting the capabilities and reconfiguring WiFi networks has been widely recognized both from academia and industry. One of the most known efforts are the so-called WiFi *mesh networks* [1], that will be defined in the new upcoming standard IEEE 802.11s [4]. The problem of these upcoming standards is that they are still far from being integrated in common commercial smart devices. What follows is a description of some of the current efforts with respect to infrastructure-mode WiFi reconfiguration. Zhang *et al.* propose a hybrid network infrastructure in which some network devices move and behave as peer-to-peer ad-hoc devices, while other devices are part of the fixed infrastructure and do not move. Their infrastructure is complemented with algorithms that maximize the probability of delivering information. MA-Fi (Mobile Ad-Hoc WiFi) [30] is another networking strategy in which network devices create mesh networks using WiFi infrastructure mode. This work assumes that it is possible to instrument the MAC level of a WiFi interface in such a way that some access points may also act as clients. WiFi-Opp [26] builds an ad-hoc network using infrastructure-mode by alternating the roles of the network devices between access point and client. They use an approach that does not consider speed and frequencies of the links, but relies instead on three fixed parameters: (*i*) the duration of time for which a client device is allowed to scan; (*ii*) duration of time for which a device stays an access point when it does not have associated clients; (*iii*) duration of time a client stays connected to the same access point. In our work we are using the same idea of alternating nodes between client mode and access point, but we improve the accuracy of the decisions by taking into account also frequencies and network traffic. Finally, Kärkkäinen *et al.* extend the concept of WiFi-Opp to use also public open WiFi hotspots as part of the opportunistic network. [15].

IV. System Model and Problem Description

In our model we assume that all the devices use a WiFi protocol with the same domain of frequencies. Client nodes may also scan for AP nodes at any time and get an estimation of their signal level and on the actual speed of the link once they are associated. We also consider that the evolution of the system from a state to the next depends on both controllable factors (e.g., frequency, AP role vs client role, associated AP for a client, etc.) and non-controllable factors (e.g., network traffic, appearance/disappearance/movement of devices, discrepancies between signal quality and actual link speed). We also assume that time is discrete when modeling system evolution.

A. State of the System

The state of the system under consideration for a given instant of time t is defined by the following parameters.

$N = \{N_1, N_2, ..., N_{nCard}\}$ is a set of $nCard = |N|$ nodes, where each node represents a wireless device.

$F = \{F_1, F_2, ..., F_{fCard}\}$ is a set of $fCard = |F|$ non-overlapping frequencies that can be used by any node.

$isAP : N \rightarrow \mathbb{B}$ is a predicate that is true iff a node is an AP. To keep expressions simple we define $A := \{x | x \in N \wedge isAP(x)\}$ as the set of actual AP nodes.

$isClient : N \rightarrow \mathbb{B}$ is a predicate that is true iff a node is a client. To keep expressions simple we define $C := \{x | x \in N \wedge isClient(x)\}$ as the set of actual client nodes.

$freqAP : A \rightarrow F$ is the function that associates an AP node to a frequency.

$remoteAP : C \rightarrow A$ is a function that associates a client to the AP it is connected to.

$isVisible : N \times N \rightarrow \mathbb{B}$ is a predicate that is true when a node is visible to another node, meaning that one can read the information about the other and negotiate a connection.

$signal : N \times A \rightarrow \mathbb{R}[0,1]$ is a function that associates the level of the radio signal between a generic node and an AP.

$speed : C \times A \rightarrow \mathbb{R}[0,1]$ is a function that returns the measured percentage of bandwidth that can be used by a client connected to an access point. A value of 100% means that the full theoretical bandwidth is actually available.

System Invariants

$\forall n \in N, \neg(isAP(n) \wedge isClient(n))$ (a node cannot be AP and client at the same time).

$\forall n1, n2 \in N, isVisible(n_1, n_2) \Leftrightarrow isVisible(n_2, n_1)$ (nodes are always mutually visible).

$\forall n \in N, a \in A, signal(n, a) > 0 \Rightarrow isVisible(n, a)$ (APs whose signal is positive are always visible).

$\forall c \in C, a \in A, a = remoteAP(c) \Rightarrow isVisible(a, c)$ (connected nodes are always visible).

B. Evolution of the System

The evolution of the system is defined as the change of its parameters as long as system invariants hold. The controllable parameters of the system (input that may be directly controlled by a node-level policy) are: *isAP*, *isClient*, *freqAP*, *remoteAP*. F (frequency domain) is the only parameter that is not allowed to change overtime. The environmental parameters that cannot be directly controlled (environmental conditions and feedback responses to changes in the controllable parameters) are: N, *isVisible*, *signal*, and *speed*.

Additional information about the evolution of the system is provided by output parameters, i.e., the parameters that can be updated at node level at run-time using just local observations. The output parameters we use are the following (defined for each node n at a given time t).

generatedTraffic $: N \rightarrow \mathbb{R}^+$ associates $n \in N$ to the total amount of generated data that has n as its destination.

deliveredTraffic $: N \rightarrow \mathbb{R}^+$ associates $n \in N$ to the total amount of data actually delivered to n.

$latency_{\Delta w} : N \rightarrow \mathbb{R}$ associates $n \in N$ to the average of the delays of each message delivered to n in the last time window Δw. Each delay is defined as the difference between the time in which the message has been delivered (e.g., it has reached its final destination) and the time in which it was generated.

Seen : set of all the nodes previously encountered by $n \in N$ since the system was started. A node $n_1 \in N$ encounters another node $n_2 \in N$ iff $isVisible(n_1, n_2)$.

lastTime : $Seen \rightarrow \mathbb{N}$: is a function that associates a node previously encountered by n_0 to a timestamp that indicates the last time the two nodes were connected.

lastSpeed : $Seen \rightarrow \mathbb{R}[0,1]$: is a function that associates a generic node n_1 seen in the past to the historical aggregated relative speed observed from previous interactions with the local node (e.g., $speed(n_1, n_2)$, where $n_1 \in C$, $n_2 \in A$, and either n_1 or n_2 is the local node).

Derived Functions

$$throughput : \sum_{n \in N} \frac{deliveredTraffic_t(n) - deliveredTraffic_{t-\Delta w}(n)}{\Delta w}$$

is the total throughput of the system measured in the last time window Δw, that is the rate of successful messages delivery to their final destinations nodes. The time window Δw is assumed to be constant.

$$deliveryRate : \frac{\sum_{n \in N} deliveredTraffic(n)}{\sum_{n \in N} generatedTraffic(n)}$$

is the percentage of the total generated traffic of the system that has actually been delivered to its final destination node. A theoretical value of 100% means that all the generated traffic has been delivered.

$$averageLatency : \frac{\sum_{n \in N} latency_{\Delta w}(n)}{|N|}$$

is the average latency of the network in the last time window Δw, which is the average time that is needed to deliver a message from the source node to the final destination node.

C. Problem Statement

The problem we want to solve is to build an infrastructure that facilitates the transfer of information among mobile devices. In analytical terms our main goal is to *optimize the following objective functions*:

$Objective\ 1 : maximize(throughput)$
$Objective\ 2 : maximize(deliveryRate)$
$Objective\ 3 : minimize(averageLatency)$

The maximization of the throughout ensures faster data transfers, which is particularly important for large data. The maximization of the delivery rate ensures that more data are actually delivered and not discarded or enqueued. Finally, the minimization of the average latency ensures that data are delivered in a timely manner, regardless of size, which is important for frequent exchanges of small data.

Since *deliveredTraffic* and *latency* cannot be directly derived from input parameters at node level, to solve this optimization problem we cannot use model information directly. The main reason is the fact that we do not have enough information about the feedback function of the system, nor we can predict environmental changes in a reliable way.

V. Self-organizing Policy

In this section we propose a decentralized policy for self-organizing the system, defined in terms of local rules executed by all the nodes. In particular, we show how to exploit some local properties of the system to increase the likelihood of optimizing our objective functions by just modifying controllable parameters.

A. Effects of controllable parameters

Due to the variability and complexity of the system, the exact response function that maps input parameters to output parameters (effects) is not known because information about traffic and topology modification cannot be predicted in a reliable way in our scenario. However, changes in the input parameters may intuitively increase the likelihood of a desired change. We modify such parameters first to increase the probability of establishing new opportunistic connections among nodes that have never communicated before (or that have not communicated for a long time), second to optimize the speeds of the links by using different wireless channels for different parts of the network. The effects on the input parameters that we consider are the following.

Effects of changing the wireless channel of an AP node. Changing the AP channel has the effect of switching to a channel that can be less busy or with less disturbances. Moreover, it causes connected clients to reconfigure, possibly replacing some of the existing end-to-end paths that include the AP. Frequent changes in end-to-end paths may increase the probability of contacts between nodes, thus facilitating information transfers and reducing delays. To decide how to change the channel we will use information about frequencies used by nearby APs and historical information about the average speeds of previously known APs. The idea is that if we exclude frequencies already used by nearby APs, we can avoid speed reductions due to interferences. Moreover, recent historical speeds give an idea of the possible performance of potential reconnections.

Effects of changing client node association. Changing the AP of a client node has the primary effect of establishing a contact with a new node. Trying to connect to new nodes will lead to the creation of new end-to-end paths, similarly to the effects of changing the channel of an AP node. It is also possible to choose an access point with a higher signal or with higher average speeds measured during previous interactions. The decision of which AP to associate should consider the fact that a new (first preference, leading to new contacts) or high-signal (second preference, leading to higher transfer rates) node may be a better choice when changing client association.

Effects of promoting a client to AP. The primary effect of creating a new AP node is that existing clients may detect it and connect to it. Nodes can only be discovered when they are APs, therefore all the nodes should periodically become APs. Another important effect is that having more APs can maximize the use of channels, thus increasing the overall system bandwidth. The choice to become an access point will be based on the number of nearby APs (i.e., if there are no access points nearby, the current node will be the first) and on the need of opportunistic connections. For example a client node that had recent interactions with all the available APs may decide to become an AP in order to be detected by other, possibly new, unknown clients.

Effects of demoting an AP to client. If an AP is underutilized (i.e., it does not have any associated clients) and there is another AP in range, it can become a normal client, thus freeing a channel and creating a new connection. Another reason for an AP to become a client is to be able to discover and actively connect to previously unknown nodes, thus increasing the probability of opportunistic connections among different partitions of the network.

B. Utility Functions

In our policy we will use two utility functions to support node-level decisions on the input parameters. Each function will be evaluated at node n_0, which denotes the local node.

$$FreqValue(f) : F \to \mathbb{R}[0,1] := \frac{1}{1+\sum_{z\in A \wedge freqAP(z)=f} signal(n_0,z)}$$

The *FreqValue* utility function is used to determine the utility value of a frequency f, as seen by node n_0. Under the assumption that the signal metric we use is proportional to the maximum achievable speed, the value of *FreqValue* estimates the maximum speed, when all the other access points in range try to transmit simultaneously. The formulation is based on the concept of *Weighted Network Utility Function* proposed and proved in [2], but normalized between 0 and 1 and using our *signal* metric as weight instead of the transmission power. To understand intuitively the formula we can see that the denominator is composed of two parts: 1 (interference caused by n_0, which has weight 1) and the sum of the interferences caused by other access points. For example, if we have only another AP in range on the same frequency with maximum signal, *FreqValue* is 0.5; if the existing APs are three with maximum signal, the *FreqValue* is 0.25.

$$APvalue(x) : A \to \mathbb{R}[0,1] := \begin{cases} lastSpeed(x), \text{if } lastTime(x) > Now - \Delta H \\ FreqValue(freqAP(x)), \text{otherwise} \end{cases}$$

The *APvalue* utility function is used to determine the value of a link between n_0 and an existing access point $x \neq n_0$. The first reason that justifies this function is that we assume that there is a correlation between the historical monitored speed of an access point and the future speeds, as discussed and empirically shown in [32]. Moreover, we also assume that historical monitored speed is a more reliable estimator for speed than the *FreqValue* of the frequency because it is a direct measure from the environment rather than a theoretical derivation that assumes an ideal scenario [7]. The actual value of $APvalue(x)$ depends on the available historical speed information in n_0 regarding x in the last period of time ΔH. If such information is not available because the node has not been seen in such a period of time from now, then we estimate the speed based on the frequency of the access point, using the *FreqValue* utility function. In other words, if the current node has historical information on the link speed (expressed as a percentage), then the value of the AP is equal to its link speed. If such information is not available, then the value is equal to the value of the AP's frequency. Since we consider that interferences are proportional to the signal level, we use the signal level as a weight for each AP. Please note that ΔH is a policy parameter that will be defined in the next subsection.

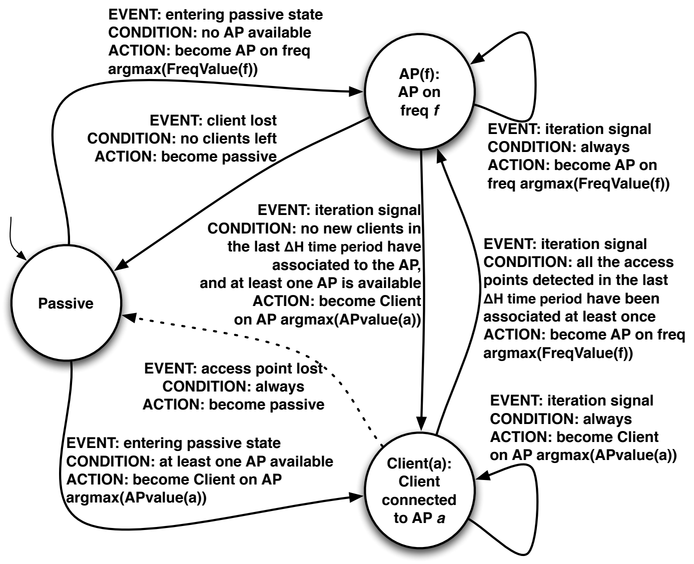
EVENT: entering passive state
CONDITION: no AP available
ACTION: become AP on freq argmax(FreqValue(f))
AP(f): AP on freq f
EVENT: client lost
CONDITION: no clients left
ACTION: become passive
EVENT: iteration signal
CONDITION: always
ACTION: become AP on freq argmax(FreqValue(f))
EVENT: iteration signal
CONDITION: no new clients in the last ΔH time period have associated to the AP, and at least one AP is available
ACTION: become Client on AP argmax(APvalue(a))
EVENT: iteration signal
CONDITION: all the access points detected in the last ΔH time period have been associated at least once
ACTION: become AP on freq argmax(FreqValue(f))
Passive
EVENT: access point lost
CONDITION: always
ACTION: become passive
EVENT: iteration signal
CONDITION: always
ACTION: become Client on AP argmax(APvalue(a))
EVENT: entering passive state
CONDITION: at least one AP available
ACTION: become Client on AP argmax(APvalue(a))
Client(a): Client connected to AP a

Fig. 2: Parametrized non-deterministic automaton that represents the self-organization algorithm.

C. The Algorithm

The input parameters for the algorithm, when executed in node n_0 are:

- State of the system, as defined in Section IV.
- ΔT, which is the average period of time between algorithm iterations.
- ΔH, which is the maximum amount of time that information about past nodes interactions $(Seen, lastTime, lastSpeed)$ is kept in memory.

The algorithm that implements the self-organization policy at node level can be modeled as the non-deterministic automaton depicted in Figure 2. Each circle in the figure represents a particular node state, which can be *Passive* (a state for nodes that are neither clients nor APs), *Client*, and *AP*. Plain arrows represent *Event-Condition-Action* (ECA) rules for moving from a state to a destination state. An Event-Condition-Action rule is activated when an event happens: in such a case, if the condition holds, then the action is executed. Dotted arrows represent unpredictable events that modify the state in a deterministic way (they have precedence over the other transitions). Circles with a parameter letter receive the parameter from the incoming transitions. Self-transitions can overwrite the state parameters. The parametric nature of the automaton is used to keep its representation concise (i.e., the proposed automaton is equivalent to one in which parameters are replaced by new states since parameters are defined over finite domains: $a \in A$ and $f \in F$). The transitions are evaluated at discrete time intervals Δt generated from an exponential distribution $Exp(1/\Delta T)$. When two transitions can be executed at the same time, only a random one is executed according, for example, to a uniform random probability distribution.

Transitions from Passive state. In passive state, a node can become an AP on the frequency $f \in F$ that maximizes the $FreqValue(f)$ utility function if no APs are available, or a client connected to an AP $a \in A$ that maximizes the $APvalue(a)$ utility function if at least one AP is available.

Transitions from AP state. In AP(f) state, a node can non-deterministically switch to frequency $f' \in F$ that maximizes the $FreqValue(f')$ utility function, or, when the node does not see new clients associated to it in the last time period ΔH and at least one other AP is available, the node can become a client connected to AP $a \in A$ that maximizes $APvalue(a)$ utility function. In the case an AP node loses its last client, it goes back to the passive state.

Transitions from Client state. In Client(a) state a node can non-deterministically switch to access point $a' \in A$ that maximizes the $APValue(a')$ utility function, or, when the node has already been associated at least once with all the APs discovered in the last time period ΔH, can become a new AP on frequency $f \in F$ that maximizes $FreqValue(f)$ utility function. In the case the client loses its access point, its state is immediately and non-deterministically changed to passive state.

D. Algorithm Analysis

Optimality discussion. The proposed policy is based on two utility functions that are periodically evaluated by each node to decide the next local action to optimize the whole system. In the previous subsections we have explained that $FreqValue(f)$ is a utility estimator to improve the usage of the available spectrum and thus to improve the overall network speed, while $APvalue(x)$ is a utility estimator for choosing neighbor nodes with higher potential speed. Due to the nature of these functions (e.g., the lack of global information and the inherent difficulty to calculate the actual global effect of each individual action) it is possible that, although individual nodes may improve their local performance, the global performance is stuck in a local optimum. To help explore more (possibly better) global system states and thus reduce the occurrence of local optima, we make use of non-deterministic choices which may lead to sub-optimal choices at local level, but can help the whole system at global level, similarly to what happens in evolutionary algorithms.

Communication Complexity. The proposed policy is executed at node level and does not involve any additional communication for coordination among the nodes other than the one required for association, scanning, and routing purposes. Therefore, for each reconfiguration period ΔT the communication overhead is O(1) since it does not depend on any parameter of the system. It is important to notice that complex routing protocols may require the exchange of more messages to possibly reconstruct routing tables. This additional overhead may be negligible when using a smart epidemic algorithm that does not need coordination, but it may be expensive when using a deterministic routing algorithm.

Computational Complexity. For each possible state, the node will simply evaluate some utility functions that may involve just the nodes that are visible and the frequencies. This results in an upper bound for the computational complexity of $O(|N'|)$, where N' is the set of neighbors of local node n_0. The explanation for this complexity is that it depends only on the *FreqValue* and *APvalue* functions: both of them depend on the number of neighbors, which results in a complexity of $O(|N'|)$.

Memory Complexity. The utility functions used by the algorithm depend on current information about the state and on some historical data, therefore the memory consumption is proportional to the quantity of historical information. Assuming that $\bar{I}$ is the average quantity of information produced in one time unit, then memory complexity is $O(\bar{I}\Delta H)$ since historical information is retained for ΔH time units by our algorithm.

VI. Evaluation

To evaluate our policy we use ONE [17], a simulator for delay-tolerant networks written in Java, which we extended to support infrastructure-mode WiFi with multiple frequencies. In our simulations we consider the scenario that we will describe in detail in Section VI-A and then we show how different parameters may affect the performance of our policy. The results of each experiment are averaged among 32 Monte Carlo simulations of the same experiment with different random number seeds. We then compare our results with the WiFi-Opp algorithm [26] described in Section III-C and with a generic WiFi network in ad-hoc mode. The WiFi-Opp algorithm has been chosen because it is based on the same technological assumptions of our work: the possibility to run the algorithm on commercially available mobile devices. This means that we assume that ad-hoc mode is unavailable without modifying the firmware of the device, and that a device can be access point only on the 2.4GHz band (even if it is capable of using the 5.0GHz band). Since in the WiFi-Opp algorithm the authors do not specify how the channels and access points are chosen, we assume that they are chosen randomly according to a uniform probability distribution. In the case of ad-hoc WiFi comparison we consider that the ad-hoc solution cannot use more than a single WiFi channel since the standard does not allow a multi-channel ad-hoc network without introducing proprietary extensions. To keep our evaluation independent from the routing approach we assume that the buffer to store delayed messages in each device is never full (this is reasonable since current devices have several GBytes of storage) and that messages are disseminated using the generic epidemic routing algorithm provided in ONE. Messages can be enqueued in the system for a maximum of 10 simulated minutes, otherwise they are dropped. To simulate interferences we use the Gupta-Kumar Radio Interference model [11] used by ONE. Each device has an unique ID (currently the MAC address of the internal WiFi chip). Such IDs are used to identify sources and destinations of messages in an independent way with respect to network topology, partitions, addressing, and routing. The actual network uses a different TCP/IP addressing space for each AP, thus the IP address of each device may change over time depending on its role (AP/client) and on the other devices it is connected to.

A. Setting

The evaluation setting considers a squared room with many people, each one carrying a mobile device and moving in a random direction, known as the *random waypoint* mobility model: each person starts from a random point of the room and decides to move to a destination point, then he/she will wait some time and decides a new destination point. All the devices periodically generate messages with a random destination among the people in the room and network performance is monitored for six simulated hours. In one experiment variation we have also used map-based movement using a subset of the Helsinki map provided by ONE.

B. Parameters

The input parameters of our simulations are the following. Reference values are used for the reference experiments, but the value of each parameter will be changed to different values to assess the impact of such parameter on the overall performance.

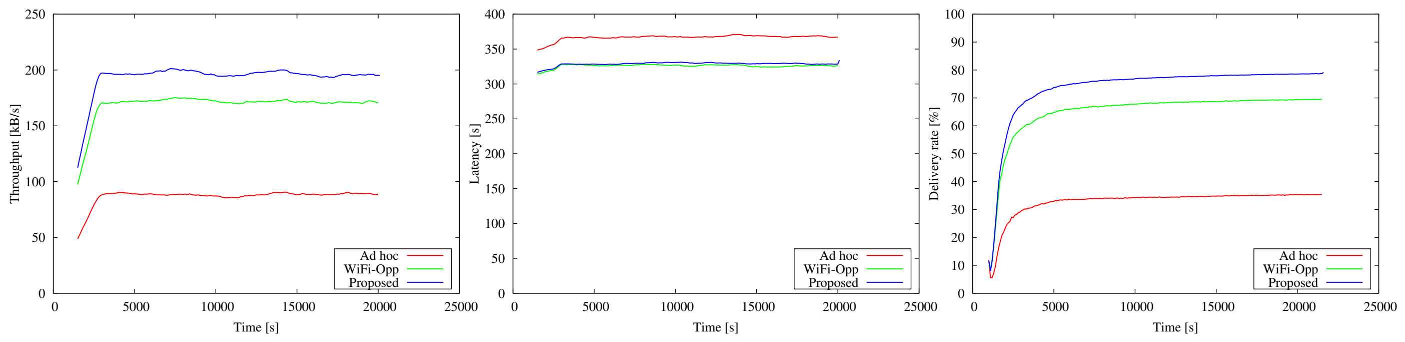

Fig. 3: Reference experiments. Plots show throughput, latency, and delivery rate.

Number of nodes. Total number of devices in the system, which will be constant. We assume a reference value of 100 nodes.

Mobility map. Environment in which the devices move. We consider an empty room of variable size (depending on the node density), which is 63×63 meters for the reference value of 100 nodes.

Node density. Total number of devices in a square meter. In the case of the empty room we assume a reference value of 0.025 nodes/m^2 that is equivalent to a room of 63×63 meters for 100 nodes.

Message size. Size of the generated messages. The reference value is uniformly distributed between 0.5MBytes and 1.5MBytes. We assume a time interval for generating new messages to be inversely proportional to the number of nodes (one message per four seconds when the nodes are 100) in order to keep the same level of network saturation.

Node speed. Speed of a device that moves among its waypoints. The reference value is uniformly distributed between 0.5 and 1.5 m/s.

Node wait. Time a device that has reached its destination waits before starting to move to a new destination. The reference value is uniformly distributed between 60 and 900 seconds.

Number of radio channels. Number of available non-overlapping radio channels. We assume a reference of 3 non-overlapping channels in the standard 802.11b/g/n on the 2.4GHz band.

Input parameters that will be kept fixed are the following. For the first four parameters, we use approximated measured values from a Samsung Galaxy Nexus smartphone. For the rest of the parameters, the values have been optimized for the reference experiment.

Maximum transfer speed. The maximum transfer speed that can be reached between two devices is 5MBytes/s.

Maximum range. The maximum communication range between two devices is 20 meters.

Transition time to AP. The transition time from non-AP status to AP status is 1 second.

Transition time to client. The transition time from non-client status to client status is 5 seconds.

ΔT. Average time between transition evaluations of our policy. We set it to 20 seconds.

ΔH. Amount of time that historical information about past interactions and device speeds are kept in memory. We set it to 150 seconds.

WiFi-Opp parameters. The parameters of the WiFi-Opp algorithm described in [26]: *t_scan* = 5 seconds (scanning time); *t_con* $\sim$ *Uniform*(10-30) seconds (client lifetime); *t_beac* $\sim$ *Uniform*(10-30) seconds (time that an AP is allowed to stay as an AP when it does not have clients).

The output parameters we evaluate are the throughput, the delivery rate, and the average latency, as defined in Section IV-B, which represent the objective functions. For the calculation of the throughput and of the average latency we consider a sliding window Δw=3000s.

C. Reference Experiment

In our reference experiment we compare our policy with WiFi-Opp and with the WiFi ad-hoc network. If we look at the throughput plot of Figure 3, we can see that the throughput stabilizes at 90kbytes/second in ad-hoc network, 175kbytes/second in WiFi-Opp, and 200kbytes/second with our policy. The reason why ad-hoc has lower performance is because it suffers interferences coming from the use of only a single channel, and because each node has a very large number of connections, which increases redundant traffic generated by epidemic routing. In the case of WiFi-Opp the performance is lower than ours because it does not use any smart utility function to choose the channel (AP mode) or access point (client mode). The reason why at stability we obtain values much lower than the theoretical throughput of 5MBytes/second is because the total bandwidth is shared among multiple nodes, moreover the nodes continuously move, and messages can be sent to opposite directions of the network, thus using radio channels of multiple parts of the network. It is important to notice that this high level of dynamism stresses the epidemic routing algorithm in such a way that it sends the messages several times from a node to the other, thus saturating the radio channels. If we look at latency and delivery rate plots of Figure 3, as the algorithms self-organize the system, an increasing percentage of messages is delivered, but with a slightly higher latency. The difference among the algorithms is explained by the fact that the throughput is different (lower in the case of ad-hoc network and WiFi-Opp with respect to our approach), therefore it takes more time to deliver the messages (higher latency), and less messages are delivered (lower delivery rate). In our results we will always show the latency and the delivery rate to have an idea of the amount and quality of traffic that is actually being forwarded.

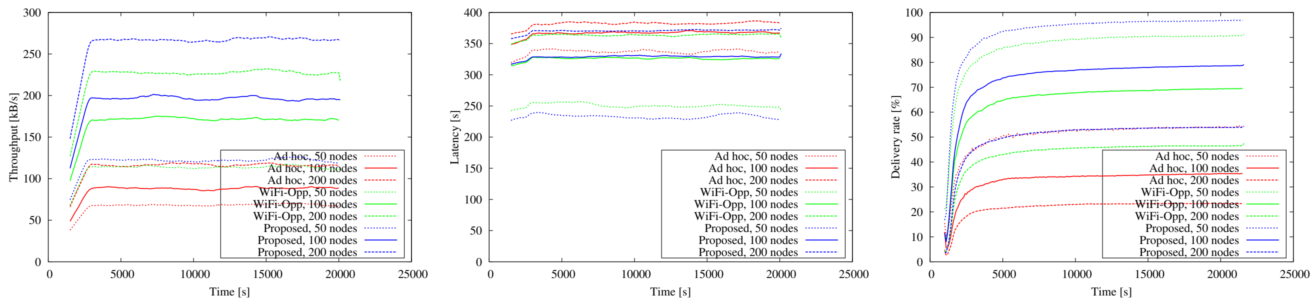

Fig. 4: Varying the number of nodes. Plots show throughput, latency, and delivery rate.

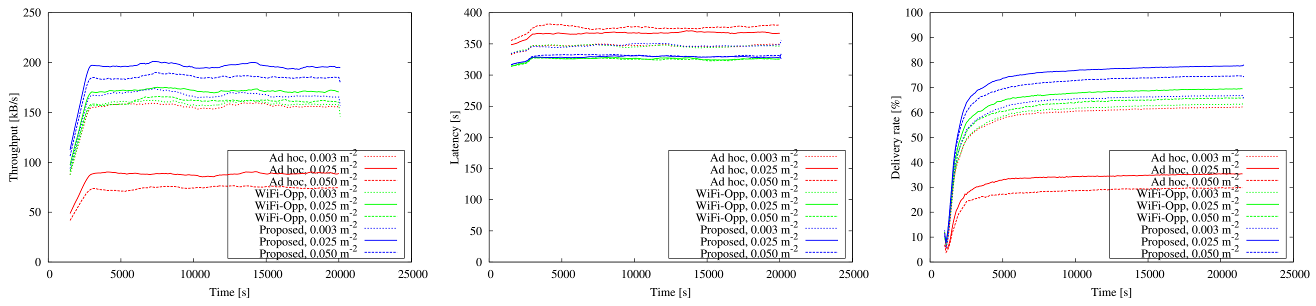

Fig. 5: Varying the density of the nodes. Plots show throughput, latency, and delivery rate.

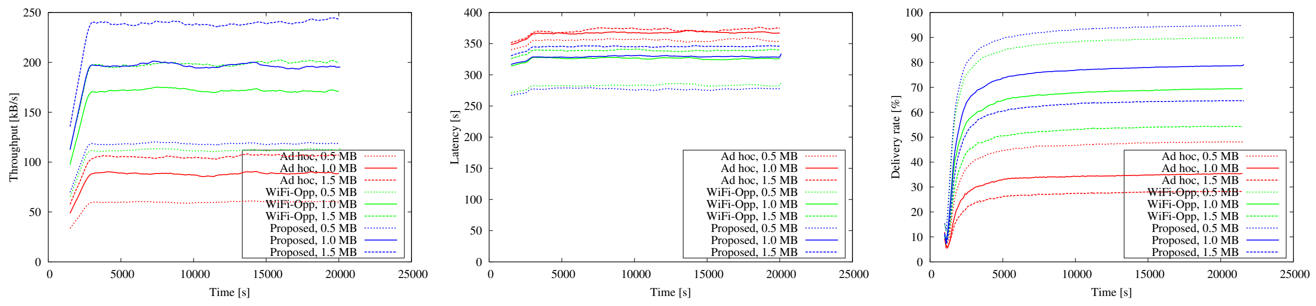

Fig. 6: Varying the message size. Plots show throughput, latency, and delivery rate.

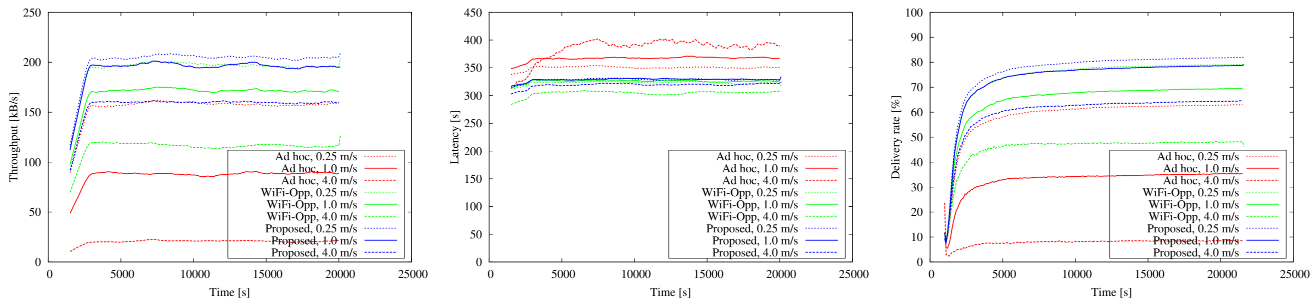

Fig. 7: Varying the node speed. Plots show throughput, latency, and delivery rate.

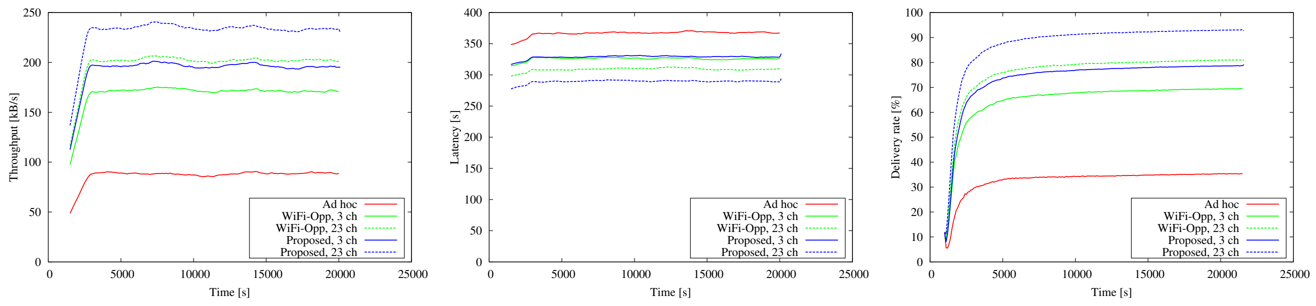

Fig. 8: Varying the number of channels. Plots show throughput, latency, and delivery rate.

D. Varying Input Parameters

In this subsection we describe the effects of variations on the input parameters on the network performance.

Varying the number of nodes. If we look at Figure 4 we can see that if we reduce the number of nodes from 100 to 50, or increase them from 100 to 200, our policy outperforms the others according to all our three objectives. In the case of node reduction we can see from the delivery rate plot that most messages are delivered, while almost half messages are lost in the case of node increase. A similar decrease in performance can also be observed in the throughput and latency plots. The explanation is that when the network is very large, the average length of the paths for sending messages will increase, therefore optimizing the system becomes more difficult.

Varying the density of the nodes. The throughput and latency plots of Figure 5 show that when we increase the density of the nodes from 0.025 to 0.05 nodes/m^2 with respect to the reference experiment, we obtain slightly worse results, and if we reduce the density from 0.025 to 0.003175 nodes/m^2, we again obtain worse results. The lower performance in the case of a dense network is because all of the three methods suffer from interferences, although ours and WiFi-Opp suffer less than ad-hoc because of the use of multiple frequencies. In the case of a sparse network the performance of our approach and WiFi-Opp is lower than the reference case because of the lower level of links among the nodes. In this case ad-hoc does not suffer from the reduced number of links at this density level and performs similarly to our approach since it does not have AP/client state transition overhead and can still maintain a much higher number of links.

Varying the message size. From the experiments reported in Figure 6 we can see an effect that is similar to the one observed when increasing/decreasing the size of the network in terms of the number of nodes. With smaller messages we can deliver more messages than with larger messages because the network is not saturated. With larger messages, the network saturates, thus dropping some messages, but increasing the throughput. The figure also shows that latency is not significantly affected. In any case (i.e., increasing and decreasing the message size), we can see that our policy performs better than the other two approaches, while differences within the same approach are similar for all three algorithms.

Varying the node speed. In this experiment we increase both the device movement speed and the time the device waits when it reaches the destination. The results of this experiment, reported in Figure 7, show that in our approach, a decrease of the node speed and waiting time does not have a significant impact on the throughput, while an increase of the same decreases the performance of all the algorithms (although ours still performs better). This is because, as the node speed increases, there is less time and thus chance to connect encountered nodes.

Varying the number of channels. This experiment shows the actual capability of our approach to exploit the different wireless channels. Even though using the channels in the 5.0GHz band is not actually supported by current commercial mobile devices, we want to show the behavior of the algorithm when we have the possibility to use the 23 non-overlapping channels of the 5.0GHz band. From Figure 8 we can see that with 23 channels, there is a significant performance improvement in both WiFi-Opp and our approach, with the latter still higher than the former. We can also notice that the delivery rate is almost optimal in our approach, while very low in the single channel ad-hoc approach.

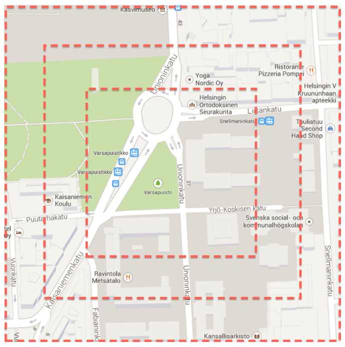

Fig. 9: Partial map of Helsinki. Dotted squares represent the different sub-maps we have considered in our experiments.

Experiments with a partial city map. In this experiment we run the simulator using the same parameters of the reference experiment, but with a real city map instead of an empty room. Devices move randomly along the streets of the map. The map we use is a subset of Helsinki shown in Figure 9, of different sizes, ranging from $200 \times 200\text{m}^2$ to $400 \times 400\text{m}^2$, enclosed in dotted squares in the figure. From the plots of Figure 10, we can see that our algorithm is always better than WiFi-Opp. In the comparison with ad-hoc, our approach is better up to $300 \times 300\text{m}^2$, then after $400 \times 400\text{m}^2$ ad-hoc becomes better. The explanation for this is that, in a large and thus less densely populated area, the advantage of ad-hoc (no AP/client overhead) surpasses the weakness of ad-hoc (poor spectrum utilization). This effect has already been seen in the node density experiment in Figure 5.

E. Discussion

In this section we have seen how our policy behaves when compared to WiFi-Opp and ad-hoc network approaches. From our exploration of the parameter space, we have seen that our approach outperforms the others in several significant scenarios, and the most important result is that, even though our approach is intuitively subject to delays due to AP/client transitions, in our scenario the observed behaviors are often better than an ad-hoc network with a single channel. We expect that the idea to prioritize the channels and the AP using utility functions can also be extended to new generation multi-channel ad-hoc networks and mesh networks, when they will be supported by commercial devices.

VII. Conclusions

In this paper we have considered the problem of creating opportunistic infrastructure-based WiFi networks for sharing information among WiFi-equipped devices to overcome the limitations of ad-hoc networks. In particular, we have proposed a policy that, when run on every device of the network, is able to reconfigure each device in such a way to create the network and keep it working and self-optimizing. In our evaluation, the simulations have shown that the algorithm is able to maintain the network efficient under different situations, with results that are comparable or even better than the ones obtained with ad-hoc networks.

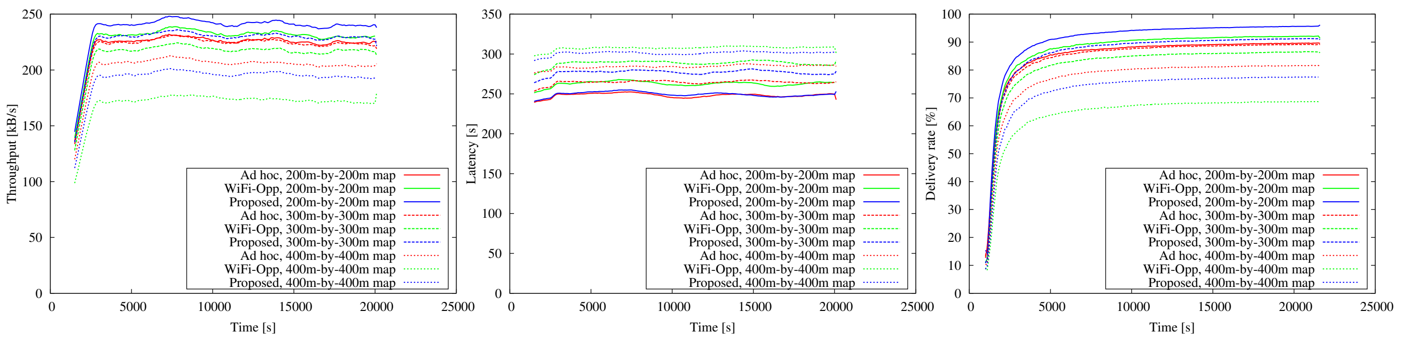

Fig. 10: Experiments with a partial city map of different sizes (Helsinki). Plots show throughput, latency, and delivery rate.

In the future we expect that, since we have made realistic assumptions regarding the characteristics of currently available devices (e.g., the possibility to use the Android-platform without any firmware modification, or the use of WiFi SD cards [25]), it will be possible to integrate this policy in an existing mobile middleware for mobile resource sharing such as [8]. Moreover, it would also be useful to investigate the possible benefits of allowing a hybrid approach in which a subset of the devices has access to an existing infrastructure (WiFi hotspot or 3G/4G network). In this case, such infrastructure may be exploited to build a bridge between two different partitions of the opportunistic network that have access to the same infrastructure (e.g., the Internet), thus facilitating the exchange of information and possibly reducing the delays. Another possible extension would be to extend the proposed algorithm to work with devices with multiple WiFi adapters. This scenario would result in a network with less partitions, and therefore with higher speeds since the chances of delayed communication are lower.

References

[1] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer networks*, 47(4):445–487, 2005.

[2] B. Babadi and V. Tarokh. GADIA: A Greedy Asynchronous Distributed Interference Avoidance Algorithm. *IEEE Transactions on Information Theory*, 56(12):6228–6252, 2010.

[3] Ö. Babaoglu, M. Marzolla, and M. Tamburini. Design and implementation of a p2p cloud system. In *ACM SAC '12*, pages 412–417. ACM, 2012.

[4] J. Camp and E. Knightly. The IEEE 802.11s extended service set mesh networking standard. *IEEE Communications Magazine*, 46(8):120–126, 2008.

[5] L. Cao and H. Zheng. Distributed spectrum allocation via local bargaining. In *IEEE SECON '05*, volume 5, pages 119–127, 2005.

[6] D. E. Charilas and A. D. Panagopoulos. A survey on game theory applications in wireless networks. *Computer Networks*, 54(18):3421–3430, 2010.

[7] Cisco Systems Inc. 20 Myths of Wi-Fi Interference. White Paper, 2007.

[8] D. J. Dubois, Y. Bando, K. Watanabe, and H. Holtzman. ShAir: An Extensible Platform for Wireless Resource Sharing on Mobile Devices. In *ESEC/FSE '13*. ACM, 2013.

[9] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *INFOCOM '01*, volume 3, pages 1548–1557. IEEE, 2001.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[11] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[12] W. K. Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

[13] D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall. Demystifying 802.11 n power consumption. In *HotPower '10*, page 1. USENIX Association, 2010.

[14] IEEE. Standard 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2012.

[15] T. Kärkkäinen, M. Pitkänen, and J. Ott. Enabling ad-hoc-style communication in public WLAN hot-spots. In *ACM CHANTS '12*, CHANTS '12, pages 31–38, New York, NY, USA, 2012. ACM.

[16] G. S. Kasbekar and S. Sarkar. Spectrum auction framework for access allocation in cognitive radio networks. *IEEE/ACM Transactions on Networking (TON)*, 18(6):1841–1854, 2010.

[17] A. Keränen, J. Ott, and T. Kärkkäinen. The one simulator for dtn protocol evaluation. In *SIMUTOOLS '09*, Simutools '09, pages 55:1–55:10, ICST, Brussels, Belgium, Belgium, 2009. ICST.

[18] C. Lacatus and D. Popescu. Adaptive interference avoidance for dynamic wireless systems: A game theoretic approach. *IEEE Journal of Selected Topics in Signal Processing*, 1(1):189–202, 2007.

[19] M. Peltomäki, J. Koljonen, O. Tirkkonen, and M. Alava. Algorithms for self-organized resource allocation in wireless networks. *IEEE Transactions on Vehicular Technology*, 61(1):346–359, 2012.

[20] G. Perrucci, F. Fitzek, and J. Widmer. Survey on energy consumption entities on the smartphone platform. In *IEEE Vehicular Technology Conference (VTC Spring '11)*, pages 1–6. IEEE, 2011.

[21] F. Piazza, S. Mangione, and I. Tinnirello. On the effects of transmit power control on the energy consumption of wifi network cards. In N. Bartolini, S. Nikoletseas, P. Sinha, V. Cardellini, and A. Mahanti, editors, *Quality of Service in Heterogeneous Networks*, volume 22 of *LNCS*, pages 463–475. Springer Berlin Heidelberg, 2009.

[22] G. Picard, M.-P. Gleizes, and P. Glize. Distributed frequency assignment using cooperative self-organization. In *IEEE SASO '07*, pages 183–192. IEEE, 2007.

[23] StatCounter GlobalStats. Top 8 Mobile Operating Systems from Apr 2012 to Apr 2013. http://gs.statcounter.com/#mobile_os-ww-monthly-201204-201304.

[24] E. Talipov, Y. Chon, and H. Cha. Content sharing over smartphone-based delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 12(3):581–595, 2013.

[25] Toshiba Corporation. FlashAir: SD Card with Embedded WLAN. http://www.toshiba-components.com/FlashAir.

[26] S. Trifunovic, B. Distl, D. Schatzmann, and F. Legendre. Wifi-opp: ad-hoc-less opportunistic networking. In *ACM CHANTS '11*, pages 37–42. ACM, 2011.

[27] G. Valetto, P. L. Snyder, D. J. Dubois, E. Di Nitto, and N. M. Calcavecchia. A self-organized load-balancing algorithm for overlay-based decentralized service networks. In *IEEE SASO '11*, pages 168–177. IEEE, 2011.

[28] B. Wang, Y. Wu, and K. Liu. Game theory for cognitive radio networks: An overview. *Computer Networks*, 54(14):2537–2561, 2010.

[29] X. Y. Wang and P.-H. Ho. Gossip-enabled stochastic channel negotiation for cognitive radio ad hoc networks. *IEEE Transactions on Mobile Computing*, 10(11):1632–1645, 2011.

[30] H. Wirtz, T. Heer, R. Backhaus, and K. Wehrle. Establishing mobile ad-hoc networks in 802.11 infrastructure mode. In *Proceedings of the 6th ACM workshop on Challenged networks*, CHANTS '11, pages 49–52, New York, NY, USA, 2011. ACM.

[31] K. Yadav, V. Naik, and A. Singh. MobiShare: cloud-enabled opportunistic content sharing among mobile peers. Technical Report IIITD-TR-2012-0009, IIIT-Delhi, Delhi, India, 2012.

[32] J. Yao, S. Kanhere, and M. Hassan. Improving qos in high-speed mobility using bandwidth maps. *IEEE Transactions on Mobile Computing*, 11(4):603–617, 2012.