BeHop: A Testbed for Dense WiFi Networks

Yiannis Yiakoumis Manu Bansal Adam Covington
Johan van Reijendam Sachin Katti Nick McKeown
Stanford University
{yiannisy,manub,gcoving,jvanreij,skatti,nickm}@stanford.edu

ABSTRACT

We present BeHop, a wireless testbed for dense WiFi networks often seen in residential and enterprise settings. BeHop aims to provide insights on the operation of dense deployments, and evaluate how different WiFi management strategies affect user experience and network behavior. It has sufficient flexibility to let us try different management techniques and setups (e.g. residential or enterprise, client or infrastructure-driven operation). It is deployed at a university dorm, where it acts as the main network for a diverse set of users and devices, exposing practical insights and implications on the operation of the network. In this paper we discuss the design and implementation of BeHop, and share our early experience over a five-month period.

Categories and Subject Descriptors

C.2.0 [**Computer Systems Organization**]: Computer-Communication Networks—*General*

General Terms

Design, Experimentation, Management, Measurement, Performance, Reliability

Keywords

Production Testbed; Dense WiFi; Residential WiFi; Enterprise WiFi; Flexibility

1. INTRODUCTION

Dense WiFi networks — where multiple APs and clients coexist — are commonplace and present unique challenges. In the US, 24% of households are in multiunit buildings [6]. World-wide, a large and growing fraction of the population lives in high-density housing. Residential networks are chaotically managed and deployed. A vast majority of users installs their WiFi APs with factory default settings, and lacks the sophistication — or desire — to change the configuration. Multiple networks run independently and share the same airtime resources, which often leads to poor channel selection, poor coverage, and unsatisfying user experience. While better managed and provisioned, enterprise networks face similar challenges. They deploy, often incrementally, a large number of APs while trying to account for coverage, performance, mobility, and interference with neighboring networks.

WiNTECH'14, September 7, 2014, Maui, Hawaii, USA.

ACM 978-1-4503-3072-5/14/09 ...$15.00.
http://dx.doi.org/10.1145/2643230.2643233.

Despite growing interest in improving dense networks both in enterprise and academia, we have little practical experience on how they really work. While we have a rough sense of useful approaches for managing a network (such as channel, power and association control), we do not have good intuition on how these will work in practice, and their implications. Research testbeds are often built around a specific usecase, and they are limited in terms of static traffic and clients, hiding the complexity of a real network (e.g. the impact of client diversity, mobility, and real traffic), which can lead to impractical assumptions or overlooked factors. On the other hand, enterprise WiFi networks apply a combination of WiFi management schemes to improve network performance, but their platforms are proprietary with little visibility into their operation.

We are principally interested in studying and evaluating the pros and cons of new ways of controlling WiFi networks. For example, we plan to evaluate the benefits of centrally controlling a large WiFi network, controlling power, channel allocation and association from a single vantage point. To do this, we built a general-purpose WiFi testbed in our university. We set out with two main requirements for our testbed, which we suspect are common goals of many WiFi testbeds:

We want to control how the wireless channel is used: Our testbed should let us *centrally* control the power and channel of each AP. It should also let us control which AP serves a client on a per-client basis. This lets us try out, and compare, a number of different control strategies: channel and power allocation algorithms, load-balancing algorithms, client-driven association versus infrastructure-selected association, and 5GHz band-steering.

We want the testbed to be real-world deployable: Our testbed should carry the traffic of real users, running any application they choose on any of their WiFi-connected devices (laptops, phones, tablets, DVRs, etc). This will let us evaluate our strategies and reveal their implications in a real setup. Furthermore, we want to eliminate (or at least minimize) the overhead for users to join the testbed. When

our testbed network fails or is taken down for servicing and upgrades, our users should fail-over to a regular production WiFi network without even noticing. When our testbed is back on line, they should automatically use it again, without any intervention from us or them.

In this paper we present BeHop, an operational testbed which aims to provide insights on dense WiFi networks, and share our experiences on how to design and build such a testbed. Our specific contributions in this paper are the following:

- We design and prototype an SDN WiFi framework which can be used to realize a wide set of managment techniques for channel, power and association control in different environments. We start by prototyping the necessary extensions for channel, power and association control. We use a *virtual-AP abstraction* to decouple our WiFi logic from the physical infrastructure and control how to expose our infrastructure to users (sparse or dense, home or enterprise network). We implement *per-client* WiFi control to simplify management, ease troubleshooting, and apply and evaluate policies on a fine granularity. §2 discusses our overall architecture and control primitives that provide BeHop the desired flexibility and control.

- We evaluate a novel deployment strategy for WiFi testbeds which practically improves realism and actual usage. BeHop runs in parallel with and is indistinguishable from the production network for our users. It serves the same SSID as the production network. As a result, users cannot really say whether they use BeHop or the legacy network. To our surprise, this has important implications on the overall behavior of our testbed. Users do not have to change any setting to use the network (such as WiFi network preferences). When our network fails or during an upgrade, they automatically fall-back to the production network without even noticing. When it comes back, users seamlessly use it again. We describe benefits, drawbacks and overhead of integration with the production network in §3.

We run BeHop as the main network for 39 users and more than 90 devices since January 2014. Using BeHop, we are able to gain useful insights on the operation of dense WiFi networks and systematically evaluate benefits and implications of WiFi management techniques. We evaluate BeHop through one specific usecase serving as an example — use of the 5GHz band — in §4, and reflect on our experience in §5. We compare with related work in §6.

2. ARCHITECTURE

We design BeHop to enable control and flexibility in a real-world environment. We leverage SDN to easily define our network-wide policies at a single point in the network. Our testbed is built using commodity APs and a centralized controller. We now discuss our extensions to expose WiFi-specific control, and describe the main components of our BeHop architecture: i) BeHop APs as our wireless dataplane, ii) a collector that gathers necessary data for monitoring and evaluation, and iii) a centralized controller where we implement our WiFi management logic (Fig. 1(a)).

2.1 BeHop AP

Our dataplane consists of commodity APs. Besides packet forwarding, BeHop APs implement low-level WiFi tasks with strict timing constraints (namely rate adaptation, beacon, and acknowledgement generation). BeHop APs act as OpenFlow switches and extend SDN to expose primitives for channel, power and association control. We implement the following primitives:

Virtual-AP (VAP): A VAP is a logical access point that appears like a physical AP to a client. It is uniquely identified by a BSSID [1], and implements two main tasks: beacon generation and acknowledgement generation. In BeHop, clients only see VAPs, not the true physical APs. VAPs decouple the operation of an AP from the underlying infrastructure. BeHop APs provide an API to add (or remove) a VAP to a specific radio interface of a physical AP.

Client Table: Each AP maintains a client table. An entry in this table contains necessary state so that the AP can serve the client (such as rate-control information, driver capabilities, and client to VAP mapping). We can remotely install appropriate state for a wireless station by adding an entry in the client table.

Forward WiFi managment traffic: BeHop APs forward WiFi management traffic to the controller. The controller is responsible for processing and responding to Probe, Authentication, and Association Requests.

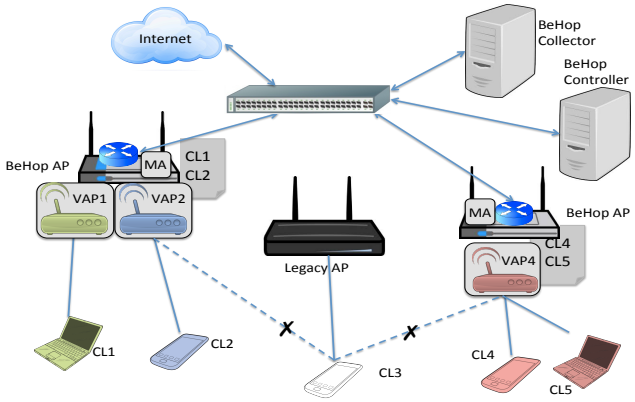
WiFi Configuration: BeHop APs expose a configuration API for channel and power allocation.

Each AP additionally has a *monitoring agent* (MA) which forwards WiFi statistics and raw WiFi packets to a collector (along with radio metadata for SNR, WiFi rate, and retries). We can configure what statistics to collect (e.g. channel utilization, retransmision ratio) and which packets to collect (we currently use Probe Requests and WiFi-acknowledgements to monitor per-station SNR).

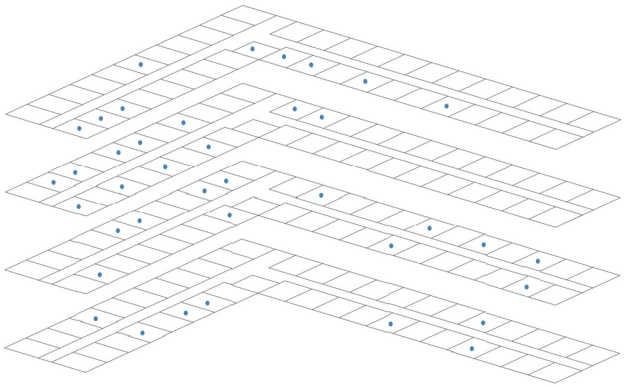
2.2 Collector

We use a *collector* as the information plane of our testbed. The role of the collector is to gather raw data from different sources to construct higher-level statistics which we later use to either inform our management algorithm or measure and evaluate the performance of the network. Our collector has three sources of data: i) WiFi statistics reports from APs, ii) raw WiFi packets with radio metadata (SNR, retries, WiFi rate), and iii) a mirror feed from the traffic on the wired backend of our testbed. The collector maintains an SNR profile for each wireless client, a database with clients and their properties (e.g. support for 5GHz, capabilities, associated user), and exposes an RPC interface which we can query from our controller. It also collects anonymized packet-traces, which we use to evaluate BeHop's performance and compare different management strategies. The collector can be co-located or separated from the controller. In our deployment we keep them separate to maintain the flexibility to collect and process large amounts of data without sacrificing the responsiveness and stability of our controller.

[1] A BSSID is a 48-bit address.



(a) Overall BeHop architecture.



(b) Deployed WiFi APs. 39 out of 150 residents participate on BeHop so far.

Figure 1: Each BeHop participant installs an AP in his/her studio apartment. APs are centrally managed from a controller. We collect traffic from BeHop and the legacy network to compare performance and get full visibility on clients' connectivity.

2.3 Controller

The *controller* orchestrates our testbed to realize different WiFi management strategies. We can implement channel and power allocation mechanisms using the relevant APIs along with appropriate information from our collector (such as channel utilization). We can "disable" a physical AP by removing all VAPs from it; we can add more than one VAP per physical AP to emulate higher density. We can apply association control by mapping VAPs to physical APs, and then clients to VAPs. For instance, to realize the default WiFi operation, we map one VAP to every radio interface, respond to all probe requests we receive, and install station-related state to the AP a client sends an association request to. In order to realize band-steering, we ignore requests in the 2.4GHz band when 5GHz is available. To mimic the scenario of a residential setting where each user may access only her own AP, we reply only to probe-requests originating from there. To get full association control, we hide all the complexity from the clients and expose a VAP with the desired properties (channel, physical AP, etc.) for each client; we can then seamlessly move a client from one AP to another to readjust load, or better support mobility.

Per-client control: Our controller allows us to define the desired WiFi management strategy on a *per-client basis*. This is helpful both for experimentation and the practicalities of a real-world deployment (run targeted tests, ease debugging and troubleshooting). For instance, we can isolate a subset of devices and test a specific strategy on them (e.g. all mobile phones); we can implement features to map the needs of actual users ("exclude my tablet from the experiment"), or we can try a new feature to a test device while continuing to serve other clients using the default mechanism.

2.4 Implementation

We now discuss some implementation details for our design that might be useful for others trying to replicate parts of the functionality.

Our datapath consists of commodity dual radio WiFi APs running OpenWRT. [2] BeHop APs run Open vSwitch (OVS) along with our SDN WiFi extensions that implement the control API. APs connect to the backbone campus network through a 48-port, OpenFlow enabled switch. We build our control plane using the POX OpenFlow controller [8].

To forward WiFi management traffic, we add a monitor interface for each radio and forward packets of interest to a virtual ethernet pair which is controlled by OpenvSwitch. We then use OpenFlow's *packet-in* mechanism to forward packets to our controller. As we want to control client association and scanning from our controller, we disable the "hostapd" daemon on the APs, and implement the relevant functionality in our controller. Such an implementation required significant engineering and debugging effort, as covering all possible WiFi capabilities and client diversity is non-trivial.

To activate multiple VAPs on a single AP, we exploit Atheros' BSSID mask. Atheros hardware generates WiFi acknowledgements for packets destined to a target that matches its own MAC address combined with a 48-bit BSSID mask. To be able to uniquely acknowledge packets destined to a specific BSSID, we structure our BSSID accordingly: the first 12 bits identify the channel, then we use the remaining 36 bits as a bitmap (every BSSID has one-bit 1, the rest 0). We can then combine up to 36 Virtual-APs on a single channel without any ACK conflicts, leading to a total of 288 Virtual-APs across 8 channels.

We extend OVS' JSON-based configuration mechanism (OVSDB) to configure WiFi settings (channel and power), add and remove Virtual-APs and stations. We implement a local agent at the AP that listens to changes in the configuration DB and appropriately enforces them (through OpenWRT's *uci* and *netfd* tools), and also implement OVSDB functionality in our controller.

[2]We use NetGear WNDRMACv2 running OpenWRT Attitude Adjustment.

3. DEPLOYMENT

Deploying BeHop in a real-world setting is essential to study and evaluate WiFi management strategies and their implications under the conditions found in a real network (client diversity, mobility, interference with neighboring networks). We want our testbed to carry real traffic of users from any WiFi-connected device; at the same time, we want to maintain the flexibility to apply frequent changes, upgrade, and occasionally break the network, ideally without any impact on users' experience. To achieve this goal, we *integrate* BeHop with the production network serving the same SSID, making it indistinguishable for users and client devices.

We deploy BeHop in a 4-story dorm at Stanford University with 150 studio apartments (Fig. 1). It currently serves real traffic from 39 users and over 90 client devices (laptops, phones, tablets). Users willing to participate install an AP in their room. The building is already covered by a campus-wide network using the Cisco LWAPP technology (25 dual-band APs installed in the building). BeHop has a much denser deployment, with 50% more APs deployed. For development purposes, we also run a smaller, 3 AP network at the Computer Science Department at Stanford.

Most importantly, BeHop runs in parallel with and is indistinguishable from the (existing) production network for our users. This is in contrast with most testbeds which run separately from the production network, and has deep implications on the operation and properties of our testbed. We highlight the most important properties of our deployment. Even if some implementation techniques are specific to BeHop, we believe that these properties can be useful for many testbeds.

Easy for users to sign-up: Users have to sign-up to participate to BeHop (to install an AP to their room, comply with our university's IRB protocol, and address Stanford's IT department concerns for potential impact on students' network service). Students sign-up through a one-minute process on BeHop's website [4], and schedule a timeslot where we go and plug-in the AP in their room. They are granted free Netflix service for the duration of the experiment as an extra incentive. We use Stanford's database to obtain a list of each user's registered devices (MAC address). Sign-up is an ongoing process and we grow our deployment incrementally.

Same SSID with production network: BeHop is seamless for users to use and is integrated with the production network. It serves the same SSID by default, and is part of the same L2/L3 network. Users do not have to manually choose our network, or change their WiFi preferences and SSID priority-list on their devices (a task which only becomes harder with many devices for every user), and they do not have to switch explicity when their device is already authenticated to the legacy network. [3] Device IP addresses stay the same and flow-continuity is preserved when switching from BeHop to LWAPP and vice-verca.

At the same time, to provide a seamless experience to users we had to trade-off control and visibility. A client device can freely switch between BeHop and LWAPP; we have little control over it. In order to serve participating devices through our APs, we compete to acquire them by responding to probe requests before the LWAPP network can. While this works well in general, it is not a perfect mechanism. Ultimately, the decision of choosing a BSSID rests with the client. Clients use diverse set of heuristics to pick the best-perceived AP. A few participating clients do end up on the LWAPP network part or most of the time. The control-visibility tradeoff became obvious in the first few days of our deployment where a bug in our code forced clients to switch to the LWAPP network after a few minutes. This lack of precise control over which AP serves a client also makes it harder to identify whether a problem is due to BeHop or the production network. To troubleshoot such issues, we had to gain visibility into the production network. We wrote a decoder for Cisco's CAPWAP protocol. [4] Combining logs from BeHop and Cisco, we have the full picture of how wireless clients behave, a feature useful both for debugging and performance monitoring.

Another issue that comes with integration is that some features may not be consistent across networks. For example, implementing a firewall on BeHop to isolate the devices of a single user from the rest of the network will be valid only as long as she is using our network.

Graceful failover mechanism: Integrating BeHop with the production network provides a graceful failover mechanism — a great feature for a testbed bound to break. When our testbed fails or is brought down for servicing and upgrade, users automatically fall back to the production network. And when we bring it back online, they rapidly switch back to BeHop. This gave us confidence to apply frequent changes to our infrastructure without affecting users. Even after a week-long downtime due to a hardware failure which could have otherwise caused users to stop using the testbed, users rapidly switched back to BeHop as it required no intervention by them or us.

Whitelist and blacklist nodes: We often have to whitelist or blacklist indivual nodes, and our controller provides an interface to do so on a per-node basis. To keep BeHop transparent to non-participants, we maintain a whitelist of participating nodes and reply only to them. Occasionally we have to blacklist a specific client, either because users experience problems or because we want to run an experiment with a subset of clients.

Opt-out mechanism: As BeHop looks similar to the production network, users cannot explicitly force connection to one or the other. This can be problematic; for instance, a bug in our code was causing a user's laptop to deauthenticate every few minutes but the user could not explicitly choose to join the production network. To deal with such scenarios, we have created two complaint mechanisms. Users can report problems and provide feedback through a web-based form. In addition, each AP has a button for opt-out - when a user presses this button, her devices automatically get blacklisted and fall-back to the LWAPP network. We manually add them back when we resolve the problem.

[3]Our university WiFi network supports campus-wide mobility, and often users are already authenticated when they enter the building.

[4]CAPWAP is the protocol used between Cisco's APs and centralized controller.

Keeping a WiFi testbed active is not trivial. Our previous experience [14] has taught us that recruiting users and getting them back after a failure is hard. Even though we did not realize it at first, integrating BeHop with the production network had a strong impact on the design and required features, and highlighted a trade-off between user convenience and long-term usage against full-control and implementation complexity. Overall, we believe seamless integration with production network is a very useful deployment strategy, especially when real usage is an important aspect of the testbed, even if it comes at the cost of relenting some control.

4. EVALUATION

BeHop acts as the main network for participating students, serving a total of ~30GB per day. Both the APs and the controller can handle the load of our deployment so far with low CPU usage. Forwarding of WiFi management traffic to the controller generates a peak-load of 630 packets-per-second. [5] Our APs can deliver up-to 180Mbps of TCP goodput, using a 40MHz channel on the 5GHz band. As we discussed in Sec. 2, we can scale up to 36 Virtual-APs per channel to ensure that there is no conflict on generated acknowledgements, and we can generate 36 beacons on a single AP using 100 milli-second interval.

We now evaluate BeHop in practice by experimenting with a specific usecase: how to use the 5GHz band to improve performance on a dense WiFi network.

4.1 Usecase: Studying 5GHz Band Steering

Better use of 5GHz is a common strategy for WiFi networks to leverage less interference and wider channels available on this band. Doing this is not straight-forward — many clients do not support the use of 5GHz band and the 5GHz band penetrates poorer through walls than the 2GHz band, making it hard to provide good coverage. As a result, we operate WiFi networks on a "dual-band" mode where each AP is configured to serve in both bands. Due to WiFi's distributed nature, clients decide how to connect to the network. They do not always make the best decisions. They might use the 2GHz band even when a less-loaded 5GHz connection is available, or stick to a 5GHz link that provides poor connectivity. Vendors and operators try to facilitate use of 5GHz through WiFi management features often referred to as band-steering.

We use BeHop to evaluate how the use of 5GHz affects the performance of the network, and what its implications are on network design. Our focus is not on improving performance per-se, but rather to highlight how we can use BeHop to gain useful insights into a real use-case.

For our experiment, we set all APs to the same channel. This is a reasonable approximation of the distribution of same-channel nodes in the setting where each user had her own AP (we have APs in ~25% of apartments and WiFi has 3 disjoint 20MHz channels in the 2GHz band). We then operate BeHop in three different modes:

BeHop-2GHz: All APs are configured for 2.4GHz-only operation. This acts as our main baseline.

BeHop-Dual-Band: APs are set for dual-band operation. Clients decide which band they will connect to.

BeHop-Bandsteering: APs are set for dual-band. We force capable clients to connect to the 5GHz band by responding to 5GHz probe requests only.

Setting up the desired mode of operation requires minimum effort, thus validating BeHop's flexibility and ease of programming. All we have to do is overwrite the handling of "probe request" packets in our controller, and apply this to the desired clients (e.g. the ones who support 5GHz). We did not have to make any changes in the APs. Going from 2.4GHz to "Band-Steering" required < 20 lines of code to program the controller.

As a reference, we also compare our results with the performance of the legacy network in an identical building across the street, supporting dual-radio and band-steering. Each experiment runs for a full day (March 18, March 22, and March 24, 2014 respectively). For our measurements, we collect all packet headers after anonymization.

We use *Packet-Delivery-Time* (PDT) as a metric to characterize the performance of the network. We calculate PDT as follows: using the wired switch behind the APs as our vantage point, we calculate the time between a TCP packet coming on the downlink, and its respective TCP acknowledgement from the client. PDT captures the actual traffic from users and accounts for everything that might degrade performance in the WiFi infrastructure: buffer build-up on the AP due to a link/channel slowdown, wireless retransmissions, channel congestion that may lead the client or AP to backoff, etc.

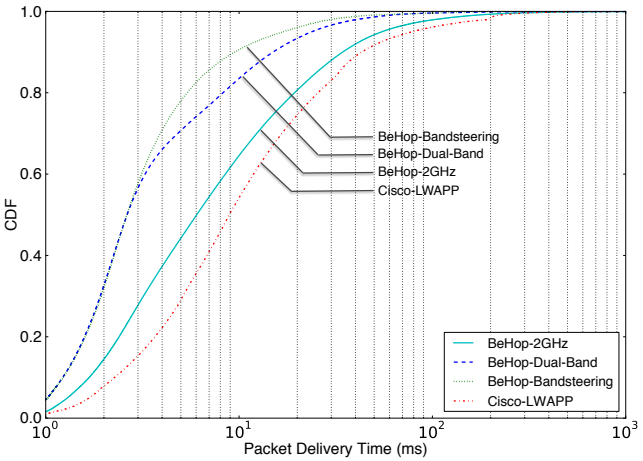


Figure 2: PDT for different strategies. Use of 5GHz results to faster delivery of packets.

Fig. 2 summarizes PDT for the different scenarios. We briefly discuss our main insights:

2.4GHz vs. 5GHz: We see that the use of 5GHz improves performance, reducing median PDT from 6.2msecs (BeHop-2GHz) to 2.3msecs (BeHop-BandSteering). This is not surprising. The 5GHz band has less interference and congestion, as signal does not propagate well through walls. Lack of interference leads to more efficient use of the channel (smaller backoff, less retransmissions, higher rates), and is also enhanced by the wider 40MHz channels. To get a better understanding of why 5GHz performs better, we zoom-in

[5] 90th percentile is less than 200 packets-per-second and WiFi management traffic comes from all clients in the building, not only BeHop users.

and look at packet-level events over a 2 second period, while streaming a Netflix video over 2.4GHz and 5GHz respectively. Both streams reach the maximum data rate (3Mbps), but the impact on the channel is dramatically different between the two bands. Fig. 3 shows a timeline of what happens in the air. Each rectangle denotes the transmission of a packet, its width being the airtime consumed to send the packet. Use of higher rates (up-to 300Mbps for data, 6Mbps for management packets) makes the 5GHz band much more efficient. Even though the maximum available rate on the 2GHz band is 144Mbps, most data packets are sent at lower rates (78Mbps, 52Mbps), probably due to bad channel conditions resulting from load (e.g. collisions).

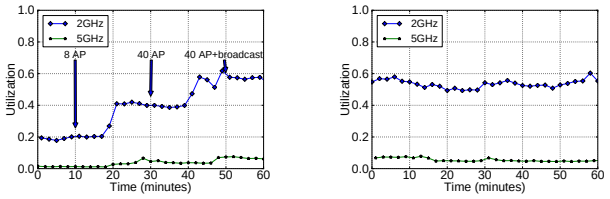
Association Control: Comparing Dual-Band with Band-Steering, we see the importance of association control on the network. When clients are free to choose (*BeHop - Dual Band*), some of them choose the suboptimal 2.4GHz band, leading to worse performance, especially on the tail (0.9th percentile is 15msecs compared to 9.4msecs when capable clients are forced to the 5GHz band). This is an example of a client making the wrong choice going to the more congested channel, and highlights the effectiveness of a simple association control policy for band steering. We have seen other examples of bad decision making from clients, with the most characteristic being a client that continuously switched APs every few seconds. We defer further discussion on association patterns to future work.

AP Density: Fig. 2 shows that BeHop outperforms the Stanford legacy network, which also supports dual-band and band-steering. For instance, median PDT is 8.9msecs, four times larger than the one our users experience. The main reason for that is the density of the physical deployment. Despite 5GHz support from the legacy network, only 30% of the overall traffic goes over it (341GB out of 1.1TB over a one week period). Even when comparing with BeHop at 2.4GHz, the legacy network observes higher latency. Unlike BeHop users that have an AP inside their rooms (39 in total), legacy users share 25 APs spread across the entire building. This results in poor-coverage for the low-penetration 5GHz-band, and often weak-links and low WiFi rates as clients associate with APs which are far away. We go back to the BeHop network to get a sense of coverage between the two bands. For every AP in our network, we monitor the beacons received by other APs over a 10-minute period at 3:00am, and look at the average SNR for each link. From this, we can answer the question: "if a client was located here, how many APs could potentially serve it?" We use this number to characterize the **density** of the network. Fig. 4 shows observed densities on both bands for a threshold of 15dB. Each line resembles a $>$ 15dB link between two nodes with its thickness proportional to the SNR. [6] While 39 APs are enough to cover the whole building in the 2.4GHz band, they are clearly inefficient to do the same on the 5GHz channels. Given that the legacy network is even sparser (25 APs), it should come as no surprise that legacy users opt to use the inefficient but stronger 2.4GHz channels.

So far, BeHop enabled us to better understand and evaluate anecdotal evidences regarding the benefits and properties of 5GHz. We verified that clients connected to the 5GHz band observe better performance. We saw that even when available, some clients prefer to use the 2GHz band instead. We got a better sense of how deployment density affects use of 5GHz and therefore overall network performance. We will now discuss how our testbed revealed a less obvious challenge faced by most modern deployments, namely the trade-off between 5GHz coverage and 2GHz overhead and interference.

[6]We hide links with smaller SNR for clarity.

4.2 Usecase: Studying 2GHz Broadcast Overhead



(a) Channel utilization due to beaconing and broadcast traffic.

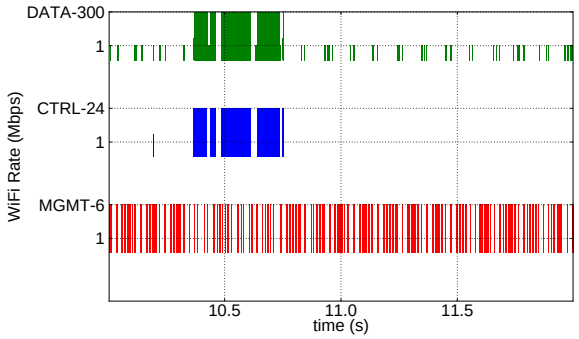
(b) Baseline utilization at Stanford's CS department during a quiet night hour.

Figure 5: Broadcast overhead and channel utilization. Impact on 2GHz band is higher due to lower rates and propagation characteristics.

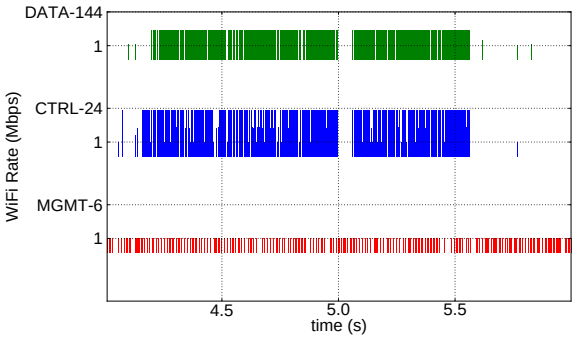
As we saw earlier, high density is necessary to achieve good coverage on the 5GHz band. But along with density, overhead from broadcast traffic on channel utilization also increases. While growing our deployment, we observed a growing baseline overhead. There are two main sources of overhead in WiFi networks — WiFi beacons and normal broadcast traffic (e.g. ARP, mDNS). Each (virtual) AP sends a beacon every 100msecs at the lowest available rate. Due to better propagation characteristics and lower rates, the impact on the 2.4GHz band is higher. Fig. 5 shows the broadcast traffic captured on the wired backend of our network, and channel utilization in both bands (up to 60% for the 2.4GHz band). To reduce the overhead, we increased the beacon interval from 100msecs to 500msecs (which is allowed by the 802.11 standard). To our surprise, this did not work well; most client devices interpet this as a lossy link and disconnect from our APs after a while.

Preceding observations highlight an interesting trade-off between 5GHz coverage and 2GHz overhead. While high density is necessary to use the 5GHz band, it leads to high overhead and interference on the 2GHz band. We expect this to be common, especially on enterprise setups with large broadcast domains, where each AP serves both bands by default. To verify our assumption, we looked at Stanford's Computer Science department, which runs an enterprise-grade WiFi network [5] with 44 APs spread across the spectrum and serving 4 SSIDs each. We observed a 50% baseline utilization on 2GHz, degrading performance of clients using this band.

BeHop helped us identify a real problem which we would likely ignore without a real-world deployment. Moreover, we were able to identify the limitations of an obvious solution, that of increasing density of dual-band APs for better 5GHz

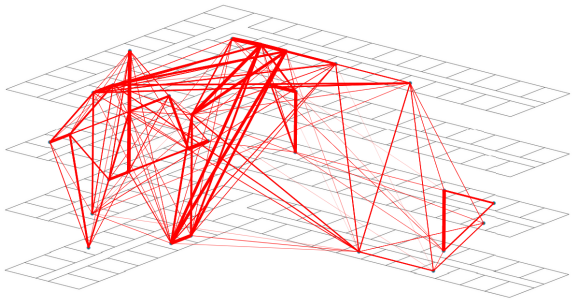


(a) 2-seconds zoom-in on the 5GHz band while streaming a movie from Netflix.

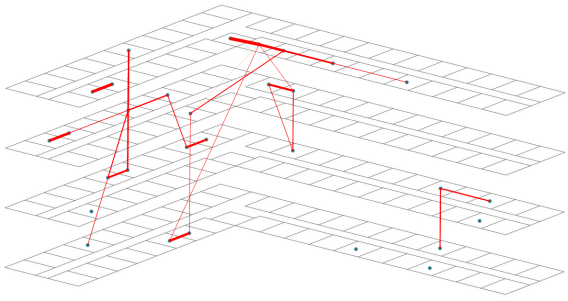


(b) 2-seconds zoom-in on the 2.4GHz band while streaming a movie from Netflix.

Figure 3: 2.4GHz vs. 5GHz: Higher rates, less congestion and lower interference make the 5GHz band more efficient than the 2.4GHz. While serving the same amount of traffic, 2.4GHz approaches maximum utilization while 5GHz remains mostly free.



(a) WiFi density at 2GHz.



(b) WiFi density at 5GHz.

Figure 4: WiFi density at 2.4GHz and 5GHz. Each line represents a link $>$ 15dB with thickness proportional to the observed SNR. 39 APs are enough to provide sufficient coverage for the whole building on the 2.4GHz band, but fail to do so at 5GHz.

converage. We believe that deploying real-world testbeds can be really valuable in exposing and studying such issues met in WiFi networks.

5. RELATED WORK

Management of WiFi networks has been widely studied in the past. MDG [3] combines channel, power, and association control to improve performance. Others [1] have suggested power control to reduce interference between neighboring home networks. DenseAP [9] argues that dense deployment of APs can significantly improve performance. It uses a centralized controller to control client associations through Probe Responses. Most of these testbeds are designed with focus on very specific ideas, and use test-only traffic and clients for evaluation, ignoring issues such as device mobility, broadcast overhead, and client diversity. The main difference between BeHop and previous testbeds is that we are incorporating the challenges of implementing these policies on a real network. Our work is complementary to previous efforts: we want to borrow ideas from earlier studies and examine how efficient and/or challenging they are in real deployments.

There have been a few real-world testbed deployments in the recent past. TFA [2] is a wide-area mesh WiFi deployment. TFA's research focus has been on optimizing topology control. Our work is complementary. We take a user-centric approach where the focus is on controlling clients on a fine-grained basis and studying wireless management. Also, we emphasize seamless inter-operation with existing production networks. Our testbed facilitates unique experiments that can eventually benefit large scale deployments such as TFA. WiSe [10] is a residential deployment with a similar goal of conducting in-line measurements as opposed to passive measurements with sniffers. However, while WiSe is focused on measurement of user-deployed and user-configured APs, we configure and manage APs from the infrastructure side. One of our primary goals is to make it easy to implement management schemes that can improve performance.

More recent testbeds follow the SDN approach. OpenRoads [14] uses OpenFlow WiFi APs and WiMax basestations to experiment with flow mobility and use of multiple networks and interfaces. OpenRoads looks at L2-layer and above, taking the default WiFi operation for granted. We are mostly interested in understanding how WiFi manage-

ment affects network performance. ODIN [13], a framework for enterprise WiFi, exposes a BSSID for every client, which is then treated as a port by an SDN controller. Our Virtual-AP abstraction is similar but it provides a more generic interface: we can map multiple clients to a Virtual-AP, or expose a subset of APs and let clients decide how to roam between them. This is necessary both in terms of experimentation (to realize different techniques), and actual management of the network.

Orbit [11] is a radio-grid emulator open to researchers and focuses on reproducible experiments. While it can be further enhanced with channel impairment models and simulated mobile nodes, it cannot capture the operating conditions of a real network. Authors in [12] use SDN to turn the (wired) production network into a testbed, and get the benefits of real deployment and traffic. Our approach is a middle-step towards this direction to account for existing networks not having the flexibility to host our experiments.

Finally, commercial WiFi systems use centralized controllers to optimize network management and performance [7, 5]. These systems claim to support some low-level AP management features such as band-steering and mobility support. However, they are closed and proprietary, and exact details on their inner workings are hard to obtain. BeHop is an open testbed based on off-the-shelf hardware, OpenWRT and OpenFlow. Also, while commercial WiFi systems focus on enterprise environments, BeHop can be tailored both for enterprise and residential usecases.

6. CONCLUSION

In this paper, we presented BeHop, a testbed for dense WiFi networks. BeHop provides a general-purpose framework to experiment with a wide range of techniques for channel, power, and association control. By integrating BeHop with the production network, we get the benefits of a real-world deployment while keeping the desired flexibility to run experiments. Our early experience with BeHop let us evaluate popular algorithms for WiFi, and revealed trade-offs that were not obvious before. We are confident that production-ready testbeds have great potential to inform the design and efficiecy of WiFi networks, and most techniques from BeHop can be reused to build similar deployments.

7. ACKNOWLEDGEMENT

This research is supported in part by the NSF POMI (Programmable Open Mobile Internet) 2020 Expedition Grant 0832820, and ONRC (Open Networking Research Center). The authors also like to thank Netgear for donating Access Points, Netflix for providing gift cards for participating students, and the Stanford IT department for their support on deploying and maintaining BeHop.

8. REFERENCES

[1] A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-Management in Chaotic Wireless Deployments. In *MobiCom'05*.

[2] O. Bejarano, S. Miskovic, E. Aryafar, and E. W. Knightly. Tfa: A large scale urban mesh network for social and network research. In *Proceedings of the 2010 ACM Workshop on Wireless of the Students, by the Students, for the Students*, S3 '10, pages 49–52, New York, NY, USA, 2010. ACM.

[3] I. Broustis, K. Papagiannaki, S. V. Krishnamurthy, M. Faloutsos, and V. Mhatre. MDG: Measurement-driven Guidelines for 802.11 WLAN Design. MobiCom'07.

[4] http://behop.stanford.edu. BeHop Website.

[5] https://meraki.cisco.com. Cisco Meraki.

[6] https://www.census.gov/. US Census.

[7] http://www.merunetworks.com. Meru Networks.

[8] J. Mccauley. Pox: A python-based openflow controller.

[9] R. Murty, J. Padhye, R. Chandra, A. Wolman, and B. Zill. Designing High Performance Enterprise Wi-Fi Networks. NSDI'08.

[10] A. Patro, S. Govindan, and S. Banerjee. Observing home wireless experience through wifi aps. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, MobiCom '13, pages 339–350, New York, NY, USA, 2013. ACM.

[11] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols. In *WCNC, 2005*.

[12] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. M. Parulkar. Can the production network be the testbed? In *OSDI'10*.

[13] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao. Towards Programmable Enterprise WLANS with Odin. HotSDN '12.

[14] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown. OpenRoads: Empowering research in mobile networks. *ACM CCR 2010*.