Sidekick: AP Aggregation Over Partially Overlapping Channels

Eugene Chai and Kang G. Shin
Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor, MI 48109-2121, U.S.A.
{zontar, kgshin}@eecs.umich.edu

***Abstract*—The uncoordinated deployment of many high-bandwidth 802.11a/g/n access points (APs) in urban areas offers the potential for WLANs to be a strong complement to cellular networks in providing ubiquitous connectivity. However, given that the bandwidth of the backhaul links connected to these APs is often an order-of-magnitude lower than that of the WLAN channel, aggregating the throughput from multiple APs is often necessary in order for the client to achieve an acceptable level of network performance.**

In this paper, we present *Sidekick* —a simple and novel AP aggregation protocol that exploits effective communication between 802.11a/g/n nodes on partially overlapping channels to attain high aggregate throughput in the face of dynamic WLAN and backhaul link conditions. *Sidekick* is built upon *Aileron*, which provides an extremely reliable and low-overhead control channel over which the APs and clients can coordinate the aggregation process. The use of such a control channel over partially overlapping channels enables *Sidekick* to quickly respond to varying bandwidth availability and probe for new transmission opportunities with little overhead. Our evaluation results indicate that *Sidekick* can make more than 30% improvement in throughput over FatVAP in a variety of situations.

I. Introduction

The proliferation of unplanned high bandwidth 802.11a/g/n APs in urban areas offers the potential for WLANs to be strong complement to cellular networks in providing ubiquitous connectivity [1], [2], [3]. However, this potential has to be tempered by the fact that the APs (a) are deployed chaotically and are not under any centralized control, (b) are connected to broadband backhaul links with bandwidths that are significantly lower than that of the WLAN channels, and (c) can be cellular 4G routers where the backhaul link, being an LTE or WiMAX channel, is subject to the usual vagaries of wireless networks. For example, 802.11n can achieve a throughput of at least 300Mbps [4], which is typically an order-of-magnitude higher than that of broadband backhaul networks.

Wireless clients can overcome this limitation by aggregating backhaul links from multiple APs [5], [6]. In such a protocol, a WLAN client connects to multiple APs, one at a time, with the order and duration of each connection determined by the parameters—such as bandwidth, queue length, congestion, etc.—of both the backhaul and the WLAN channel. However, two significant obstacles stand in the way of the efficient scheduling of connectivity across multiple APs with only one WLAN interface on the client. First, the client node can typically only communicate with one AP at a time. This gives rise to an obvious chicken-and-egg conflict: the client needs to know the available bandwidth from an AP before it can construct a connection schedule, but it can only know the available bandwidth after it has connected to the AP and measured or downloaded traffic statistics. Second, the time-varying nature of traffic on both the wireless and the backhaul links means that an aggregating client who only obtains bandwidth information after its AP association will never be able to track the bandwidth variation accurately and thus, cannot adjust its connection schedule to maximize the achievable backhaul throughput. Figure 1 illustrates the number of bytes downloaded by a static Bittorrent client in consecutive $100ms$ intervals over a WiMAX network in Korea. Note that the steady-state bandwidth can vary by more than three orders-of-magnitude and change significantly as seen at the 1000s mark. Hence, an efficient and accurate method of measuring the available backhaul and WLAN bandwidths is of paramount importance to effective aggregation of bandwidth from multiple WLAN APs.

AP aggregation is further complicated by the growing acknowledgment that fine-grained channelization and dynamic spectrum access [7], [8], [9] is critical to enhancing the utilization of wireless channels. Such fine-grained spectrum-usage patterns increase the chance of interference from partially overlapping transmissions, which are not decodable by current PHY/MAC protocols.

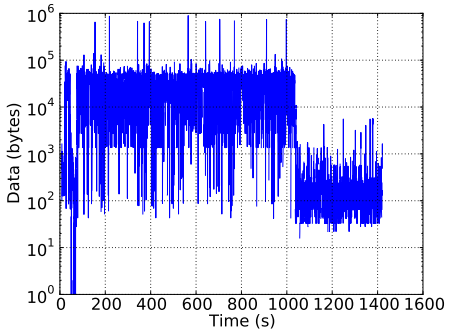


Fig. 1. Number of bytes received by a static Bittorret client in consecutive $100ms$ windows over a WiMAX network in Seoul [10].

In this paper, we present *Sidekick*—a simple yet novel

802.11a/g/n AP aggregation protocol that achieves efficient multi-AP communications by enabling the APs to take an active role in aggregation by notifying clients of the exact number of backlogged packets through an in-band signaling channel that is based on *Aileron* [11]. A key innovation here comes from the fact that the clients *need not be on the same channel as the AP to receive this status information*. The in-band signaling technique can efficiently and accurately convey bandwidth information to clients that are tuned to channels that only partially overlap with the channel of the AP. *Sidekick* also includes a MAC-layer protocol that integrates this real-time traffic information into an optimal schedule that maximizes the achievable throughput over multiple APs.

Sidekick offers the following benefits over existing AP aggregation techniques:

Retrieval of traffic information over partially-overlapping channels. *Sidekick* nodes can exchange traffic information as long as the spectrum of the channel used by the AP partially overlaps with the spectrum used by the client; the client and AP do not have to be tuned to the same channel. Communication through partially overlapping channels has been used in [12], but that method is only applicable to the older 802.11b standard and cannot be employed with OFDM-based 802.11a/g/n networks. *Aileron* offers a novel and reliable signaling channel with a performance that is independent of the bandwidth of the overlapping spectrum.

Low overhead signaling. *Sidekick* nodes can exchange traffic information with very low overhead. With *Aileron*, a *Sidekick* AP can embed queue length information in a side-channel using the RTS/CTS or data frames that are used for regular co-channel transmissions; *Sidekick* clients on partially overlapping channels can extract this queue information from the side-channel without requiring any additional signaling or synchronization bits. This feature stands in stark contrast with regular co-channel communications where proper synchronization in the form of a known preamble along with channel access procedures involving SIFS and DIFS delays are needed to accurately transmit network state information from an AP to a client.

Accurate tracking of time-varying channel state. As a net result of effect communication over partially overlapping channels and low-overhead signaling, a *Sidekick* client can efficiently determine the number of queued packets for itself at every AP with minimal probing overhead.

The rest of the paper is structured as follows. We discuss related work and some background information in §II and §III, respectively. We then give details on the design of *Sidekick* in §IV and §V. We evaluate *Sidekick* in §VI and §VII before concluding the paper in §VIII.

II. Related Work

Multi-Net [13] is the first virtualization platform for wireless interfaces. It consists of a specially crafted device driver that exposes multiple virtual devices, one for each available AP, to the rest of the network stack; a fixed and an adaptive scheme are used to govern the switching policies among different APs.

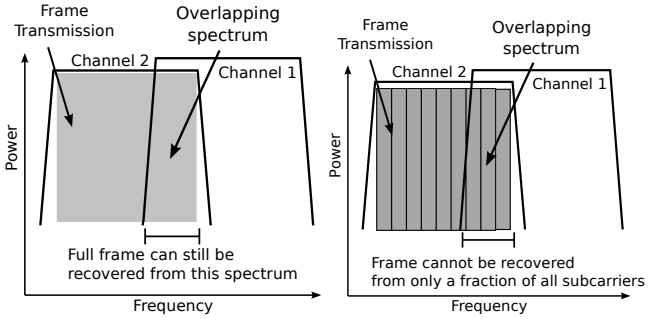


Fig. 2. Partially overlapping channels in (Left) 802.11b and (Right) 802.11a/g/n OFDM networks.

Juggler [14] is built upon Multi-Net and improves its ability to quickly switch between multiple APs, thereby allowing efficient use of AP aggregation under dynamically changing network conditions. FatVAP [5] is an AP aggregation scheme that focuses on achieving maximum aggregate throughput by optimizing the duration and the order of AP connections using dynamic programming. Arbor [15] is a similar aggregation scheme with the added focus on aggregation over secure wireless networks. THEMIS [6] takes a different approach with a focus on fairness between multiple aggregating clients; in a blind aggregation scheme such as FatVAP well-connected clients can easily consume an excessive amount of bandwidth at the expense of more poorly connected clients.

ViFi [16] is an extension of this multi-AP aggregation concept to the mobile scenario: it exploits the diversity offered by simultaneous use of multiple APs to provide continuous WiFi access to moving vehicles. JellyNets [17] is another interesting integration of AP aggregation with pocket hypervisors on mobile devices.

III. Background

In this section, we discuss the challenge faced by *Sidekick* when communicating over partially-overlapping OFDM networks as well as the *Aileron* protocol that allows *Sidekick* to maintain a reliable signaling channel even when the AP and client are not on the same channel.

A. Communicating Over Partially-Overlapping Channels

Partially-overlapping channels in 802.11a/g/n WLANs pose a significantly larger problem than that in 802.11b WLANs. In 802.11b WLANs, the transmitted frame is modulated using the Direct Sequence Spread Spectrum (DSSS) technique. Here, we differentate between the *information signal* and *transmitted signal*. The information signal refers to the modulation version of the data frame; this information signal is multiplied with a pseudonoise sequence to produce a transmitted signal that is sent over the wireless channel. The energy of the information signal is spread by the pseudonoise sequence such that the transmitted signal occupies a larger bandwidth than the information signal. This modulation is done to ensure that transmissions are resilient to narrow-band noise. At the destination node, the information signal is recovered by correlating the received signal with a complex conjugate of the same pseudorandom sequence; any narrow-band noise will not be

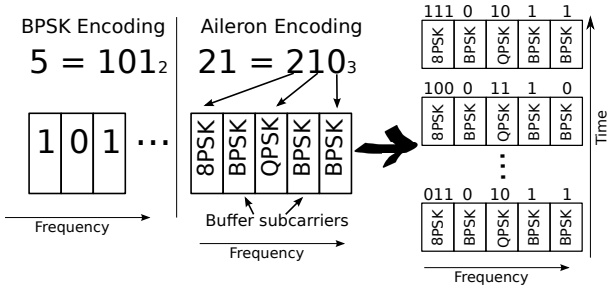


Fig. 3. Information encoding using *Aileron* and typical digital communication techniques.

highly correlated with the pseudonoise sequence and will thus be filtered out.

A key property of an 802.11b transmission is its information homogeneity: the original information signal can be recovered using sufficient energy from any portion of the received spectrum. Figure 2(Left) illustrates a situation involving two partially-overlapping 802.11b channels — if the bandwidth of the overlapping spectrum is sufficiently wide, then the transmission on Channel 2 can be recovered from Channel 1 from the energy in the overlapping spectrum. The overlap between the two 802.11b channels can be modeled using the I-factor metric [12], which is a measure of the fraction of energy of the transmitted signal that can be recovered by a receiver in the overlapping channel center.

Due to the use of orthogonal subcarriers, the spectral information content of an 802.11g/n transmission is not homogeneous across the entire frequency band. Figure 2(Right) shows an example of two partially-overlapping OFDM channels. A partially-overlapping channel (Channel 1, in this case) can only recover a fraction of all the subcarriers used in the transmission in Channel 2, and therefore is not able to recover the transmission. This poses two significant obstacles to communication across partially-overlapping channels as folllows.

(a) Frames cannot be detected. A preamble is typically used to demarcate the start of an OFDM frame. At the receiver, the received signal is correlated with a time-domain representation of the known preamble; correlation peaks indicate the start of a data frame. Since only a fraction of all subcarriers are present in the overlapping band, a receiver that is searching for frames using correlation will fail because the omission of subcarriers will change the time-domain representation of the preamble.

(b) Frames cannot be decoded. Even if a receiver is able to detect the OFDM frame, it will not be able to correctly decode it. This is because information in each subcarrier is not repeated on any other subcarrier, and the partially-overlapping receiver cannot decode the original frame using only a fraction of the subcarriers. Hence, in typical 802.11a/g/n networks, communicating between partially-overlapping channels is not possible, regardless of the I-factor value. Co-channel transmissions can be recovered using subcarrier remapping [18], but this method still does not enable communications between partially-overlapping channels.

B. Aileron: Low Overhead Control Channels

Aileron [11] is a novel signaling protocol that transmits information by encoding them in the *modulation rate* of the OFDM subcarriers. This is in contrast to typical digital communication techniques where information is encoded in the *symbol values* carried by each OFDM subcarrier. *Aileron* encodes information in a base-3 notation, with binary, quadrature and 8-phase shift keying serving as the basis values. This is analogous to the integers 0 and 1 that form the basis for a binary (base-2) number system used in modern communication networks. In *Aileron*, BPSK, QPSK and 8PSK are mapped to integers 0, 1 and 2, respectively.

Figure 3 compares the encoding process used by *Aileron* with that found in typical communication systems.

Typical digital communications. Consider, for example, that the base-10 integer 5 is to be transmitted over an OFDM network. Assume that each OFDM subcarrier carries information using the QPSK modulation scheme. The base-10 integer 5 is first converted to the base-2 number 101_2. Adjacent pairs of bits are then mapped to a QPSK symbol value and stored in a OFDM subcarrier. The final set of OFDM subcarriers are then transmitted over the wireless channel using standard PHY-layer techniques.

Signaling using Aileron. *Aileron* embeds a low-rate control message in the data stream that is used to transmit a regular data frame over an OFDM link. We differentiate between the *data* stream and the *control* stream that are sent over the channel. This embedding is carried out with a two-stage process. In the first stage, the control messages are mapped to the *modulation rates* used by each OFDM subcarrier. Assume that a base-10 integer 21 is encoded by *Aileron* and transmitted over an OFDM channel. Its base-3 form is 210_3 where the leftmost integer, 2_3, is the most significant integer. Each base-3 integer is then mapped to the modulation rate of a subcarrier: 0, 1 and 2 are mapped to BPSK, QPSK and 8PSK, respectively. These base-3 values only determine the modulation rate of the subcarriers; any constellation point from the chosen modulation scheme can be selected. In the second stage, the bits in the data stream are partitioned according to the modulation rate of the different subcarriers and encoded accordingly. Specifically, 1, 2 and 3 bits are assigned to subcarriers that are to be modulated using BPSK, QPSK and 8PSK, respectively. The modulation rates determined by the control stream are used to transmit at least 10 OFDM symbols, as shown in Figure 3. The receiver collects these OFDM symbols and determines the modulation rate used in each subcarrier using statistical inference techniques.

Aileron is an effective and accurate method of signaling across partially-overlapping channels with little overhead. With *Aileron*, the control message on an OFDM channel can be both detected and decoded even if the receiver cannot recover all the subcarriers used in the data stream. *Sidekick* uses the low-rate signaling channel to transmit information on the queue length from the AP to the client. The data stream is used for normal co-channel communication between the AP and the clients that are associated with it.

Preliminary real-world evaluations with a single client has shown that *Sidekick*, using *Aileron*, can detect BPSK, QPSK

and 8PSK with an accuracy of 100%, 98% and 86% [11] respectively.

IV. Sidekick MAC Protocol

Sidekick consists of both PHY and MAC-layer protocols. The PHY-layer design enables accurate communication across multipe partially overlapping channels while the MAC-layer harnesses this ability to efficiently aggregate multiple backhaul links across different APs. The *connection schedule* computed by each *Sidekick* client determines both the duration and the order in which the client connects to the multiple APs. We present two different algorithms for computing the schedule, `Sidekick-ILP` and `Sidekick-Greedy`. `Sidekick-ILP` constructs the schedule using an Integer-Linear Program (ILP), similar to that used by FatVAP [5], while `Sidekick-Greedy` visits the APs greedily in order of decreasing queue length.

A. Overview

We consider a scenario with with N *Sidekick* APs $X_1, \ldots, X_N$ and a single *Sidekick* client. A single wired backhaul link is connected to each AP. Each AP X_i, $i \in \{1, \ldots, N\}$ has a backhaul link with throughput of b_i. This backhaul link can be wired, as is the case for home broadband networks, or wireless, as is the case for 3/4G routers. The wireless throughput between X_i and the client is denoted by w_i. In order for the aggregation of multiple backhaul links to be feasible, the inequality $b_i < w_i$ must be met. As is the case with ordinary WLAN clients, the *Sidekick* client is assumed to know the channel of each available AP.

The connection schedule is represented by a pair of lists (P, D). P is a list of APs to be visited and each $X_i \in P$ has a corresponding entry $t_i \in D$ representing the length of time that the client should remain connected to AP X_i. When a *Sidekick* client switches away from an AP, it uses the 802.11 power-save mode feature to ensure that packets that arrive at the AP in its absence are buffered.

B. Sidekick-ILP

Given APs $X_1, \ldots, X_N$, each with backhaul and wireless bandwidths b_i and w_i, $i \in \{1, \ldots, N\}$, respectively, the schedule can be computed using an algorithm similar to that used in [5]:

$$\max \quad \sum_i f_i p_i \quad \text{s.t.} \tag{1}$$

$$\sum_i (f_i T + \lceil f_i \rceil s) = T \tag{2}$$

$$\forall i \quad 0 \leq f_i \leq \min\left\{\frac{q_i}{p_i}, 1\right\} \tag{3}$$

where s is the delay incurred when switching from one AP to another, T is the time quantum of the schedule, q_i is the length of the queue at AP X_i and $p_i = w_i T$ is the maximum number of packets that can be transmitted from X_i to the client within the time duration T. The connection schedule is then constructed from the solution of the optimization algorithm as (P, D) where $P = [X_1, \ldots, X_N]$ and $D = [f_1, \ldots, f_N]$.

This optimization algorithm seeks to maximize the total number of packets downloaded within a time interval T by determining the optimal length of the duty cycle, $f_i T$, that should be spent at each AP X_i. The length of this duty cycle is proportional to the ratio of the current queue length to the maximum number of packets that can be transmitted over the wireless link within one time quantum. This time quantum, T, determines the maximum duration of all duty cycles and is an upper bound on the TCP acknowledgement delay from the wireless node. We select $T = 100ms$ so that a fair performance comparison can be made with FatVAP. The constraint (2) ensures that the time consumed by the duty cycles and the switching overhead do not exceed the stated time quantum.

The optimization algorithm shown here does not explicitly ensure an upper bound on the time interval between two consecutive visits by the client to the same AP. Hence, it is possible for the length of the queue at some AP X_i to grow beyond the number of packets that can be transmitted over the wireless link within one time quantum. The resulting ratio $q_i/p_i > 1$ will cause constraint (2) to be violated. Constraint (3) ensures the feasibility of the optimization by restricting the upper bound of f_i for all APs X_i.

This optimization problem can easily be reformulated as an Integer-Linear Program

$$\max \quad \sum_i f_i p_i \quad \text{s.t.} \tag{4}$$

$$\sum_i (f_i T + y_i s) = T \tag{5}$$

$$\forall i \quad 0 \leq f_i \leq \min\left\{\frac{q_i}{p_i}, 1\right\} \tag{6}$$

$$f_i \leq y_i \leq 1, \; y_i \in \mathbb{Z}, \tag{7}$$

thus allowing the use of off-the-shelf optimization routines.

C. Sidekick-Greedy

Sidekick clients have up-to-date information on the length of the packet queues at the APs. Hence, a simple greedy algorithm can also be employed where the client connects to APs in decreasing order of queue lengths. In contrast to `Sidekick-ILP`, `Sidekick-Greedy` returns an *ordered* connection schedule; the AP connections under `Sidekick-ILP` are not guaranteed to be carried out in any particular order. Fig. 4 shows the pseudocode for `Sidekick-Greedy`.

In `Sidekick-Greedy`, a max-heap is used to keep track APs, in decreasing order of queue lengths, that have not yet been scheduled. For each AP X_i at the top of the heap, the total time needed to empty the queue, T_i is calculated first. If this time T_i can fit into the current schedule without the total schedule time exceeding the time quantum T, then X_i and T_i are appended to the schedule lists P and D, respectively. Otherwise, the remaining available time in the schedule, if any, is assigned to X_i and the completed connection schedule is returned.

input : The queue length, $Q = q_1, \ldots, q_N$, and bit rate, $R = r_1, \ldots, r_N$, of each AP X_i, $i \in 1, \ldots, N$
output: The connection schedule P and connection duration D for all APs

begin
  $T \leftarrow$ time quantum, $s \leftarrow$ switching time;
  $h \leftarrow$ `make_max_heap`(Q), $t \leftarrow 0$;
  $P \leftarrow$ `empty_list`(), $D \leftarrow$ `empty_list`();
  while h *is not empty* **do**
    $q_i \leftarrow$ `pop_heap`(h);
    $T_i \leftarrow q_i / r_i$;
    if $t + T_i + s \leq T$ **then**
      $t \leftarrow t + T_i + s$;
      $P \leftarrow$ `append`(P, X_i);
      $D \leftarrow$ `append`$(D, q_i / r_i)$;
    else
      $P \leftarrow$ `append`(P, X_i);
      $D \leftarrow$ `append`$(D, T - t - s)$;
      $t \leftarrow t + T_i + s$;
      break;
    end
  end
  return (P, D);
end

Fig. 4. `Sidekick-Greedy` algorithm.

D. Using the Entire Time Quantum

Under both `Sidekick-ILP` and `Sidekick-Greedy`, the total connection time in the schedule may be less than the time quantum. Hence, we adjust the connection times of the client to each AP to be proportional to the relative queue length of that AP. The pseudocode for this step is shown in Fig. 5.

This adjustment to the connection schedule is made to improve the overall utilization of the wireless channel. In the adjusted schedule (P, D'), the client visits each AP once during each time quantum, as opposed to multiple times per time quantum without the adjustment, and therefore, reduces the time wasted on the AP switching.

input : Connection schedule (P, D).
output: Adjusted connection schedule (P, D') such that $|D| \cdot s + \sum_{d_i \in D'} d_i = T$, where T is the time quantum and s is the switching delay.

begin
  $T_D \leftarrow |D| \cdot s + \sum_{d_i \in D} d_i$;
  $T_R \leftarrow T - T_S$;
  $D' \leftarrow$ `empty_list`();
  for $k \in 1, \ldots, |D|$ **do**
    $D'[k] \leftarrow D[k] + (D[k]/T_D) \cdot T_R$;
  end
  return (P, D');
end

Fig. 5. Adjusting the connection schedule to ensure that the entire time quantum is utilized.

E. Responding to Bandwidth Changes

Sidekick uses partially overlapping channels for control messages, thus adapting to varying bandwidth is an integral portion of the *Sidekick* protocol. When a new *Sidekick* AP comes online, *Sidekick* nodes exchange information on the bandwidth increases using a protocol that has two distinct portions: a broadcast protocol that is run on the AP and an adaptation protocol that is run on the client.

The broadcast protocol used by the AP is straightforward: an AP broadcasts its available aggregation capacity by piggybacking such notifications on the RTS/CTS frames that are used for co-channel communication. Such broadcasts occur at least once per time quantum. If the co-channel transmission rate is lower than one packet per time quantum, the AP will broadcast its available capacity using a special short broadcast frame. This frame will be described in §V.

The adaptation protocol running on the client responds to these broadcast messages and adds the newly-available APs to the pool of APs considered by the scheduling algorithms. The client assigns a new TCP flow to each new AP that broadcasts its availability. New APs are added to the scheduling algorithm one at a time. This is to ensure that the client can allocate sufficient connection time to an AP to allow TCP to quickly run through its slow-start phase to reach its steady-state transmission rate. When a new-AP broadcast is detected by a client, it connects to the AP for a duration of $T/2$ and starts a new TCP connection through that AP. This AP is then added to the pool of APs for use by the next iteration of the scheduler. If multiple broadcasts are detected, the APs are added one at a time in a random order, with only one AP added between consecutive calls to the scheduling algorithm.

If no packets are detected from an AP for a duration of $10T$, the AP is assumed to be offline and will be removed from the pool of APs used by subsequent invocations of the scheduling algorithm.

F. Overall Sidekick MAC Protocol

input : Time quantum, T

begin
  while true do
    if *New AP* **then**
      Broadcast available capacity;
      `sleep` (T);
    else if *Transmitting RTS or CTS frame* **then**
      Embed queue lengths and IDs of at most 8 randomly selected clients into the RTS/CTS frame;
    end
    else if *No RTS/CTS transmission for T seconds* **then**
      Broadcast queue lengths and IDs of at most 8 randomly selected clients into a *Sidekick* broadcast frame;
    end
  end
end

Fig. 6. Sidekick MAC protocol on the access point

Figures 6 and 7 show the pseudocode of the overall *Sidekick* MAC for the AP and the client, respectively. Note that the *Sidekick* client needs to handle the situation where the length

```
   input : Time quantum, T
1  new_ap ← empty_list ();
2  active_aps ← empty_list ();
3  ap_queue ← empty_list ();
4  /* Wireless bandwidth                                   */
5  wl_rate ← empty_list ();
6  while true do
7      if Broadcast from new AP, X, received then
8          new_ap ← append (new_ap, X);
9      end
10     if |new_ap| > 0 then
11         X ← remove_head(new_ap);
12         Connect to X for T/2 seconds and start new TCP
           connection;
13         Q_X ← queue length of X;
14         R_X ← wireless bandwidth between client and X;
15         active_aps ← append(active_aps, X);
16         ap_queue ← append(ap_queue, Q_X);
17         wl_rate ← append(wl_rate, R_X);
18     end
19     if length of all data queues of active APs is zero then
20         Associate with a random AP and wait for queue length
           updates via control messages ;
21     end
22     (P, D) ← Sidekick-ILP (ap_queue, wl_rate) or
       Sidekick-Greedy (ap_queue, wl_rate);
23     Connect to the APs according to the connection schedule
       (P, D);
24     if No queue length update from X for 10T seconds then
25         Remote R_X, Q_X and X from wl_rate, ap_queue and
           active_aps respectively;
26     end
27 end
```

Fig. 7. Sidekick MAC protocol on the client

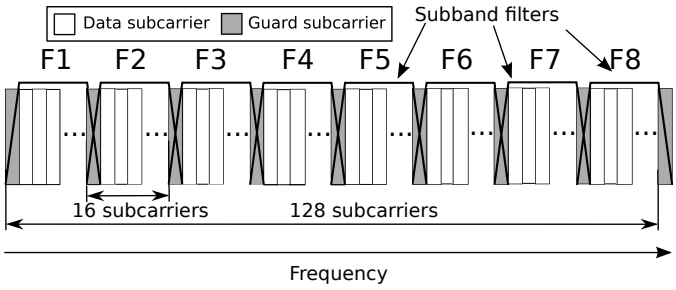


Fig. 8. PHY-layer signaling frame.

of all queues of active APs are zero (lines 19 and 20 of Fig. 7). This can occur sporadically due to the bursty nature of TCP packet arrivals and the fact that the wireless bandwidth can be significantly larger than the backhaul bandwidth.

V. Sidekick PHY Protocol

A. Design of the Control Channel

Sidekick uses two different control messages to convey queue information from the AP to the client: broadcast and directed. Broadcast messages are used by APs to notify clients of new transmission opportunities and are described in §IV-E; directed messages are sent from an AP to a specific client and are used to notify the client of the number of its packets queued at the AP.

The PHY-layer design of the two messages is illustrated in Fig. 8. Here, we focus on the 40MHz channel of the 802.11n network with 128 OFDM subcarriers, but the control message design is similar for networks of other bandwidths.

A key feature of *Sidekick* is its ability to pass queue length information between the AP and the client nodes regardless of the bandwidth of the overlapping spectrum between the AP and client using *Aileron*. Recall from §III-B that *Aileron* encodes information in the modulation of the subcarrier rather than its symbol value. In 802.11n WLANs, adjacent channels are separated by 5MHz, which is spanned by 16 subcarriers. *Sidekick* takes advantage of this fact and divides the 128 subcarriers into 8 groups of 16 subcarriers each. We refer to each group of 16 subcarriers as a *subcarrier group*. In order to minimize interference between adjacent subcarrier groups, a single subcarrier between two adjacent subcarrier groups is designated as the guard subcarrier and is not used for data transmission. Of the 15 remaining subcarriers, 8 are used as spacing subcarriers, as required by *Aileron*, and are only modulated with BPSK; the other 7 data subcarriers can be encoded with either BPSK, QPSK or 8PSK as described in §III-B.

Sidekick uses these 7 data subcarriers as follows: 3 subcarriers are used for an address, which is the client address in a directed message, or a special broadcast address for broadcast messages; 4 subcarriers are used to encode the queue length. *Sidekick* can therefore transmit queue lengths of up to 16 packets and any queue containing more than 16 packets is simply encoded using the largest supported value. Note that the division of subcarriers between client addresses and queue lengths can be varied according to the network configuration. We leave such configuration details to future work.

Sidekick encodes a different client address and associated queue length in each subcarrier group, as shown in Fig. 8, with a different random mapping between client addresses and subcarrier groups in every control message.

Aileron constructs the control message by repeating the PHY-layer layout and modulation encoding of Fig. 8 in at least 10 consecutive OFDM symbols. *Sidekick* transmits these control messages using two possible methods: embedded into an RTS/CTS frame or as a separate control frame. With embedded transmission, the modulation used in subcarriers of the RTS/CTS frame are set according to that shown in Fig. 8; if a separate control frame is used, *Sidekick* transmits 10 consecutive OFDM frames carrying random data, with the modulation rates of the subcarriers also set as shown in Fig. 8.

B. Addressing the APs

The source address of partially overlapping *Sidekick* APs can be transmitted in three different ways:

(a) Encoded using *Aileron*. Some of the subcarriers in each overlapping subcarrier group can be used for encoding the AP IDs. We can also increase the total number of subcarriers in each OFDM symbol to obtain more subcarriers for encoding the AP/client addresses and the queue lengths.

(b) AP-specific preambles. Each AP can use a unique preamble that is repeated in every subcarrier group. This preamble

is generated based on the ID of the AP.

(c) Fixed channel-to-AP mapping. Each *Sidekick* AP can be assigned to a unique channel that is not occupied by any other *Sidekick* AP. The address of the AP can then be inferred from the offset of overlapping AP transmission from the channel of the *Sidekick* client. The advantage of this approach is that no additional subcarriers are needed to encode the AP address.

C. Receiving Control Messages

There are three key steps for a client to correctly decode the queue length information in the *Aileron* packet: detecting the transmission, finding the edge of the partially overlapping packet and finally decoding the modulation-encoded message.

Detection. The RTS/CTS and separate *Aileron* control frames are significantly shorter than the standard WLAN data frames. Hence, the *Sidekick* client can differentiate control from data frames from the duration of the energy burst [19]. After the transmission has been detected, edge detection is carried out.

Edge Detection. A partially overlapping transmission will only occupy a fraction of all the OFDM subcarriers available to the client. Edge detection enables the client to determine the subcarrier groups that contain a valid transmission from an AP. The client, when operating over a 40MHz 802.11n channel, uses 8 channel filters, each spanning a 5MHz bandwidth. Fig. 8 illustrates the arrangment of these filters as well as the associated labels $F_1, \ldots F_8$. The edge of the partially overlapping transmission can be determined using the algorithm shown in Fig. 9. Here, the *lower limit* refers to the edge of a partially overlapped transmission that spans F_1 to F_k for $2 \leq k < 8$, while the *upper limit* refers to the edge of a transmission that spans F_k to F_8 for $1 \leq k < 8$. If the limits cannot be found (lines 18-19), that means that the received *Aileron* control message was transmitted from an AP tuned to the same channel as the client. Note that we assume that the wireless channels of all nodes have the same bandwidths, thus a control message sent over a partially overlapping control cannot have both a upper and lower limit; we leave the case of networks with heterogenous channel bandwidths to future work.

Decoding. Once we have located the boundary of the partially overlapping transmission, we can decode the modulation-encoded message (i.e. client address and queue lengths) from all the subcarrier groups that it occupies. The decoding accuracy depends on the Signal-to-Interference-and-Noise Ratio (SINR) of the channel. The interference is due to other partially overlapping transmissions to the same client node.

D. Multiple Sidekick Clients

For simplicity, our exposition of *Sidekick* has thus far focused on the multi-APs-single-client case. We now give an overview of the simple extensions needed for *Sidekick* to operate in a multi-APs-multi-clients environment. We leave the detailed evaluations of *Sidekick* in this multi-clients scenario as our future work.

Sidekick APs maintain a separate packet queue for each *Sidekick* client. The *Sidekick* APs then embeds the ID of a client and its associated queue length in the transmitted control messages. The *Sidekick* PHY can transmit information on up to 8 different clients in a single broadcast message. If the number of clients is greater than 8, the AP will simply embed queue information on 8 randomly selected clients in each control message.

```
   input : Aileron Control Message
1  begin
2      lower_limit ← −∞;
3      upper_limit ← ∞;
4      for k ← 1 to 7 do
5          if energy(F_k)/energy(F_{k+1}) > δ then
6              lower_limit ← k;
7              break;
8          end
9      end
10     for k ← 8 to max(lower_limit, 2) do
11         if energy(F_k)/energy(F_{k−1}) > δ then
12             upper_limit ← k;
13             break;
14         end
15     end
16     if lower_limit > upper_limit then
17         return NULL;
18     else if lower_limit = −∞ and upper_limit = ∞ then
19         return Co-channel control message received;
20     end
21     return (lower_limit, upper_limit);
22 end
```

Fig. 9. Search for the upper and lower limits of partially overlapping control messages transmitted over a 40MHz 802.11n channel with 8 subcarrier groups.

A *Sidekick* client that decodes this control message can receive information on its queue on an AP if (a) it is one of the 8 random clients selected by the AP and (b) its queue length information lies in the overlapping subcarriers of the client and AP. If a *Sidekick* client does not find its queue information in the control message, it simply omits the current AP from the schedule computation.

In such a scenario, *Sidekick* clients may not have complete information on the state of the AP queues. *Sidekick* will not be able to find a schedule that maximizes the transmission opportunities at the APs, thus resulting in a reduced aggregated throughput. However, we expect this reduced throughput to still be greater than the throughput that can be achieved without *Sidekick* and we leave detailed evaluations to future work.

VI. Evaluation of the *Sidekick* PHY

A. Experimental Setup

We implemented *Sidekick* PHY using GNURadio and evaluated it over simulated channels. This use of simulated channels allows us the flexibility of systematically exploring the performance of the *Sidekick* PHY over a wide range of channel conditions, without any constraints imposed upon us by the physical layouts of our office environment. The parameters used to evaluate the *Sidekick* PHY is summarized in Table I.

In order to evaluate *Sidekick* PHY in a simulated environment, we first generate two partially-overlapping transmis-

PHY Parameter	Value
Center frequency	2.4GHz
Total bandwidth	12.5MHz
Total subcarriers	512
Cyclic prefix length	256
No. of subcarrier groups	8
No. of subcarriers per subchannel	64

TABLE I
PARAMETERS USED IN THE *Sidekick* PHY.

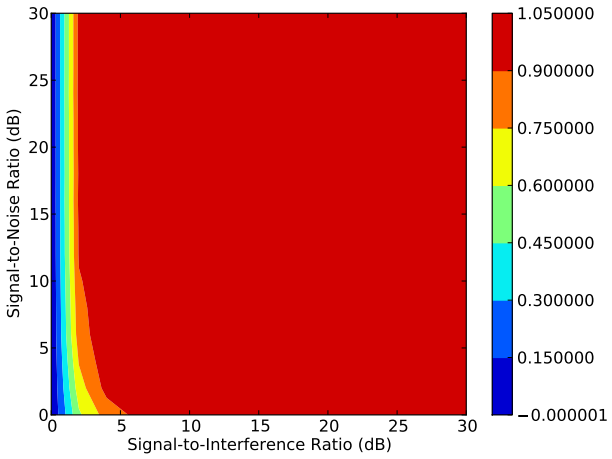


Fig. 10. Probability of correctly detecting the edge of S_D in channels with different interference and noise energy levels

sions: the data transmission, S_D, spans $F_1, \ldots, F_5$ while the other interfering transmission, S_I, spans $F_3, \ldots, F_8$. These streams are passed through a MATLAB filter that combines them and fading and shadowing effects, along with Gaussian noise, to the signal. MATLAB keeps the signal energy of S_D constant while varying that of S_I to produce different Signal-to-Interference values (SIR); the added noise energy is also varied to control the Signal-to-Noise (SNR) of the output signal. This distorted signal is then passed to the *Sidekick* receiver where the original modulation-encoded information is recovered. The *Sidekick* PHY is evaluated using the following channel models in MATLAB: `jtcInResC`, `jtcInOffC`, `jtcInComC` that correspond to "Indoor Residential C", "Indoor Office C" and "Indoor Commercial C". We only show the simulation results using `jtcInOffC` as it is similar to the performance of *Sidekick* under other channel models.

B. Results

Fig. 10 shows a contour plot of the probability of correctly detecting the edge of the data transmission, S_D, under different interference and noise energy levels. Observe that the ability of *Sidekick* to accurately locate the edge of a transmission is highly dependent on the interference energy: at an SIR above 6dB, *Sidekick* can determine the edge of a partially-overlapping transmission with over 90% accuracy. Furthermore, there is a sharp change in the edge detection probability: from 0 to 6dB, the probability of accurately finding the edge increases rapidly from 0 to 90%. Also note that the levels of Gaussian noise energy has little impact on edge detection

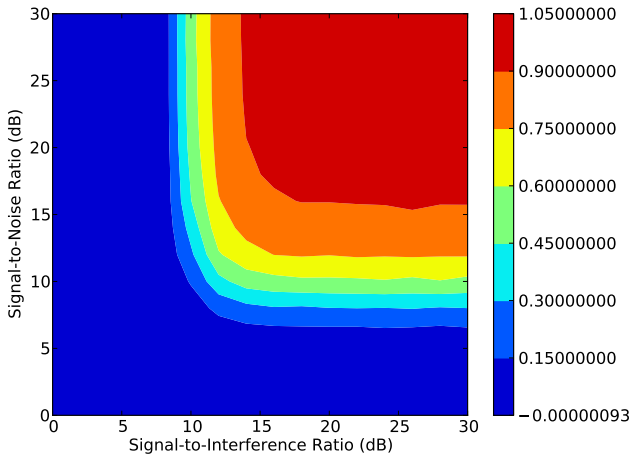


Fig. 11. Probability of correctly decoding the client ID and queue length information in S_D under different interference and noise energy levels

accuracy: for a given SIR level, the edge detection accuracy remains fairly constant over all SNR levels.

Fig. 10 illustrates the edge detection performance with only a single interfering transmission. However, the results shown here are representative of a *lower bound* on detection accuracy. This is because the SIR level at which 90% edge detection accuracy occurs actually *decreases* with increasing numbers of interfering transmissions: by the Law of Large Numbers, as the number of interfering transmissions increases, the statistical properties of the interference approaches that of Gaussian noise, which has very limited impact on the edge detection accuracy of *Sidekick*.

After *Sidekick* detects the edge of a partially-overlapping transmission, it decodes the data encoded in the *Aileron* packet. Fig. 11 shows this decoding accuracy at different interference and noise levels. Observe that a 90% decoding accuracy can be achieved only at SIR and SNR above 14dB and 16dB, respectively. In contrast to the edge detection performance, both the interference and noise energy levels have significant effects on the decoding accuracy. Hence, as long as the client ensures that SIR and SNR on the operating channel are at 14dB and 16dB, respectively, it can be assured that the edge and decoded data can be recovered with at least 90% accuracy.

VII. EVALUATION OF THE SIDEKICK MAC

We implemented the *Sidekick* MAC on ns-2 and evaluated its performance under a myriad of conditions. Table II lists the parameter values used in the simulation evaluation of *Sidekick*. We consider a scenario with multiple APs and one or more clients. Each AP has a single backhaul link that is connected to the Internet.

In this section, we will evaluate the *Sidekick* MAC with schedulers `Sidekick-ILP` and `Sidekick-Greedy`. For brevity, we will refer to these two *Sidekick* configurations as `Sidekick-ILP` and `Sidekick-Greedy` directly. The performance of these *Sidekick* configurations will be compared to that of FatVAP, which is a notable multi-AP aggregation protocol. FatVAP does suffer from an inability to receive

Number of APs	5, 10, 15, 20
Bandwidth of each backhaul link	1Mbps
Bandwidth of wireless channel	54Mbps
Wired backhaul cross traffic rate	0.1 to 1Mbps, in increments of 0.1Mbps
Cross traffic model	Exponential, Pareto
Average On/Off duration	1s On, 2s Off
Simulation duration	250s
Number of repetitions per experiment	10

TABLE II
PARAMETERS USED IN THE EVALUATION OF *Sidekick* MAC IN NS-2

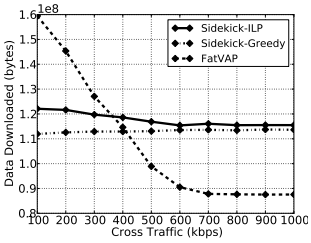


Fig. 12. Total data downloaded by a single client from 10 APs over the 250s simulation run with different cross traffic speeds on the backhaul link.

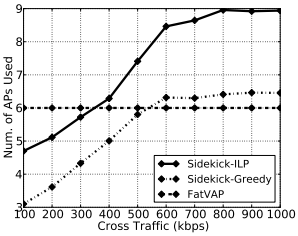


Fig. 13. Mean number of APs active in a connection schedule under different cross traffic rates. A total of APs are present and the channel of *Sidekick* AP partially overlaps with that of exactly one other randomly-selected AP.

out-of-band queue or bandwidth information from candidate APs — it must first connect to an AP before it can measure traffic statistics that are necessary for constructing a connection schedule. We will demonstrate th performance gains that come from the partially-overlapping signaling capability of *Sidekick*.

A. Performance Under Static Conditions

We first evaluate *Sidekick* under static network conditions: all backhaul links, along with the associated cross traffic, are active at the start of the simulation and only one client is present. We compare the ability of *Sidekick* and FatVAP to efficiently select the best subset of APs to use.

Fig. 12 shows the total download by the single client over the entire 250s simulation run. Here, the APs are configured such that each AP has a degree of two: the channel used by each AP overlaps with exactly two other randomly-selected APs. Observe that the total amount of data downloaded by `Sidekick-ILP` and `Sidekick-Greedy` are relatively independent of contending traffic on the backhaul link, with `Sidekick-ILP` outperforming `Sidekick-Greedy` by a margin of less than 10%. FatVAP, on the other hand, outperforms *Sidekick* when the backhaul links are lightly loaded — with the cross traffic throughput is under 400kbps, FatVAP can download up to 45% more data than `Sidekick-Greedy`. However, when the backhaul links are heavily-loaded, both `Sidekick-Greedy` and `Sidekick-ILP` download at least 30% more data than FatVAP.

The reason for this behavior lies in the number of APs that are selected by *Sidekick* and FatVAP as part of the connection schedule. Fig. 13 shows the number of APs that are active in the schedule computed by `Sidekick-ILP`, `Sidekick-Greedy` and FatVAP under varying cross traffic throughput. FatVAP selects its set of APs based on the average wireless and wired throughput measured over a 2-second window. This minimizes the impact of that short term variations, due to the on-off nature of the cross traffic and the bursty nature of typical TCP flows, will have on the resulting schedule. Hence, it maintains a constant set of 6 APs that are active in every connection schedule.

The number of APs used by *Sidekick* does not increase further with increasing overlapping degrees of each AP. Hence, in the rest of this section, we will consider only APs with overlapping degrees of two.

The *Sidekick* client, on the other hand, receives real-time information on the actual length of the queue at each AP and is therefore more sensitive to variations in the queue lengths over short time scales. The burstiness of the packet arrival increases as the throughput of the cross traffic increases and during time quantums in which many APs have short queues, *Sidekick* adapts by increasing the number active APs in its schedules. The use of queue lengths (*Sidekick*) instead of transmission rate (FatVAP) in constructing the connection schedule also ensures that the client will eventually connect to slow APs when the queues on those APs have grown to be sufficiently large. This enables *Sidekick* to maintain its data transfer rate in the face of many low bandwidth backhaul links.

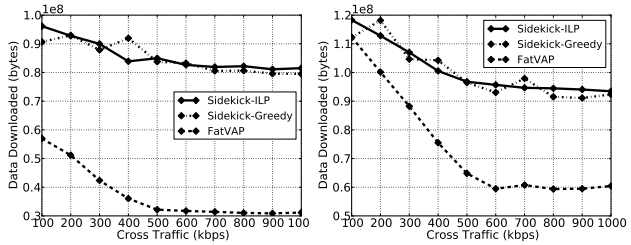
B. Adapting to Significant Bandwidth Changes

Sidekick is able to search for new APs while simultaneously connecting to the currently active set of APs in is connection schedule due to the use of in-band *Aileron* signaling. Here, we evaluate the efficacy of this AP discovery mechanism. This proceeds as follows: we run the ns-2 simulation with a total of 10 APs as before, but only a fraction of these APs are active at the start of the simulation. After 100s, the remaining non-active APs are brought online and begin to advertise bandwidth availability to the *Sidekick* client.

Fig. 14 shows the total data downloaded over the 250s simulation run with different numbers of active APs at the start of the simulation. When only 2 APs are active at the start of the simulation, FatVAP can only achieve a maximum download of 60MB, while both `Sidekick-ILP` and `Sidekick-Greedy` can download at least 80MB in 250s. Similar behavior can be observed when 4 APs are active at the beginning of the simulation. This stark difference in performance between FatVAP and *Sidekick* is due to the fact that *Sidekick* can quickly detect the new APs at the 100s mark and add these APs to the connection schedule; FatVAP, on the other hand, does not probe for additional transmission opportunities and therefore cannot take advantage of the bandwidth offered by the newly active APs. When 6 and 8 APs are active at the beginning of the simulation, FatVAP does achieve its maximum performance as seen earlier in Fig. 12 because it only uses a maximum of 6 APs in its schedule.

C. Performance with Wireless Contention

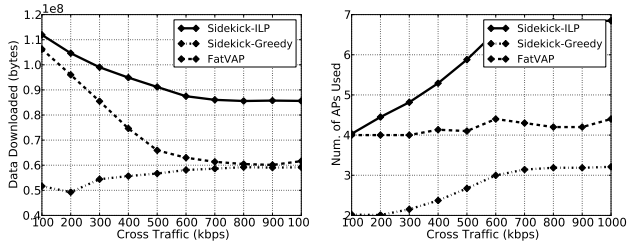
In this section, we evaluate the performance of *Sidekick* in the presence of channel contention from other wireless clients.



(a) 2 APs active (b) 4 APs active

(c) 6 APS active (d) 8 APs active

Fig. 14. Average total data downloaded over 20 simulation runs under different cross traffic throughput and number of active APs at the start of the simulation.



(a) Total data downloaded (b) Mean #APs per schedule

Fig. 15. Total data downloaded under different cross traffic data rates with wireless contention

We model the effect of wireless interference by randomly varying the bandwidth of each channel from an AP to the client between 22 and 54Mbps. This setup succinctly captures the effects of channel interference from both WLAN nodes and other noise sources while enabling us to focus on the behavior of *Sidekick*. The bandwidth of each wireless channel is fixed at the start of the simulation and each experiment is repeated 20 times. We run the simulation with 10 APs so that both *Sidekick* and FatVAP will not be constrained by the available backhaul bandwidth; all APs are active at the start of the simulation.

Fig. 15 shows the performance of *Sidekick* and FatVAP in this scenario. Observe that `Sidekick-ILP` outperforms both `Sidekick-Greedy` and FatVAP. The improvement of `Sidekick-ILP` over FatVAP comes from its access to real-time information on the queue length and wireless rate. `Sidekick-Greedy`, on the other hand, only takes the queue length information into account and hence cannot determine the optimal order of APs in its connection schedule when faced with wireless links of significantly varying throughput. This is also evident by the fact that `Sidekick-Greedy` uses significantly fewer APs in its connection schedule, as compared to `Sidekick-ILP` and FatVAP — `Sidekick-Greedy` often gets "stuck" on APs with long queue sizes and low wireless throughput.

VIII. Conclusion

In this paper, we presented *Sidekick*—a novel protocol that utilizes efficient signaling across partially overlapping channels to enhance the achievable aggregate throughput over multiple APs. *Sidekick* uses *Aileron* to send queue-length information from the AP to each client concurrently with parallel with normal data transmissions. The availability of real-time queue-length information enables each client to achieve up to 30% throughput gain over FatVAP and improve its ability to rapidly exploit new transmission opportunities via the introduction of new APs.

Acknowledgement

The work reported in this paper was supported in part by the NSF under Grant No. CNS-1114837. Special thanks to our shepherd, Sujata Banerjee.

References

[1] "Fon." Website. http://corp.fon.com/en.
[2] "Meraki." Website. http://meraki.com.
[3] T. Dasilva, K. Eustice, and P. Reiher, "Johnny Appleseed: wardriving to reduce interference in chaotic wireless deployments," in *MSWiM*, 2008.
[4] V. Shrivastava, S. Rayanchu, J. Yoonj, and S. Banerjee, "802.11N Under the Microscope," *IMC*, 2008.
[5] S. Kandula, K. Lin, T. Badirkhanli, and D. Katabi, "FatVAP: aggregating AP backhaul capacity to maximize throughput," in *NSDI*, 2008.
[6] D. Giustiniano, E. Goma, A. Lopez Toledo, I. Dangerfield, J. Morillo, and P. Rodriguez, "Fair WLAN backhaul aggregation," in *Mobicom*, 2010.
[7] L. Yang, W. Hou, L. Cao, B. Zhao, and H. Zheng, "Supporting Demanding Wireless Applications with Frequency-agile Radios," *Proc. of NSDI*, 2010.
[8] K. Tan, J. Fang, Y. Zhang, S. Chen, L. Shi, and J. Zhang, "Fine-grained channel access in wireless LAN," in *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, pp. 147–158, ACM, 2010.
[9] H. Rahul, N. Kushman, D. Katabi, C. Sodini, and F. Edalat, "Learning to share: narrowband-friendly wideband networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 147–158, 2008.
[10] S. Kim, X. Wang, H. Kim, T. Kwon, and Y. Choi, "CRAWDAD trace snu/bittorrent/tcpdump/static (v. 2011-01-25)." Downloaded from http://crawdad.cs.dartmouth.edu/snu/bittorrent/tcpdump/static, Jan. 2011.
[11] E. Chai and K. Shin, "Low Overhead Control Channels in Wireless Networks," Tech. Rep. CSE-TR-574-11, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, 2011.
[12] A. Mishra, V. Shrivastava, and S. Banerjee, "Partially overlapped channels not considered harmful," *SIGMETRICS*, 2006.
[13] R. Chandra, P. Bahl, and P. Bahl, "MultiNet: connecting to multiple IEEE 802.11 networks using a single wireless card," in *INFOCOM*, 2004.
[14] a.J. Nicholson, S. Wolchok, and B. Noble, "Juggler: Virtual networks for fun and profit," *IEEE Transactions on Mobile Computing*, vol. 9, Jan. 2009.
[15] X. Xing, S. Mishra, and X. Liu, "ARBOR: hang together rather than hang separately in 802.11 wifi networks," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, IEEE, Mar. 2010.
[16] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine, and J. Zahorjan, "Interactive wifi connectivity for moving vehicles," in *SIGCOMM*, 2008.
[17] P. Gilbert, E. Cuervo, and L. P. Cox, "Experimenting in mobile social contexts using JellyNets," in *HotMobile*, 2009.
[18] L. E. Li, K. Tan, H. Viswanathan, Y. Xu, and Y. R. Yang, "Retransmission $\neq$ repeat: simple retransmission permutation can resolve overlapping channel collisions," in *Mobicom*, 2010.
[19] K. Chebrolu and A. Dhekne, "Esense: communication through energy sensing," in *Mobicom*, 2009.