Outsourcing Coordination and Management of Home Wireless Access Points through an Open API

Ashish Patro, Suman Banerjee
{patro, suman}@cs.wisc.edu
Department of Computer Sciences, University of Wisconsin Madison

***Abstract*—In dense wireless deployments at homes, such as apartment buildings, neighboring home WLANs share the same unlicensed spectrum by deploying consumer-grade access points in their individual homes. In such environments, WiFi networks can suffer from intermittent performance issues such as wireless packet losses, interference from WiFi and non-WiFi sources due to the rapid growth and increasing diversity of devices that share the spectrum. In this paper, we propose a vendor-neutral cloud-based centralized framework called COAP to configure, coordinate and manage individual home APs using an open API implemented by these commodity APs. The framework, implemented using OpenFlow extensions, allows the APs to share various types of information with a centralized controller — interference and traffic phenomenon and various flow contexts, and in turn receive instructions — configuration parameters (e.g., channel) and transmission parameters (through coarse-grained schedules and throttling parameters). This paper describes the framework and associated techniques, applications to motivate its potential benefits, such as, upto 47% reduction in channel congestion and our experiences from having deployed it in actual home environments.**

I. INTRODUCTION

Networking at homes continues to get complex over time requiring users to configure and manage them. Central to home networking infrastructure, are wireless Access Points (APs), that allow a plethora of WiFi-capable devices to access Internet-based services, e.g., laptops, handheld devices, game controllers (Wii, XBox), media streaming devices (Apple TV, Google TV, Roku), and many more. Standalone APs today are supplied by a number of diverse vendors — Linksys, Netgear, DLink, Belkin, to name a few. Given its central role in home networks, the performance and experience of users at homes depend centrally on efficient and dynamic configuration of these APs. In this paper, we argue for a *simple vendor-neutral API that should be implemented by commodity home wireless APs to enable a cloud-based management service that enables coordination between neighboring home APs, provides better performance, and reduces burden on users.*

A vendor-neutral API and a cloud-based management service for home wireless APs: In our proposed service, called COAP (Coordination framework for Open APs), participating commodity wireless APs (across different vendors) are configured to securely connect to a cloud-based controller (Figure 1). The controller provides all necessary management services that can be operated by a third-party (potentially distinct from the individual ISPs). We believe that our architecture is particularly applicable to large apartment buildings where one could envision that the apartment management contract with a single controller service (e.g., through a fixed annual fee) and all residents are asked to utilize the designated controller service within the building. This service would be no different than many other utilities, e.g., water, electricity, etc., which is arranged by the apartment management. Individual residents can also pick different controller services to realize many of our proposed benefits. *However, some advanced features, e.g., interference management and mitigation, are better served if neighboring APs participate through the same service.*

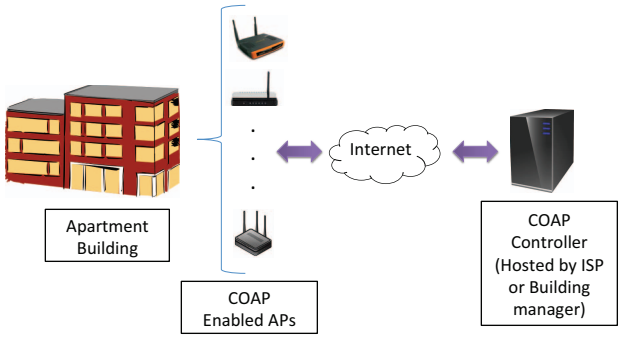

Fig. 1: An example of COAP deployment within a residential apartment building consisting of diverse COAP capable home APs and a cloud based controller.

Proprietary cloud-based solutions to manage *individual* home APs [7] exist today. However, such solutions do not attempt to leverage cooperation or coordination between neighboring APs. Our proposed COAP framework focuses on techniques to share various types of spectrum interference phenomenon and to enable cooperation between neighboring, participating APs and leads to significantly improved management, not otherwise possible. These techniques range from aggregating airtime utilization information from COAP APs to effect efficient channel assignments (§IV-A), using co-operative transmission schedules for interference mitigation (§V), to using "learning-based techniques" (§VI) to enable more sophisticated AP configuration algorithms. For example, if a non-WiFi device (e.g., microwave oven) is frequently used at certain times of day, the controller can *pre-emptively* configure neighboring COAP APs to other channels and mitigate interference (§VI).

Using Software-Defined approach in homes vs enterprise wireless networks: The concept of Software Defined Networking (SDN) for WLAN management, using a centralized controller for managing APs is gaining popularity in enterprise WLANs [26]. Further, a few commercial solutions, e.g.,

Meraki [3], provide vendor-specific proprietary cloud-managed service for APs in enterprise environments. We believe that a SDN based approach (using an open API) is also important in home environments where each wireless neighborhood has a diverse set of APs. But, unlike enterprise environments, homogeneity in such environments is likely hard to achieve. Furthermore, enterprise WLANs consist of a tightly managed control and data plane with very low latencies between the controller and the APs (order of micro-seconds). For home APs, it is only possible to create a loosely coupled control and data plane due to the Internet scale latencies between the APs and controller (tens of milliseconds). In this paper, we make the following main contributions:

- We provide an overview of the COAP framework which uses an open API and a cloud-based SDN controller to manage commodity WiFi APs in dense multi-dwelling apartment units (§III). The cloud-based controller enables co-operation between neighboring residential APs.
- Through a combination of real-world AP deployments and controlled experiments, we motivate the benefits of the COAP framework by describing two applications that benefit from co-operation between nearby residential APs: (i) efficient channel assignments resulting in *upto 47% reduction* in airtime utilization (§IV), (ii) mitigating wireless interference issues due to channel congestion or hidden terminal style interference resulting in more efficient airtime utilization due to *50% reduction in MAC losses* and improved quality for HTTP video flows (§V).
- We discuss how learning about prior non-WiFi and WiFi activity (e.g., client and traffic properties) using data collected by COAP cloud controller can help in predicting future activity. For example, *using 5 days of microwave oven activity* data from APs in an apartment building lead to high predictability of future activity (§VI-A). Also, using client device and traffic type information (§VI-C) resulted in *at least* $2\times$ *better accuracy* in predicting wireless traffic properties (e.g., session lengths). Such information can enable the COAP controller to pre-emptively configure home APs to mitigate interference.

II. MOTIVATION FOR A CENTRALIZED FRAMEWORK

Dense residential WiFi deployments today (e.g., apartment buildings) experience many problems during to the ad-hoc deployment of APs. We briefly describe the main observations from our prior study [19] with a deployment of 30 home APs.

Inefficient spectrum usage due to static Access Point configurations. More than 50% of the home APs amongst 300 APs observed during a 1 month period [19] were statically configured to use a single WiFi channel. This causes the APs to miss opportunities to use better WiFi channels to prevent the aforementioned problems. To compare relative activity across different channels at the 30 deployed APs, we used the traces to compute the fraction of total time with lowest airtime utilization amongst channels 1, 6, 11. These values were obtained from a secondary wireless card on each AP which hopped across channels 1, 6 and 11 in a round robin fashion (500 ms intervals). Figure 2 shows that some locations experienced highly uneven activity across channels (e.g., APs 5, 16 and 21). At these locations, one channel was least utilized for 80% of the time, indicating opportunity for more efficient channel assignments.

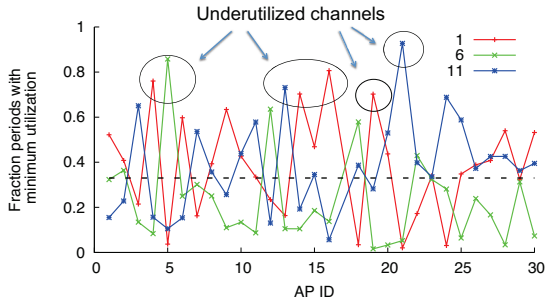

Fig. 2: Fraction of periods with the lowest airtime utilization on a particular channel (amongst channels 1, 6, 11) across different APs (6pm - 1am) across two weekdays.

WiFi Interference Home APs can experience high airtime utilization/channel contention due to factors such as high external traffic and/or use of low PHY rates by legacy clients. During periods of active traffic from such transmitters, the average airtime utilization at the neighboring APs increased upto 70% [19]. Such activity from a single AP can increase the congestion experienced by nearby APs operating on the same channel. Also, some links can occasionally experience hidden terminal (HT) style interference from nearby APs, resulting in packet losses at the clients.

Non-WiFi interference. Unlike the 802.11 protocol, some non-WiFi devices (e.g., microwave ovens, cordless phones) do not backoff to existing wireless activity on the same channel and can cause packet losses at the receiver due to overlapping transmissions with WiFi packets. In [19], the authors observed that the microwave ovens caused upto 80% degradation in link performance across multiple locations.

In each of these cases, If an AP is able to determine the wireless *context* (e.g., traffic information) at its neighboring APs and WiFi channels, it can determine the best remedial measure. Individual APs, however, do not have the full view of wireless activities, since each AP can only observe nearby activity on its current channel. If the entire set of nearby APs can aggregate this information at a COAP controller service, the latter can then instruct the participating APs on potential adaptations (§IV - §VI).

III. COAP FRAMEWORK AND API

We now present the details of the COAP framework that centrally manages and enables cooperation between APs across neighboring homes. COAP only requires a software upgrade for commodity home APs.

A. Framework details

To participate in the COAP framework, APs need to expose a vendor-neutral open API (Table I) to communicate with the COAP controller. COAP is complementary to SDN proposals ([27], [29]) that manage residential wired broadband networks.

Module	Description
APConfigManager	Receive configuration commands from the controller and execute them at the AP.
BasicStatsReporter	Report aggregate wireless statistics (airtime, link statistics, client and traffic information etc.) to the controller.
DiagnosticStatsReporter	Report detailed wireless statistics (MAC losses, non-WiFi activity) for diagnostic purposes.

TABLE I: Overview of the main functions implemented by the COAP APs. Our prior short workshop paper [18] presents the full COAP API and a detailed white paper is available on our project homepage [2].

To implement COAP, we extend the popular OpenFlow [15] SDN framework to implement the modules shown in figure 3. We implement the COAP controller modules over Floodlight [5] (an open-source Java-based OpenFlow SDN controller) to communicate with the OpenFlow module at the APs.

Our implementation consists of 9000 lines of code (LOC) for the controller, 4000 LOC for the COAP APs and 800 LOC to implement OpenFlow wireless extensions. Following are additional implementation details about the COAP framework:

Access Points. In our current implementation for OpenWrt [4] based Access Points, we use the open-source *click* [1] framework to implement the statistics gathering capabilities of the *BasicStatsReporter* and *DiagnosticStatsReporter* modules for the COAP APs. We use it to parse the packet headers of the local and neighboring WiFi links to obtain anonymized aggregate (or detailed) link level statistics. We use Airshark [22] to provide the non-WiFi device detection capability using commodity WiFi cards. The OpenFlow module at the COAP APs interfaces with Airshark and click-based modules (figure 3) using *netcat* and a standard messaging format. The netcat interface allows AP vendors to replace the click and Airshark modules with their own implementation for reporting statistics (e.g., using SNMP) described in Table I.

The *APConfigManager* module interfaces with the OpenWrt configuration tool (*luci*) to perform AP level configurations (e.g., channel, transmit power and traffic shaping). It is important to note that COAP is not dependent on the OpenWrt platform. It can interface with other configuration protocols (e.g., netconf [9]), depending on the AP platform.

Controller. The COAP controller is implemented over the Java based open source OpenFlow controller, Floodlight, and currently runs on a standard Linux server. All COAP controller modules, *StatsManager, COAPManager and ConfigManager*, are implemented as modules within Floodlight.

All server modules and tasks are managed by the *COAPManager* module, through which it manages and configures the connected APs. In the COAP framework, the COAPManager module allows administrators to implement custom policies based on their requirements. The server also maintains logs about previous wireless activity to learn and build models to predict wireless usage characteristics (§VI).

Wireless extensions for OpenFlow. The OpenFlow communication protocol currently consists of capabilities to exchange switch related statistics (e.g., statistics per switch port, flow, queue, etc.). We augmented this feature to exchange COAP related wireless statistics. To transmit wireless configuration updates from the controller to the APs (e.g., switch channel, throttle airtime), we extended the OpenFlow protocol to use a mechanism analogous to the one used to send OpenFlow switch configuration updates.

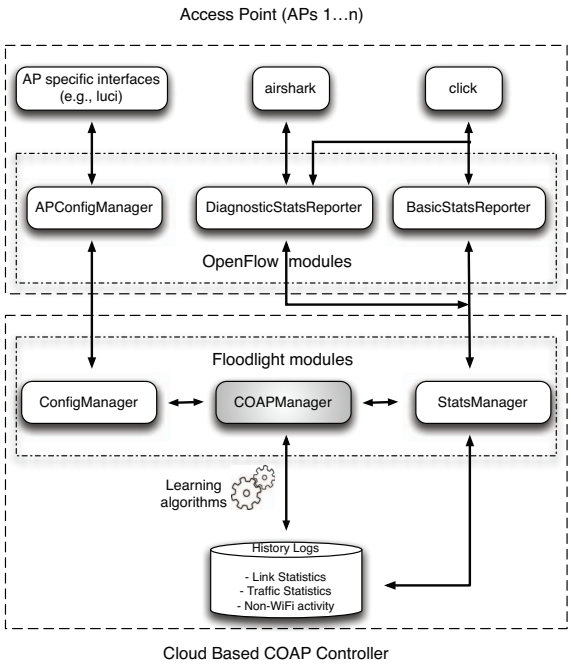

Fig. 3: Access Point and controller components of the COAP framework. The "COAPManager" is the main module which manages other modules at the controller.

Deployment and Traces. We deployed 12 OpenWrt based COAP APs in a large apartment building for more than 8 months to gather statistics as well as experiment with the basic management capabilities. They were used as private APs inside individual apartment units. We complemented this effort with a local testbed to perform additional experiments for this paper. We also analyze the traces from our earlier measurement study using 30 home APs [19] to motivate COAP based applications. The traces can be downloaded from our project homepage [2].

IV. LEVERAGING COOPERATION ACROSS HOME APS THROUGH COAP

To get maximum benefits from the framework, a single controller should be used for managing COAP APs within a multi-dwelling residential unit. We now discuss two such applications – channel assignment and airtime management.

A. WiFi channel configuration

In enterprise WLANs, AP configurations (e.g., channel, transmit power) are managed by a vendor specific central controller. In homes, such configurations are usually performed manually by users or done based on local-only AP observations. This can lead to inefficient utilization of the spectrum (§II).

In the COAP framework, such functions (e.g., channel assignment) can be delegated to the controller. For example, by combining recent information about airtime utilization

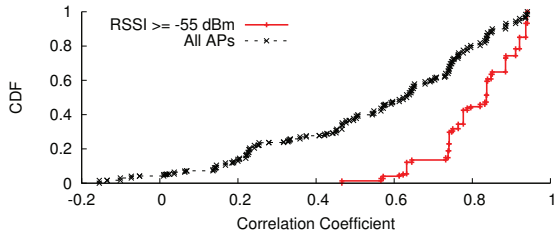

Fig. 4: CDF of the Pearson's correlation coefficient for time-series per-channel airtime utilization observed by neighboring AP pairs.

on different channels (e.g., within last 'x' minutes) from a COAP AP's neighbors, the controller can perform better channel assignments compared to only local AP observations. To motivate this application, we first analyze the feasibility of gathering airtime utilization statistics from neighboring APs to augment an AP's local view of the spectrum.

Can the controller leverage spatio-temporal locality of nearby APs for better channel selection?

In home WLANs, an AP can only observe the wireless activity for its current channel. This might cause the AP to miss opportunities to use a better channel during periods of channel congestion. With COAP, a single AP's view of the entire spectrum can be augmented using wireless activity observed by nearby COAP APs. We studied the feasibility of estimating the airtime utilization for other channels at an AP by using information from nearby APs (RSSI > -55 dBm) on those channels, as discussed next.

Using traces from [19], figure 4 shows the CDF of the Pearson's correlation coefficient values between the timeseries per-channel airtime utilization values (1-minute averages) observed by neighboring AP pairs. *It shows that more than 60% of nearby AP pairs whose mutual RSSIs were higher than -55 dBm exhibited a high correlation coefficient of more than 0.8*. Furthermore, almost all such AP pairs exhibited a correlation higher than 0.6. In contrast, only 15% of all AP pairs exhibited a correlation of more than 0.8 between airtime utilization values. These results show that it is possible to exploit the *spatio-temporal locality* of wireless activity between nearby APs to estimate airtime utilization on other channels. This can assist the COAP controller to perform better AP channel assignments by applying well known algorithms [23].

What are the performance improvements due to centralized channel assignments?

To study the benefits of using COAP framework for channel assignments, performed "airtime-aware" dynamic channel assignments to minimize airtime utilization/congestion (and improve throughput) experienced by our residential deployment of 12 COAP APs. Due to the sparsity of our deployment (12 APs in a large apartment), we used a secondary wireless card on our APs to collect airtime utilization information across all channels in a round robin fashion. In this manner, we emulated the use of nearby APs to collect airtime utilization statistics.

Over a span of 6 days, we employed 2 different channel configuration strategies on alternate days (1 day per configuration): static (and random) channel assignment, and dynamic

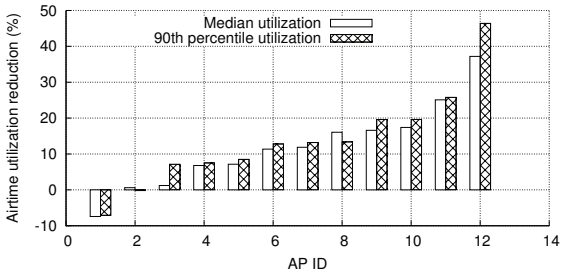

Fig. 5: Percentage reduction in median and 90th percentile airtime utilization in our deployment of 12 COAP APs using airtime-based channel configuration.

channel assignments using COAP to reduce airtime utilization at COAP APs. In the second case, we used airtime utilization statistics from the most recent 5 minute interval to estimate external wireless activity and select the best channel per AP [23]. These channel configuration updates were only applied during quiet periods to avoid service disruption for users.

Figure 5 shows the percentage reduction for median and 90th percentile airtime utilization values across the 12 COAP APs for the COAP-based dynamic channel configuration over the static channel assignment scheme, which is the case across majority of homes today. *It shows that the dynamic "airtime-aware" scheme performed better than a random channel assignment scheme for 10 out of the 12 APs. Four out of the 12 APs (APs 9 - 12) experienced 20% or more (upto 47%) reduction in airtime utilization.* This shows the potential gains achievable through a centralized approach to home AP management. Even non-participating APs in the vicinity of COAP APs can benefit from dynamic channel assignment of COAP APs due to decrease in overall channel contention. Further proliferation of WiFi-capable devices and high volume traffic (e.g., HD videos) will further increase the future utility of COAP based approaches to reduce channel contention.

V. AIRTIME MANAGEMENT

For the COAP framework, we developed the *SetAirtimeAccess(transmit_bitmap, slot_duration)* API which provides a lightweight and flexible primitive to the COAP controller to manage the airtime access patterns of COAP APs. We propose this API for solving two problems experienced by home APs (§II): (i) Channel congestion caused by nearby AP traffic, (ii) hidden terminal style interference resulting in packet losses. The intuition is to mitigate these scenarios by controlling the airtime access patterns of the interfering APs.

For example, the *impact of hidden terminal interference can be mitigated by preventing overlapping transmissions between the two interfering APs*. To detect hidden terminal interference, we borrow techniques from prior work [19] that correlates timeseries packet losses and activity of nearby links. Similarly, *channel congestion experienced by high priority flows (e.g., VOIP, HTTP based video flows) can be mitigated by reducing the airtime access of nearby APs with low priority flows* (e.g., bulk downloads). Lightweight and coarse-grained classification techniques (e.g., [13]) can be used to detect flow types and identify these sensitive flows.

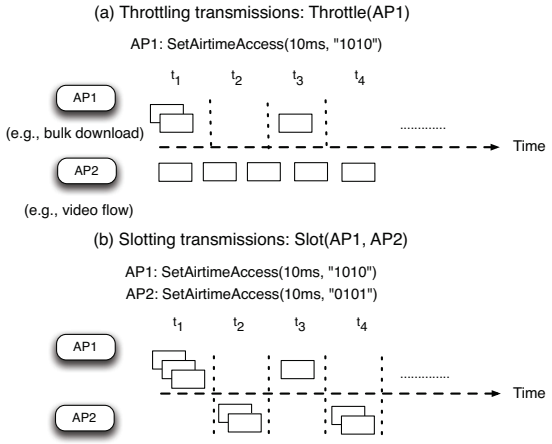

Fig. 6: Two mechanisms to manage airtime access of COAP APs: $Throttle(AP_x)$ (top) and $Slot(AP_x, AP_y)$ (bottom). $Throttle$ limits the airtime access of a single COAP AP while $Slot$ coordinates the airtime access of two interfering COAP APs.

A. API details

We use two mechanisms to manage the airtime access of COAP APs (Figure 6): $Throttle(AP_x)$ and $Slot(AP_x, AP_y)$. For this purpose, the controller divides time into small "slots" (10 - 20 ms) and can enable/disable transmissions of COAP APs on a *per-slot* basis to manage their airtime access. $Throttle(AP_x)$ is used to limit a COAP AP's transmission to certain slots to decrease its airtime utilization, and thus reduce channel contention at the neighboring APs. $Slot(AP_x, AP_y)$ is used by the COAP controller to configure two APs to transmit in *non-overlapping time slots*. This is useful in mitigating hidden terminal interference since overlapping transmissions cause packet losses at clients. We note that these mechanisms are only applied to data packets.

In contrast to enterprise-centric centralized approaches such as CENTAUR [28] that explicitly schedule flows on the data path to mitigate interference, it is not possible to do across home APs due to the lack of a centralized data-plane. But the centralized SDN framework still allows the COAP controller to coordinate transmissions between nearby home APs using the aforementioned API. This approach is motivated by RxIP [12], which uses a distributed mechanism for coordinating AP transmissions but can result in scalability issues in large multi-dwelling residential units.

API usage. To use this API, COAP APs are configured with the two parameters: *"slot duration"* and *"transmit bitmap"*. The *slot duration* (e.g., 10 ms) determines the time granularity to use for airtime access management (enable/disable transmissions). The per-AP *transmit bitmap* specifies the slots in which the AP is allowed to transmit. For example, a transmit bitmap of "1010" indicates that the AP is allowed to transmit in time slots 0 and 2 and disabled in slots 1 and 3 (Figure 6, top). This pattern represents a throttle value of 50% since the AP can only transmit data packets during 50% of the time.

For the $Slot(AP_x, AP_y)$ mechanism, the COAP framework uses a passive mechanism from WiFiNet [25] to synchronize

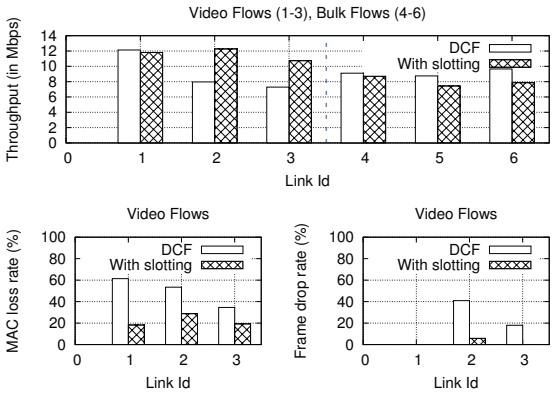

Fig. 7: TCP throughput (top), MAC layer loss rate (bottom-left) and frame drop rate (bottom-right) for 3 video links experiencing hidden terminal interference. Using non-overlapping time slots for interfering link pairs decreased MAC losses of video flows by more than 50% and frame drop rates to < 4%. Links 4 - 6 observed similar MAC losses in both cases (not shown).

the APs' clocks with < 0.5 ms error which is much smaller than the slot duration of 10 to 20 ms. This slot duration is large enough for frame aggregation while not impairing delay sensitive traffic, such as VOIP [6].

Implementation. We have instrumented the ath9k wireless drivers for our 802.11n based APs to support this API. For example, to throttle the airtime access of an AP, we disable the AP's transmit queue (using the AR_Q_TXD register) to block packet transmissions for the required period of time. Since this is a software-only modification, the underlying implementation of this feature is driver specific and transparent to COAP.

B. Testbed evaluation

To perform controlled experiments and measure ground truth accurately, we setup a local testbed consisting of 6 802.11n based COAP APs and 6 clients. We emulated two targeted problem scenarios and evaluate the benefits of using the airtime management APIs: (a) hidden terminal client-side interference and (b) channel congestion experienced by APs.

Amongst these 6 links, 3 links consisted of HTTP based video traffic and we emulated bulk data downloads on the remaining 3 links using iperf traffic. We setup a video server and used the open source Strobe Media Playback (SMP) media player [17], a popular browser based HTTP based video streaming client to generate TCP based HD video traffic. The SMP client was also used to log various video quality metrics (e.g., frame drops, stalls, buffer period etc.). We used a set of 3 metrics for evaluation: (i) the per-link TCP throughput, (ii) MAC layer packet loss rates to indicate WiFi link quality and (iii) the frame drop rate for video flows, indicating the percentage of total video frames dropped by the SMP player.

Hidden terminal scenario. We setup 3 hidden terminal scenarios by creating hidden terminal like conditions between the 3 video clients streaming a 10 Mbps HD video using Firefox and the 3 bulk download clients with 10 Mbps traffic. Figure 7 shows the performance metrics for the video (clients

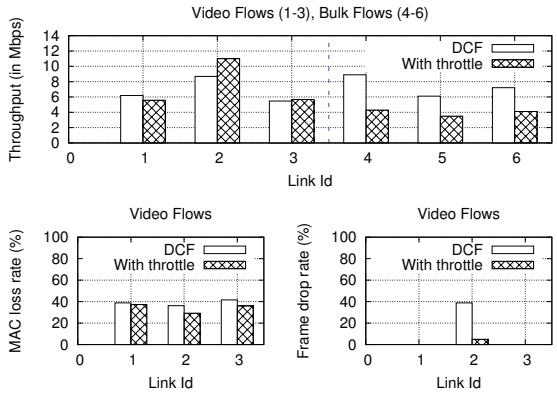

Fig. 8: TCP throughput (top), MAC layer loss rate (bottom-left) and frame drop rate (bottom-right) for 6 links (3 video, 3 bulk flows) experiencing congestion. Throttling the bulk flows to 50% airtime access improved the performance of video link 2.

1 - 3) and bulk flows (clients 4 - 6) with the usual 802.11 DCF mechanism and using the $Slot(AP_x, AP_y)$ mechanism to mitigate interference by configuring interfering links to transmit in non-overlapping time slots.

In the DCF scenario, all three video flows experienced high MAC layer losses (38% or higher) due to interference. Furthermore, video clients 2 and 3 experienced a high frame drop rate (20 - 40%) due to degraded throughput (< 8 Mbps). Using the $Slot(AP_x, AP_y)$ mechanism improved the throughput of all video links (> 11 Mbps) while the bulk flows still obtained similar throughput. *Furthermore, the MAC losses for the video links decreased by more than 50% indicating better WiFi link quality and more efficient airtime utilization while the frame drop rate was less than 4% across all video links, indicating good video quality.*

Mitigating channel congestion. We emulated a congested channel scenario by placing 2 medium bit-rate (5 Mbps) HTTP video links, 1 high bit rate (10 Mbps) HTTP video link and 3 bulk flows (10 Mbps) on the same WiFi channel. In the DCF scenario, the high bitrate video link (link 2) experienced high frame drop rates (40%) due to channel congestion while the medium bit rate video links (links 1, 3) experienced good performance (no frame drops) due to the lower throughput requirements. To reduce congestion, the 3 bulk flows (clients 4 - 6) were throttled to 50% airtime access using the $Throttle(AP_x)$ mechanism. *While the throughput of the bulk flows went down by 50%, the performance of the video link 2 improved (< 4% frame drops) due to the higher throughput achieved by the link (11 Mbps).* This shows how the COAP framework can improve the performance of more sensitive flows (e.g., HTTP based video) by leveraging cooperation between neighboring APs for airtime access. The framework collects information about client and traffic type (§VI-C), which can be used as hints to identify such flows.

VI. LEARNING FROM PRIOR CONTEXT TO PREDICT FUTURE ACTIVITY

The controller can also analyze logs about prior wireless activity collected from COAP APs to employ *learning techniques* and mine "context-related" information. We present two examples in this section: (i) Predicting future non-WiFi activity using microwave ovens as an example, (ii) using client device type and traffic information to predict traffic characteristics.

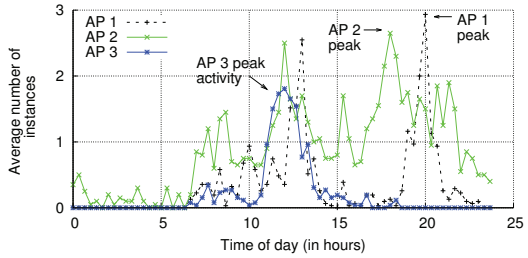

Fig. 9: "Activity vectors" for microwave oven activity observed by three different COAP APs (2 weeks).

A. Modeling and learning about non-WiFi activity

Prior work such as TIMO [24] use MIMO based techniques to mitigate the impact of non-WiFi interference. In this work, we investigate the feasibility of an orthogonal approach to avoid non-WiFi interference by predicting future activity. The intuition here is that certain non-WiFi interferers (e.g., microwave ovens) get used at specific periods of the day.

In the COAP framework, the controller records non-WiFi activity observed by COAP APs using commodity WiFi cards [22]. If a non-WiFi interferer is regularly active during specific time periods, the controller can perform pre-emptive configuration of the COAP AP that gets consistently impacted by the device. We now analyze the "predictability" of future non-WiFi activity using microwave ovens as an example.

Building "activity vectors". To represent a particular device type's activity (e.g., microwave oven) observed by an AP, we create a per-AP "activity vector" for each non-WiFi device. Each *activity vector* is represented as a vector (V) to represent the non-WiFi activity during the 24 hours of a day:

$$V :< c_1, c_2, ..., c_n > \forall 1 \leq i \leq n \ , c_i \geq 0.0$$

Each element (c_i) in the vector records the average number of daily instances of non-WiFi activity observed during a time "bin period" (e.g., 8pm - 8:20pm - using 20 minute *bin periods*). The activity vectors are constructed by aggregating information about the non-WiFi activity over a span of multiple days. Figure 9 shows the activity vectors and peak-activity periods for microwave oven activity at three COAP APs. Furthermore, there is large diversity in the temporal activity at these APs. Our goal is to analyze the fidelity of these activity vectors over a period of time.

B. Predicting non-WiFi activity.

We analyzed microwave oven activity traces collected from 30 different APs in [19] to build the aforementioned activity vectors over consecutive days using different "time spans" (1, 2, ... 30 days). This allowed us to capture their aggregate activity characteristics over different time periods. For a given time span of k days, using per-AP activity data (total of d days), we

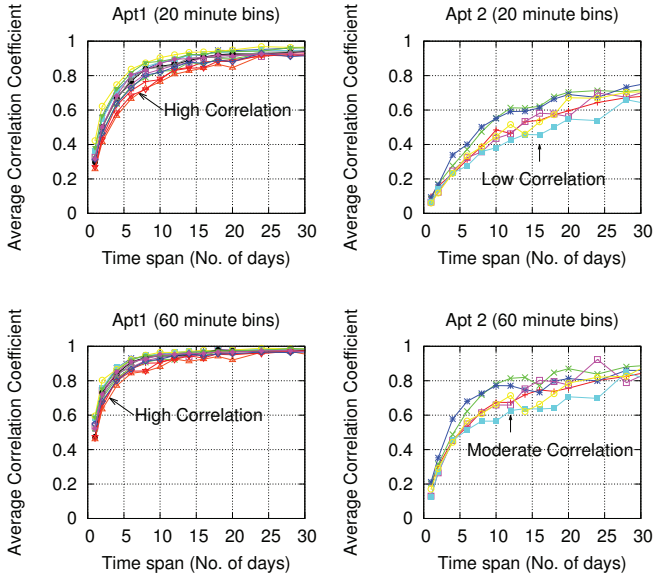

Fig. 10: Average correlation between consecutive microwave oven activity vectors for APs in apartment buildings Apt 1 and Apt 2 as a function of the "time span". The top graphs use "activity vectors" with 20 minute bin periods while the bottom graphs use 1 hour bin periods.

obtained a sequence of activity vectors $(V_1^k, ..., V_{d/k}^k)$. Using all time spans $(k = 1...30)$, we obtained 30 different sequences of activity vectors:

$$(V_1^1, V_2^1, ..., V_d^1), ..., (V_1^{30}, ..., V_{d/30}^{30})$$

We used these sequences of activity vectors to determine the predictability of future non-WiFi activity (V_{i+1}^k) based on the most recent record of non-WiFi activity (V_i^k). For each *time span,* k $(1 \leq k \leq 30)$, we computed the per-AP Pearson's correlation coefficient between consecutive activity vectors and averaged them:

$$\frac{1}{n-1} \sum_{i=0}^{(d/k)-1} corr(V_i^k, V_{i+1}^k)$$

A high correlation for a given time-span, e.g. k = 10 days, indicates high similarity for non-WiFi activity between the consecutive 10 day periods and indicates the fact that using information about non-WiFi activity from the previous 10 days can be used as an indicator for future activity.

Results. For the correlation analysis, we experimented using activity vectors with different *bin periods* to capture microwave oven activity at different time granularities. We discuss the results for two different bin periods - 20 minutes (figure 10, top) and 1 hour (figure 10, bottom). The results are divided between the APs located in two residential building – Apt 1 and Apt 2. In Apt 1, the APs were deployed within individual apartment units across 5 floors of the building. In Apt 2, the APs were deployed in common areas and shared by the different tenants within the building.

The results provide some interesting insights. With 20 minute bin periods (figure 10, top-left), the correlation between consecutive activity vectors for short time spans (1 - 2 days) across both buildings was less than 0.5. This indicates that short time spans were not able to capture the diversity in microwave oven activity. Using time spans greater than 5 days resulted in moderate (0.6) to high correlation (0.8) for microwave oven activity at all APs in Apt 1. Using a time span greater than 10 days did not provide additional gains. By using more coarse grained bin periods of 1 hour (figure 10, bottom-left), it was possible to capture this information in a shorter time span of 3 - 4 days. *This indicates that a span as low as 5 days could capture enough information to predict future microwave oven activity nearby APs in Apt 1.*

Interestingly in Apt 2, by using 20 minute bin periods (figure 10, top-right), the trends did not converge even after 20 days as indicated by the lower correlation (0.6) for the activity vectors across all APs. One of the major reasons for this behavior is the location of these APs in common areas of the building. Furthermore, there were microwave ovens present in the common areas of the building. This resulted in lower predictability of future activity due to the high diversity of microwave oven activity as well as poor detection accuracy for farther devices. Using more coarse grained bin periods of 1 hour (figure 10, bottom-right) resulted in better predictability – correlation coefficient of 0.6 to 0.8 for time spans > 10 days. This analysis shows how the COAP controller can incrementally learn about such spatio-temporal properties about non-WiFi activity at different time granularity (based on the deployment). Depending on the nature [22] and extent of interference impact, *this information along with the activity vectors can be used by* COAP *controller for pre-emptive configuration of APs (e.g., WiFi channel, transmit power) to avoid non-WiFi interference.*

C. Learning client and traffic context to predict future activity

Since home APs act as a gateway to all WiFi-based devices, they can capture useful coarse-grained *contextual information* about the wireless activity. They include client device type information (e.g., iPhone, Google TV, laptop) as well as the traffic source ID (e.g., Netflix, Pandora). In the COAP framework, the controller can use such contextual information to predict future traffic characteristics and perform traffic aware configuration of APs. Following is an example scenario – *"AP 4 has started a high-volume HD video flow on his wireless TV, which is going to last for 30 minutes with 90% probability. Therefore, move adjacent APs to other WiFi channels to reduce channel contention during this traffic."*

Why device type and traffic context can help?

For the following analysis, we use the traffic traces collected from the COAP deployment to motivate the benefits of using device and traffic context to predict traffic characteristics. The traffic sources were determined by finding the TLD (Top Level Domains) names corresponding to the traffic flows. For COAP's deployment, the TLD can be hashed and anonymized to preserve user privacy. The client device types were determined

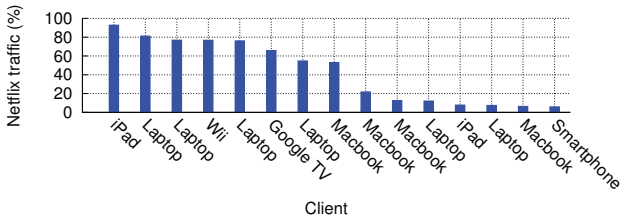

Fig. 11: Total proportion of Netflix traffic across top clients.

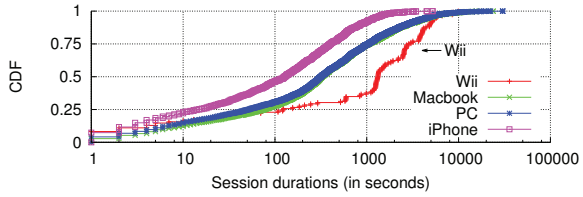

Fig. 12: CDF of session lengths by client device type for 4 different client devices.

by only using the prefix of the MAC address (first 24 bits) determine the Organizationally Unique Identifier (OUI) as well as leveraging hints from the host name (e.g, many devices such as iPhones use the device type in their host name). We used the traffic traces to extract "active periods" for flows (> 40 packets/second) and obtain the following traffic properties during the active periods:

- *Burst properties.* A *burst* is a time-period during which the gap between consecutive active periods is less than 10 seconds. For bursts, we compute the duration, downloaded bytes, the average and maximum download speed.
- *Session properties.* A *session* is a sequence of consecutive traffic bursts with a gap of less than 5 minutes. These periods indicate active usage of the traffic source. For each session, we compute gaps between consecutive bursts, duration, bytes downloaded and download speeds.

Client and traffic context information can be helpful in predicting traffic properties due to the following reasons:

Bias in traffic usage profile by device type. In our traces, Netflix was the highest traffic source by volume. Figure 11 shows the proportion of Netflix traffic across the top Netflix client devices. Analyzing the distribution on a per-device basis reveals a good correlation between device and traffic usage. Such bias can be exploited to predict traffic characteristics.

Impact of device usage characteristics. The device type can also impact user behavior (e.g., the time that they spend on a device). For example, a user may use his smartphone for only short periods of time but use a gaming console for longer periods of time. Figure 12 exhibits higher session duration for the Wii devices (median of 20 minutes) compared to other devices: median duration of 5 and 2 minutes for laptops (Macbook, PCs) and iPhones respectively.

Platform specific traffic behavior. The type of the platform (e.g., client device, OS) can impact traffic characteristics. For

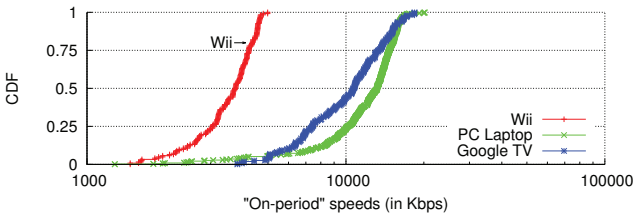

Fig. 13: CDF of maximum download speeds for Netflix traffic during burst periods on different device types.

example, video streaming services have been observed to use different strategies [21] based on the browser, device type etc. In our analysis, we observed a similar behavior for some popular traffic sources (e.g., Netflix). Figure 13 shows that while the maximum per-burst download speeds on Wii devices was only around 5 Mbps, it went upto 16 Mbps on PCs and Google TV. Since we do not have additional information about the underlying video traffic, we believe that the lower speeds for Wii devices might be due to the lower bit-rates used by the Netflix application or device limitations.

D. Predicting traffic characteristics

To quantify the benefits of using *device and traffic context information* for predicting the aforementioned traffic properties (e.g., session duration, downloaded bytes). we used the machine learning tool, Weka [16] and traffic traces collected by COAP APs (30 day period).

We represent the "context" as a collection of the following traffic and device related features which can be obtained at the start of a new traffic flow – *AP ID, client device id, traffic source id, time of day, day of week*. These input features were used to predict the burst and session related properties discussed earlier (e.g., session duration and bytes downloaded per session). We compared the performance of these predictions to the ones when only *AP ID, time of day and day of week* were used as input, i.e., excluding any device and traffic context. We used 10-fold cross validation on the input data to train and evaluate the prediction models.

Prediction Performance. We experimented with different algorithms within Weka and found that the REPTree algorithm [20] performed the best. It uses information gain to build a tree based model to generalize the properties of the underlying data.

Figure 14 (top) and figure 14 (bottom) show the CDF of per-AP and per-client device type (e.g., Wii, iPhone, PC) relative prediction errors (80th percentile values) respectively, for the session lengths and bytes downloaded per session. *It shows that using client device and traffic context information resulted in 2× or lower prediction errors compared to no-context scenario.* Also, these 80th percentile relative prediction errors are below 100% for the context-aware scenario indicating that it is possible to predict these traffic properties within a factor of 2 to 3 times in majority of cases. We observed similar results for predicting other traffic properties (e.g., download speeds). Thus, client device type and traffic context can provide valuable information to the COAP controller for predicting future wireless activity and perform potential adaptations (§IV, §V).

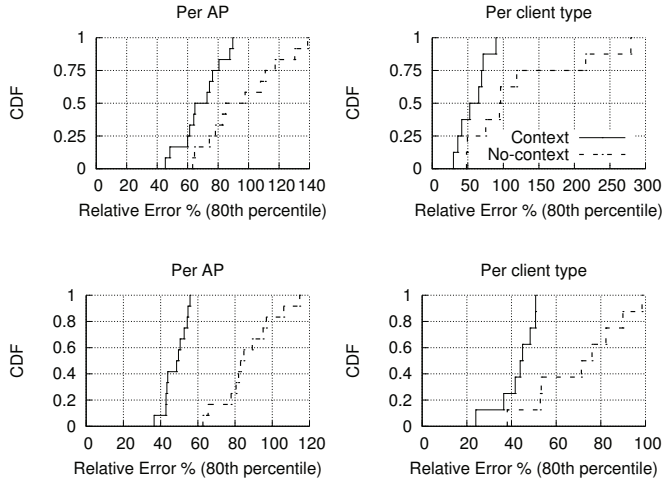

Fig. 14: CDF of the per-AP and per-device type 80th percentile prediction error values for session lengths (top) and bytes downloaded per session (bottom) with and without using client device + traffic context.

VII. RELATED WORK

SDN style management of APs. Meraki [3] pioneered the concept of cloud-based management of enterprise APs based on an proprietary solution that configures their homogeneous APs remotely though the cloud. Dyson [14] proposes a centralized framework to manage APs and clients in enterprise WLANs using a set of APIs at both APs and clients. We propose an open API by augmenting existing SDN frameworks to enable cooperation between heterogeneous neighboring APs in residential settings. OpenFlow [15] is an open and popular SDN framework designed to manage routing policies and other networking-related configurations. Prior work on using OpenFlow for 802.11 networks [11], [26] focused on building an experimental platform and managing mobility of devices. Our goal is to use the SDN approach to actively manage wireless parameters of 802.11 networks and build management frameworks, especially for residential settings. It is also possible to use other open AP management standards (e.g., CAPWAP [8]) to implement COAP related APIs.

Using multi-AP support in non-enterprise 802.11 deployments. Prior research has proposed mechanisms to leverage support from multiple APs to increase throughput using client-based backhaul aggregation [10], use AP virtualization to improve video performance through bandwidth sharing [27] and mitigate hidden terminals using distributed algorithms [12]. With COAP, we motivate the use of a cloud-based framework to enable cooperation between nearby home APs and mitigate wireless problems in residential WiFi deployments. This is done by aggregating airtime utilization information from nearby APs for channel configuration, coordinating airtime access across APs to alleviate both channel congestion and hidden terminal scenarios and learning from prior activity to predict traffic usage patterns and future non-WiFi interference.

VIII. CONCLUSION

We presented the COAP framework, which uses a vendor-neutral open API and a cloud based controller to enable cooperation between heterogeneous and collocates home WiFi APs in dense residential deployments We describe multiple applications to motivate the benefits of using this centralized framework – better channel assignments using airtime utilization information from co located home APs and managing airtime access of neighboring APs to reduce channel contention for important flows (e.g., HTTP based video) as well as mitigate hidden terminal interference. We also show that learning about previous wireless activity can enable the controller to better predict future non-WiFi interference and wireless traffic properties. This can allow the controller to pro-actively configure home APs for interference mitigation as well as perform traffic aware adaptations at APs.

REFERENCES

[1] Click modular router. http://www.read.cs.ucla.edu/click/click.
[2] COAP Project. http://research.cs.wisc.edu/wings/projects/coap/.
[3] Meraki enterprise cloud management. www.meraki.com/products/wireless/enterprise-cloud-management.
[4] Openwrt. https://openwrt.org/.
[5] Project floodlight. http://www.projectfloodlight.org/floodlight/.
[6] Skype requirements. http://download.skype.com/share/business/guides/skype-connect-requirements-guide.pdf.
[7] D-Link. Cloud router. http://www.dlink-cloud.com/solutions.aspx.
[8] IETF. Capwap protocol specification. http://tools.ietf.org/search/rfc5415.
[9] IETF. netconf. http://datatracker.ietf.org/wg/netconf/charter/.
[10] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi. FatVAP: Aggregating AP Backhaul Capacity to Maximize Throughput. NSDI'08.
[11] Kok-Kiong Yap et al. The Stanford OpenRoads Deployment. In *Proceedings of the annual conference on WinTech, 2009*.
[12] J. Manweiler, P. Franklin, and R. Choudhury. RxIP: Monitoring the health of home wireless networks. In *INFOCOM 2012*.
[13] A. W. Moore and K. Papagiannaki. Toward the accurate identification of network applications. PAM'05.
[14] R. Murty, J. Padhye, A. Wolman, and M. Welsh. Dyson: An Architecture for Extensible Wireless LANs. USENIX ATC'10.
[15] Nick McKeown et al. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 2008.
[16] T. U. of Waikato. Weka. http://www.cs.waikato.ac.nz/ml/weka/.
[17] OSMF. Strobe media playback. http://osmf.org/strobe_mediaplayback.html.
[18] A. Patro and S. Banerjee. Coap: A software-defined approach for home wlan management through an open api. MobiArch '14.
[19] A. Patro, S. Govindan, and S. Banerjee. Observing Home Wireless Experience Through WiFi APs. MobiCom '13.
[20] Pentaho. Reptree. http://wiki.pentaho.com/display/DATAMINING/REPTree.
[21] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous. Network Characteristics of Video Streaming Traffic. CoNEXT '11.
[22] S. Rayanchu, A. Patro, and S. Banerjee. Airshark: Detecting non-WiFi RF devices using commodity WiFi hardware. IMC '11.
[23] E. Rozner, Y. Mehta, A. Akella, and L. Qiu. Traffic-Aware Channel Assignment in Enterprise Wireless LANs. In *ICNP 2007*.
[24] S. Gollakota et al. Clearing the RF smog: making 802.11n robust to cross-technology interference. SIGCOMM '11.
[25] Shravan Rayanchu. Catching Whales and Minnows Using WiFiNet: Deconstructing non-WiFi Interference Using WiFi Hardware. NSDI'12.
[26] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao. Towards programmable enterprise WLANs with Odin. HotSDN '12.
[27] Vijay Sivaraman et al. Virtualizing the Access Network via Open APIs. CoNEXT '13.
[28] Vivek Shrivastava et al. CENTAUR: Realizing the Full Potential of Centralized Wlans Through a Hybrid Data Path. MobiCom '09.
[29] Yiannis Yiakoumis et al. In *Proceedings of the 2nd ACM SIGCOMM workshop on Home networks*, HomeNets '11.