IdleChat: Enabling High Bandwidth Real-time Applications in Residential Broadband Networks

Ramya Raghavendra
IBM Research, Hawthorne

Michael Kaminsky
Intel Labs Pittsburgh

Konstantina Papagiannaki
Telefonica Research

Srinivasan Seshan
Carnegie Mellon University

Elizabeth Belding
University of California Santa Barbara

Residential broadband links are characterized by low and variable upload capacity and large latencies, leading to poor video performance. Aggregating multiple backhaul links using 802.11 Access Points has been considered as a solution to increasing the backhaul capacity limit. In this paper, we revisit the problem of backhaul aggregation in the context of interactive applications. We present the IdleChat architecture that uses wireless connectivity to aggregate idle broadband connectivity in a neighborhood to overcome the limits of individual asymmetric broadband uplinks. In contrast to prior work, IdleChat aggregates in a traffic-aware manner, focussing specifically on low-latency, real-time link aggregation in support of interactive applications such as video chat. IdleChat schedules packets in a deadline-aware manner to improve interactivity. We evaluate IdleChat both in an experimental testbed in our lab and in a residential deployment. Our results show that use of IdleChat in existing deployments can improve video SNR by up to a factor of 10 db in the presence of a single neighboring AP. This results in video quality improvement from a MOS score of bad to excellent.

I. Introduction

The popularity of social networking and chat applications have resulted in an increase in the amount of data that users are generating and uploading to the web, in contrast to web traffic that is predominantly in the downlink direction. The delivery of multimedia content is growing at a tremendous rate, increasing 76% every year on average, with Video communication and real-time traffic growth predicted to increase tenfold by the year 2013 [1].

Residential broadband networks that provide Internet connectivity to residences and hotspots in urban environments are characterized by low upload speeds, insufficient to sustain high bandwidth video applications. Real-time video applications that cannot take advantage of techniques such as buffering are forced to operate at low bandwidth and consequently result in poor user experience.

In urban environments, residential users are often in range of multiple APs [2] connected to broadband links that are predominantly idle [3]. The 802.11 link speeds are typically an order of magnitude higher than the broadband connections. As a result, it is possible to achieve higher backhaul speeds by aggregating multiple low-speed backhaul connections using the high-speed wireless links. That is, a single wireless card can be used to switch between APs in range in order to utilize each of their backhaul capacity.

Several recent trends indicate that ISPs and vendors favor sharing of broadband connections in neighborhoods. Community efforts such as FON [4], Wi-Sh [5] and Meraki [6] indicate that users are willing to share their wired and wireless connectivity in the presence of appropriate incentives (e.g., for a monetary reward). There exist instances of telco-managed sharing schemes where residential WiFi gateways are shared across all users that subscribe to the service. Internet Service Providers such as British Telecom (BT) and Telefónica are leaning in this direction with efforts such as the BT Home Hub [7] and THEMIS [8]. The BT Home Hub promotes wireless community networking and sharing of wired connectivity by supporting two simultaneous SSIDs, one of which forms part of the FON network. THEMIS is a recent effort by a large ISP to provide bandwidth aggregation in a fair and distributed manner while bypassing the backhaul capacity limit, and is being evaluated in a commercial trial by the ISP. Finally, a number of manufacturers now offer affordable, open-firmware wireless routers (e.g., FON, OpenMesh Linksys, and Netgear [9, 10, 11]), which enables easy deployment of new functionality. Similarly, manufacturers like Broadband Bonding [12] and Wi- Boost [13] build custom hardware for the bonding of wired links through wireless connections.

Prior work [14, 15] has looked at aggregating bandwidth from multiple access points for faster file uploads and downloads. Using an aggregation scheme designed for file transfers to transmit interactive traffic would lead to severe degradation in video quality. As opposed to reliability, a real-time video transmission scheme should attempt to maximize the number of video frames that reach the destination before their deadlines elapse. Specifically in the case of real-time video, where the playout buffering is limited, video packets have *stringent playback* constraints and hence there is a need for aggregation schemes that addresses timely delivery.

When multiple APs, each connected to its own backhaul link, are available for connection, we need to address the problem of *AP Selection*. Prior schemes [14] are designed such that stations greedily maximize their individual throughput. However, wireless and backhaul links have widely varying packet loss, latency and jitter properties that affect video quality and hence need to be taken into account when selecting APs.

Video traffic lacks flow semantics of TCP, so the problem of *traffic division* across APs must be addressed while scheduling traffic over multiple links. Video encoding creates video frames with varying priorities and loss of a single frame can affect video quality for a perceivable length of time. Prior aggregation schemes targeting TCP transfers are traffic-agnostic, i.e. all packets have the same likelihood of being transmitted. A cross-layer video transmission scheme would stand to gain from scheduling packets in a content-aware manner.

To address the above challenges, we design and implement IdleChat, a single-radio system that exploits wireless connectivity to aggregate the bandwidth of multiple broadband uplinks. Our system then makes this aggregate bandwidth available to users for high-resolution real-time video communication. To do so, IdleChat periodically measures the wireless and backhaul link properties and selects APs such that the video performance is improved. The client toggles between these selected APs and transmits packets in a traffic-aware manner. The key contributions of our system are:

- We design IdleChat, an end-to-end link aggregation scheme for high bitrate real-time video traffic streaming on low capacity broadband links. The design incorporates various aspects of link quality estimation, access point selection, deadline-driven packet scheduling and adaptive retransmissions. The implementation works with existing 802.11 deployments, requires a single wireless card with no access to firmware and needs no modification at the APs.
- Through evaluations on a testbed and in a residential deployment, we show that IdleChat is capable of aggregating links to improve video quality. In a presence of a single neighboring AP, we increase the video streaming resolution from videos encoded in standard definition with 480p resolution (704×480) to 720p resolution (1280×720), while maintaining an “excellent” video quality.
- IdleChat clients coexist with other clients. We show that IdleChat share the medium fairly with other unmodified clients running TCP or UDP traffic, as well as with other IdleChat clients.

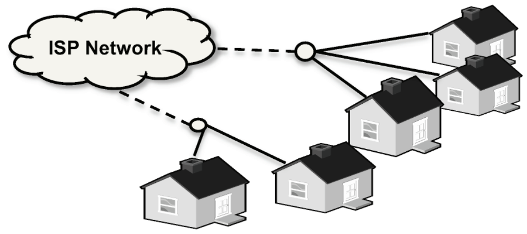


Figure 1: Architecture of typical residential access networks. The ISP core essentially splits into several regional headends (or DSLAMs in the case of a DSL network) and each headend connects a set of homes.

II. Background and Motivation

Residential networks comprise of a broadband link (predominantly cable or DSL) providing Internet connectivity, with an 802.11a/b/g/n AP extending connectivity within the house. Residential broadband networks provide residences with last mile Internet access. A large and rapidly growing percentage of residential Internet connections are via broadband DSL or cable modem technologies. Figure 1 shows the typical architecture of residential broadband networks. In both cable and DSL networks, the core branches out into a number of central offices that support a cluster of geographically proximal customers. Despite having different physical layer technologies, the ISP core essentially splits into several regional headends (or DSLAMs in the case of a DSL network) and each headend connects a set of homes. A key difference between cable and DSL access links is that DSL customers have a dedicated connection to the regional center, whereas cable users share the connection to the headend.

Broadband networks are also known to have a large disparity in the upload and download speeds. In particular, the measurement study by Dischinger, et al. [16] shows that upstream bandwidths were less than 500 Kbps, even in cases where the downstream bandwidths exceeded 5 Mbps in the 600 residences that were measured. The ratio of downstream to upstream traffic is shown to be high, over a factor of 10 in the case of cable hosts.

Usage studies of broadband links, however, show that these links are typically used only a fraction of the time [17]. The per-user rate limiting of broadband networks ensures that users get a fair share of broadband access when the network usage is high, but when the network is not being used to its capacity, leaves a significant amount of bandwidth on the table.

In residential networks, a common method of accessing broadband networks is using the Wi-Fi network through the wireless access point (AP) deployed in a house. In this paper, we study the video streaming quality over the broadband network, as well as the wireless link on the first hop. Numerous past studies have shown the dense deployment of 802.11 APs in residential areas [18, 2]. Wireless signals often reach across homes, making it possible to sustain TCP and UDP connections with open APs in the neighborhood [2, 19].

We conduct an experimental study to measure the video streaming performance in a typical residential neighborhood. The setup consisted of a laptop that was connected to the broadband link through an 802.11b/g AP. The laptop then streamed the video to a client that was connected to a high-speed academic network. The experiment was repeated in five different neighborhoods across three towns and was a mix of apartments and houses with either cable and DSL connections. Results from 1000 streaming sessions of a 3 Mbps bitrate video are shown in Figure 2. The x-axis is ordered in ascending order of video PSNR. We use PSNR (Peak Signal-to-Noise Ratio) as the metric for video quality estimation. PSNR is the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Video PSNR is well correlated with the perceived video quality. The relationship between user perception and PSNR is indicated in the figure. As shown in Figure 2, over 90% of the sessions experienced *poor* to *fair* video quality.

The challenges in streaming video traffic in residential networks stem from both the nature of video traffic as well the properties of the broadband and wireless networks that are used to access the Internet from residences. We outline the important properties and challenges that need to be addressed in the design of a real-time high-resolution video delivery system. We outline the most relevant prior work to address these challenges and discuss how our system differs from them.

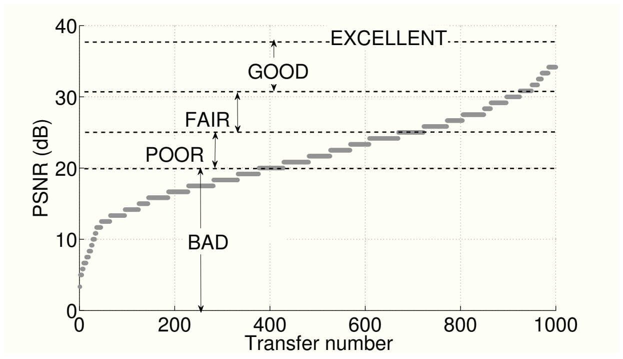


Figure 2: PSNR values of 1000 video streaming sessions from 8 residences. The sessions are arranged in ascending order of their PSNR value for visual clarity. The mapping between PSNR range and MOS based user perceptions are indicated.

II.A. Broadband Networks

Measurement studies that have characterized the properties of broadband access networks [20, 16, 21, 22] show that there are several challenges posed by these networks for streaming real-time traffic. First, the last mile access networks constitute the bottleneck in Internet access [20]. Such bottlenecks severely impact the performance of multimedia traffic, particularly of applications such as video chat that requires equal upload and download bitrates. Second, broadband networks have shown to have a large disparity in the upload and download speeds. The measurement study by Dischinger, et al. [16] showed that the ratio of downstream to upstream traffic was high, over a factor of 10 in the case of cable hosts. Third, the bandwidth available over broadband networks tends to be unstable, even over short periods of time [21, 22]. We address each of these challenges in the design of our system.

While on one hand the upload bandwidth available is typically low, usage studies of broadband links however show that they are used only a fraction of the time [3]. The per-user rate limiting of broadband networks ensures that users get a fair share of broadband access when the network usage is high, but when the network is not being used to its capacity, leaves idle bandwidth on the table.

II.B. Wireless Networks

Numerous past studies have shown the dense deployment of 802.11 Access Points (APs) in residential areas [23, 2]. Wireless signals often reach across homes, making it possible to sustain TCP and UDP connections with open APs in the neighborhood [19, 2] presenting an opportunity for using wireless for backhaul aggregation. However, the wireless links can be prone to losses and interference from neighboring networks [20, 22].

In the following sections, we describe the design and evaluation of IdleChat, which addresses the challenges described above to build a system capable of delivering high bitrate video streams within real-time constraints. IdleChat uses the local wireless connectivity in order to aggregate broadband links, while ensuring that the demands of the video traffic can be met.

II.C. Network Connection Sharing:

Several schemes have been proposed to aggregate multiple broadband links using wireless to increase upload or download speeds [14, 24, 15, 8]. FatVAP [14] is an 802.11 driver design that aggregates the bandwidth available at nearby APs and load balances traffic across them. FatVAP incorporates an AP scheduler that maximizes client throughput based on load on each AP, a load balancer that maps traffic to APs according to their available bandwidth and an AP switching mechanism that is fast and loss-free. FatVAP is designed for fast TCP based file transfers. ARBOR [24] is a solution recently proposed to add security mechanisms over systems such as FatVAP. As described in detail in § I and § III, the key difference between our scheme and these is in the class of applications for which we target our solution. While these solutions are designed specifically for file transfers, we study aggregation challenges for real-time video streaming applications. We extend the mechanisms of AP selection and traffic division, and add traffic-awareness to the aggregator.

At least two commercial products also offer network connection sharing: *Mushroom Networks* and *Wi-Boost*[1]. Based on the limited literature on their products, it appears that they also focus on flow load balancing and unicast distribution for TCP file transfers.

II.D. Video Traffic

Video traffic imposes stringent demands on the bandwidth and latency of the network. The bandwidth consumed by the video traffic varies with the video resolution and the video encoding used. Raw HD video can have a bit rate ranging from a few tens to few hundred Mbps. As an example, a common high definition resolution of 720p ($1280x720$ pixels) with a frame rate of 24 per second in its raw format with YUV 4:2:2 color model (common for video) has an uncompressed bit rate of over 44 MBps. Commercially available HD videos from Comcast DOCSIS 3.0, on-demand movies from Apple TV 2.0, HD downloads from XBOX360 are highly compressed using compression codecs such as H.264 or VC-1 and have a bit rate in the range of 1.5 Mbps - 7 Mbps.

Real-time video poses another challenge of bounded end-to-end delay. In real-time video streaming, video frames are played as they are received and packet delay variation, or jitter, degrades the perceptual quality. A video packet that arrives beyond its playback deadline is useless and can be considered lost. In the case of real-time video, the recommended total end-to-end delay is 200ms [2].

There has been a significant amount of research in the video and systems community on challenges in streaming video over Internet as well as wireless networks, [25] provides a summary. Dynamic transcoding [26, 27] is a technique used to enhance the quality of streaming video. This body of work looks at estimating the channel quality and choosing a video rate to suit the bandwidth available. Our work is complimentary to these schemes in that we rely on the ability of video transcoding, and provide the network support to detect path characteristics. Selective error protection using FEC [28, 29] and multiple description and layered coding schemes [30] are examples of schemes that improve video delivery over lossy links. We find this work complimentary to our schemes. Given that APs provide multiple paths for the video packets, and have varying bandwidths, we believe that the problem of traffic division and bandwidth estimation need to be addressed for these complimentary schemes to be useful.

III. Design Overview

The overall design of the IdleChat video delivery system is shown in Figure 3. At a high level, IdleChat bonds neighboring broadband links by connecting to the APs within range with an end goal of achieving high resolution real-time video communication. By bonding multiple APs that have spare capacity, we can increase the effective uplink capacity, enabling high bandwidth video transfer.

[1]www.broadbandbonding.com and www.wiboost.com.

[2]www.ieee802.org/802_tutorials

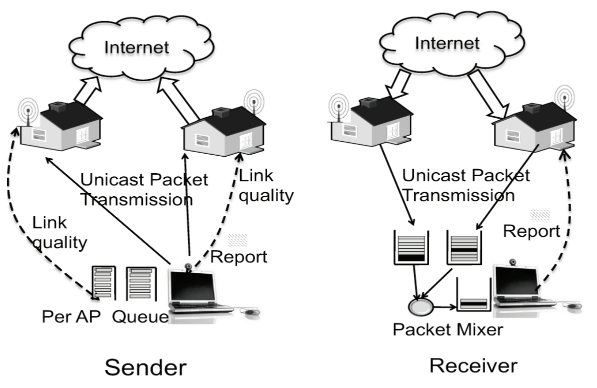


Figure 3: Schematic representation of the IdleChat video delivery system.

We describe the design and evaluation of IdleChat for MPEG-4 encoded videos, which is the state-of-the-art for video compression, especially for video delivery over the Internet. A *Group of Pictures* (GoP) is an independently decodable sequence of frames. All frames in a GoP are not created equally. An MPEG-4 frame can be encoded in one of three ways: I,P and B frames. A GoP comprises of an I frame followed by a sequence of P and B frames. An I frame is encoded without reference to any other frame of video. P frames are coded predictively from the nearest reference frame (either I or P) and B frames are coded bidirectionally from preceding and succeeding reference frames. For a P frame to be successfully decoded, the previous reference frame must be received and for a B frame,both the previous and the next reference frame must be received. A lost I frame means that the rest of the frames in the GoP must be discarded too.

IdleChat involves an unchanged media server that uses RTP as an application layer streaming protocol. In the design of our system, we have kept the APs unmodified, and have added all the intelligence at the end hosts. The node generating video traffic (the sender) selects a set of APs to connect to based on the bandwidth it can obtain from each AP, and decides how much traffic it will forward to each AP based on this available bandwidth. During transmission, the sender transmits packets in a content-aware fashion such that high priority packets have a greater chance of being received. The receiver downloads the video substreams and reassembles them into a single video stream that can be played back. In order to provide feedback to the sender to facilitate adaptation, the clients are modified to send periodic reports on packet delivery rates, delays and losses observed along each path that the sender used to forward traffic. The problem of video packet transmission over bonded links can be formulated as: *For a set of video packets, determine how many and what packets to send to each AP such that as many of them as possible arrive at the destination before the packet deadline expires.*

In the rest of this section, we present our solution to each component of this problem and describe how the solutions can be put together to build a real-time video streaming solution. While we describe our scheme for MPEG4 video, our approach generalizes to any other media encoding where the content is structured into layers and there are different levels of priority for each layer.

III.A. Access Point Selection

Given a set of APs in the neighborhood to which a client can connect, we need to decide to which APs to connect and for how long we need to connect to each of them. The objective of the scheduler is to find the subset of APs that can collectively provide the bandwidth required for the video, and ensure that the packets reach the destination meeting the real-time constraints. While the objective of the problem is similar to the AP scheduling problems in FatVAP [14] and ARBOR [24], we address this in the context of video applications. There are several critical differences between IdleChat and the solutions proposed in prior work:

- We build a link aggregator for multimedia traffic that predominantly uses UDP as the transport layer, and does not have flow semantics that can be used for traffic division or aggregation. This requires us to redesign traffic division at the sender and reassembly at the receiver.
- Since video traffic is highly sensitive to bandwidth fluctuations, our system should not be aggressive in utilizing all the available bandwidth, but leave spare capacity to account for traffic and link quality fluctuations.
- Packet delays in broadband networks can be very large (as much as 1s), thus rendering interactive video streaming infeasible. We explicitly account for the end-to-end delay and packet jitter in several ways. Our scheduler prunes APs whose estimated delay is over the maximum delay tolerable by real-time applications. We also monitor for changes in end-to-end delay and jitter, and reschedule APs when delay increases above tolerable threshold.

The steps involved in AP selection and creating a AP scheduling are outlined below.

III.A.1. Estimating Bandwidth

Multimedia streaming is sensitive to bandwidth fluctuations, and as a result streaming quality depends to a large extent on correct bandwidth estimation. Both media scaling techniques and client side buffering rely on the bandwidth estimate of the underlying network.

Most available bandwidth estimation techniques are designed to provide accurate bandwidth information for wired networks at the cost of long convergence times and high intrusiveness [31]. These tools assume that the sender transmits packets at a consistent size and rate; however, a general multimedia application is unlikely to have these two properties. Further, the granularity of bandwidth reported, as well as the frequency at which the measurements are updated, are too high to be useful for multimedia traffic.

End-to-end Bandwidth: We define the end-to-end bandwidth through an AP as the rate at which we can sustain the video flow with a receiver through that AP. There are two phases in bandwidth estimation: a) *Estimation* phase that computes the available bandwidth initially and b) *Verification* phase that ensures that the AP continues to have the same bandwidth that was initially estimated.

In the bandwidth estimation phase, we use the receiver-based packet probe method (RBPP) that uses two consecutively sent packets to determine the bandwidth share based on the delay between the two packets [32]. If $t1$ and $t2$ are the arrival times of two packets sent back to back and $s2$ is the size of the second packet, then the receiver estimates the bandwidth to itself as:

$$\mathcal{B} = \frac{s2}{t2 - t1} \tag{1}$$

Measurements present one caveat in this method. Because of access variabilities due to queuing on the intermediate links, packets could have long one way delays, but still arrive at the destination back-to-back. In order to not overestimate the available bandwidth, we discard measurements where the packets are time compressed, i.e, the inter-arrival times of packet reception is lower than the sending time. At the sender, the driver hands a packet to the HAL where it is queued for transmission. Without access to the HAL, the driver does not know when a packet was transmitted. Instead we look at the timestamps on the ACK packets (t_a). The HAL timestamps each packet with the time it was ACKed, and the send time ts of two packets $p1$ and $p2$ is given by $ts = ta_{p2} - ta_{p1}$ In order to provide buffer capacity to account for variability in video bitrate and in broadband links, we schedule traffic for 80% of the estimated bandwidth.

In the bandwidth verification phase, we use the inter-packet delay variation between the video packets at the receiver to estimate changes in available bandwidth. The receiver computes the delay trends over the previous 100ms period. We use a weighted moving average to detect trends in delays for the 100ms window, given by:

$$D = D_{k-1} * (1 - \alpha) + D_k * \alpha, k \geq 0 \tag{2}$$

There is one subtle issue posed by using this method with video traffic. Because the packet sizes are variable, an increasing trend in delay could either be due to an increase in network delays or increase in packet sizes. To account for increasing packet sizes in variable bit rate video coding schemes, we use the same method to observe the trends in packet sizes. We infer that the send rate has exceeded the available bandwidth if the packet size distribution is not showing an increasing trend, but the one-way delay variations are increasing.

Wireless Bandwidth: The client measures the wireless bandwidth available to each of the AP that it can connect. Wireless bandwidth varies depending on the modulation rate at which the clients can communicate with the AP and the amount of contention in the medium. The client estimates the bandwidth by measuring the time between a packet was transmitted to the time the packet was acknowledged. This time includes the time taken in contention, transmission, retransmission and ACK. The wireless bandwidth w_i is calculated by dividing the packet size B by the transmission time of the packet t_p.

III.A.2. AP Ranking

The AP ranking phase sorts the APs according to their link qualities, such that APs with a higher chance of transmitting packets on time are picked with a higher priority. We avoid connecting to APs with delays greater than the tolerable threshold for interactive video traffic.

The AP ranking phase then ranks APs in the order of the bandwidth available and then their expected packet latencies. The expected packet latency associated with an AP is the sum of the delay and jitter measured through that AP. The delay measurements are described below, and jitter measurements are described in § III.A.1. To account for delay and jitter variations, we associate an AP with the 80^{th} percentile of its delay and jitter value. The APs ranked in the order of packet latencies along with information about the bandwidth available is then used to schedule the APs.

Estimating Packet Latency: Initially the sender sends a short stream of ICMP probes to the destination through each AP, receives the ICMP responses and uses the RTT values to compute delays. APs that have paths to the destination but with large delays are excluded from the AP schedule. Once we are connected to the AP, we need to continue to monitor the link delays. The receiver computes one way delays using sender timestamps (t_s) on the video packets and the time at which the packet was received (t_r). The receiver computes one way delay $\delta = t_r - ts$. We then observe whether the delay values are increasing. The sender and the receiver clocks are not synchronized, so this does not give us the exact value of delay. But assuming there is no clock drift at the sender, this gives us information about the trends on how delays are changing over time. The receiver reports the computed delay values back to the sender, which the sender uses to decide whether the delay on an AP has exceeded the threshold. If the delay has exceeded, the AP is removed from the schedule, and is probed at a later stage to check whether the delay value has decreased.

III.A.3. AP Scheduler

The goal of the scheduler is to find the subset of APs to toggle between such that: a) it can achieve the aggregate bandwidth taking into account the overhead of switching between APs, and b) ensuring low latency paths for packets. The AP scheduler simply takes the APs ranked in the order of their bandwidths and packet delays tries to find a schedule such that the aggregate bandwidth is optimized. If multiple APs with comparable bandwidths are available, we always pick the one with the least expected delay.

We formalize the scheduling problem as follows. Let e_i be the end-to-end bandwidth that can be obtained by connecting to an AP_i. Since the end-to-end bandwidth is determined by the bandwidth availability on the wireless (w_i) and backhaul (b_i) links, e_i is given by $e_i = min(w_i, b_i)$. The schedule is computed such that the total duty cycle is D, i.e., the total time to toggle between all the scheduled APs. The scheduler needs to compute the fraction f_i of D that a client connects to AP_i to collect its available bandwidth plus the cost of switching overhead. δ_i be the latency through AP_i.

The objective of the scheduler is to maximize the client throughput while keeping the latency within the allowable limit. Hence, the constraints to the scheduler are that the total time should be under D and the latency should not exceed the maximum allowable latency δ, i.e.,

$$\max_{fi} \sum_i e_i \tag{3}$$

$$s.t.\ f_i D + s \leq D \tag{4}$$

$$\delta_i \leq \delta \tag{5}$$

This optimization problem is similar to the unbounded knapsack problem, and we solve it using greedy approximation method. The APs are sorted in decreasing order of value per unit cost and the scheduler picks as many APs as it can fit within the duty cycle D while adhering to the delay constraint.

Given the size of our playout buffer of 200ms, we set a conservative duty cycle value of 100ms which leaves room for fluctuations in delay that is often seen in wireless and broadband links, yet is long enough to benefit from toggling between APs. The scheduler is run initially while deciding the set of APs to connect. After that, the scheduler is run when the bandwidth is reported to have dropped by the bandwidth verification phase.

If the scheduler cannot schedule APs to the desired bit rate because there is not enough capacity available, it can notify the application to scale down its bitrate. In our implementation, we mimic this behavior by encoding videos at different bitrates. The driver estimates the amount of bandwidth available and the streaming application starts to stream the video with the appropriate bitrate.

III.B. Adaptive Retransmissions

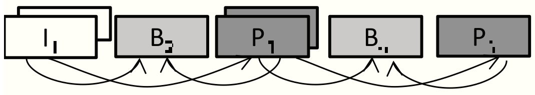


Figure 4: Dependencies between frames in Group of Packets (GoP). Each frame can be comprised of multiple packets.

In an unmodified 802.11 system, the link layer retransmits a packet if a transmission attempt is not successful. In Atheros cards, the retry limit for data packets is set to 7. Since different video packets have different levels of utility, adapting the retransmit limit in a content-aware manner gives two benefits: a) when packets are backlogged for transmission, a low-priority packet that is waiting for retransmission and causing head-of-line blocking of high-priority packet queued behind it can be dropped potentially without affecting video quality, and b) a packet that is certainly going to miss its deadline due to retransmissions can have its retry limit decreased to prevent transmitting a dead packet and wasting network resources.

Prior solutions on prioritizing video frames to minimize rate distortion at the receiver do so by modeling the packet dependencies as a directed acyclic graph, and then using LP based techniques to assign priorities [29]. While these techniques are useful, we propose a simpler technique that does not add computational overhead or require buffering packets at the source to observe packet dependencies.

Consider a video sequence with GoP size α and inter-frame interval λ. Without losing generality, we assume α and λ are fixed and known a priori. The frames dependency shown in Figure 4 can be expressed by
$S = \{F_{1,1}F_{1,2}F_{1,3}\ldots F_{1,\alpha}F_{2,2}\ldots F_{i,j}\ldots\}$
where $F_{i,j}$ denotes the j^{th} frame within the i^{th} GOP. Frame $F_{i,j}$ is composed of video packets with $P_{i,j}^{(k)}$ denoting the k^{th} video packet in $F_{i,j}$. For each $P_{i,j}^{(k)}$, we compute the utility of a packet as

$$U(P_{i,j}^{(k)}) = M(F_{i,j}) + 1 \qquad (6)$$

where M($F_{i,j}$) is the number of frames inter-coded with respect to $F_{i,j}$. For simplicity, all video packets corresponding to the same frame are assigned the same U. For example, in Figure 4, for the first I-frame we assign 5λ as its packet utility because 4 frames are directly or indirectly inter-coded with respect to it, and we add one for the frame itself. By associating a reference frame (and all its composing video packets) with a larger U, Equation 6 represents the relative importance of each packet in terms of its error propagation properties.

In order to adapt the retransmission count of a packet, we consider the packet utility, as well as its deadline. For now, lets assume that the packet deadline $d'_{i,j}$ is known, after which the packet is not useful at the receiver. The estimation of this deadline is described in § III.B.1. Since we know the expected delay to the AP, we compute $n(P_{i,j}^{(k)})$, which is an estimate of the number of times a packet can be retransmitted such that it reaches the receiver before the deadline expires. The retransmission limit is adapted to [3]

$$R(P_{i,j}^{(k)}) = min(U(P_{i,j}^{(k)}), n(P_{i,j}^{(k)})) \qquad (7)$$

In wireless NICs, reception of an ACK confirms the successful delivery of a packet. If a packet is not followed by an ACK, the NIC attempts to retransmit the packet based on the limit set for the packet. If transmission is not successful after the preset number of attempts, the packet is discarded. The higher layers are expected to attempt retransmission if reliability is desired. We modified the driver to save a packet for the duration of $RTT + \delta$ (where RTT is the wireless round trip time). If we do not receive an ACK, we put the packet back in the queue and schedule it for a retransmission *only if* it can be transmitted before the deadline.

[3]Note that U, n, R are all unitless numbers.

III.B.1. Deadline estimation

The deadline of a frame is estimated as:

$$d = \text{Playout time} + \frac{(|GoP| - 1)}{\lambda} + \Delta + \epsilon$$

Playout time is the instance at which the first packet was received for playback. This information is sent by the receiver with the first report. $|GoP|$ is the index of the GoP that the current frame belongs to. Δ is the playout buffer at the receiver and ϵ is added to account for overhead incurred due to switching and related operations.

III.C. Online Packet Scheduling

The problem of scheduling packets with different utilities and deadlines, among a given set of links such that the total value is maximized, is known to be a NP-hard problem [33]. We consider the case of scheduling packets online, i.e., a packet is scheduled for transmission at the instance it arrives, or discarded. We do not design an offline scheduling scheme, where in a window of packets is observed before making a scheduling decision, in order to avoid packet buffering at the source. The additional delay introduced by buffering does not outweigh the benefits of packet scheduling in a real-time system.

The outline of the scheduling algorithm is shown in Algorithm 1. The scheduler ties together the estimated bandwidth $\mathcal{B}$, the packet deadline d and delay δ and makes a decision on packet transmission. Each packet is associated with a deadline, which is the playout time of the packet. We assume a model wherein a packet that reaches the destination after its deadline is of no use. A packet that arrives by its deadline contributes a certain "value" equal to its utility. Each AP provides the packet with a different path to its destination, and each path is associated with an average delay. Given this model, our objective is to maximize the aggregate value achieved by packets reaching the destination. Toward this objective, we implement a packet scheduler that schedules outgoing packets on the AP such that the utility is maximized.

Let p_k be the packet that is being scheduled on queue Q_i which has a bandwidth $\mathcal{B}_i$ and delay δ_i. Let t be the instant at which packets are being scheduled and d_k be the time at which a packet p_k is due at the receiver. The sender needs to decide whether to a) transmit on the current AP or switch to the next AP in the schedule based on the counter $\mathcal{C}$ that maintains the available bandwidth on each per-AP queue, or b) drop packet. In this way packets that are unlikely to reach the destination before the deadline are not scheduled for transmission, thus avoiding wasting resources on the wireless

Algorithm 1 Packet scheduling algorithm

```
Require: $p_k, d_k, \delta, \mathcal{B}, \mathcal{C}$
 1: $Q_i = \phi$
 2: $i = 1$
 3: if $(t + \delta_i) > d_k$ then
 4:     Packet to be dropped
 5:     continue
 6: else if $|p_k| \geq \mathcal{C}_i$ then
 7:     $i = i + 1$
 8:     continue;
 9: else
10:     Add packet $p_k$ to $Q_i$
11:     $\mathcal{C}_i = \mathcal{C}_i + |p_k|$
12: end if
13: return Q
```

or the broadband access link. While scheduling these packets among m queues (each queue representing an AP), packets are scheduled until each queue is full, i.e. the bandwidth of the AP has been reached. Scheduling a packet on a queue means that the packet will be added on to the per-AP queue maintained in the driver, and is handed down to the NIC for transmission immediately.

IV. Evaluation

We evaluate our implementation of IdleChat through extensive tests in controlled testbed settings and in a residential area with cable and DSL broadband uplinks. Our setup consisted of the following:

Drivers: We compare three drivers: a) *Unmodified madwifi* driver v0.9.4[4]; (b) *BWIDTH-AGG*: This is our implementation of a content-agnostic bandwidth aggregator that bonds broadband links for UDP flows. This driver mimics the behavior of prior bandwidth aggregation schemes [14, 8]; and (c) *IdleChat Driver:* This includes our extensions to madwifi v0.9.4, along with the channel switching capabilities provided by the MIT implementation of FatVAP [14]. The mean switching overhead of the driver is 3000μs irrespective of which channel the APs are transmitting on. Our implementation extends madwifi to include features described in Section III.

Hardware: Our residential experiments involve a variety of commercial APs in 802.11b/g mode. Our testbed uses unmodified Linksys APs. The APs transmit on 802.11g channels. The testbed comprises of commercially available Linksys WRT54G APs, and can buffer up to 200 KB for a client that enters the power-save mode. Testbed APs are assigned different 802.11g channels. The wireless throughput to all APs in our testbed is in the range $[19 - 24]$ Mb/s. APs in hotspots and residential experiments have their own channel assignment which we do not control. Our wireless clients are laptops with 2GHz x86 processors that run Linux v2.6. with Atheros AR5212 chipsets.

Traffic Shaping: In order to control the backhaul bandwidth of the APs in the testbed, we add the linux traffic shaper tool provided by the Network Emulator utility present in the linux 2.6 kernel on each of our testbed APs.

Video traffic: We use rtptools[5] to generate video flows. We use H.264/AVC software JMEncoder[6] to encode MPEG-4 videos at different bitrates. We use PSNR as the metric for video quality estimation. The PSNR of a video is well correlated with the perceived quality of video experienced by the user. The relationship between user perception expressed in Mean Opinion Score (MOS) and is summarized in Figurer̃effig:videostream.

We first present a detailed analysis of IdleChat evaluation in a controlled testbed in § IV.A to IV.D and validate IdleChat operation in a residential neighborhood.in § IV.E.

IV.A. Can IdleChat aggregate bandwidth?

The key component of the IdleChat architecture is the aggregation of video traffic. We evaluate the aggregation capability of our driver by increasing the number of APs that a client can connect to and streaming video flows over this aggregate link. The APs are on different channels and we vary the number of APs from 1 to 5. Each AP is throttled to have a backhaul capacity of 5Mbps. The wireless clients transmit on 802.11g and the traffic consists of long lived UDP flows. Each experiment is first conducted using the unmodified madwifi driver and then repeated with IdleChat driver. The objective of this experiment is to see if the driver can perform the basic aggregation function in order to increase the capacity at the client.

The throughput results shown in Figure 5 are the average of 10 runs. The unmodified madwifi client can connect to one AP at a time, so at any time it achieves a constant throughput equal to uplink capacity. The IdleChat client starts to see increase in throughput when a second AP becomes available for connection. The IdleChat client is able to aggregate the capacity at each available AP, until it reaches the wireless maximum capacity, accounting for switching overheads.

[4]www.madwifi.org

[5]http://www.cs.columbia.edu/irt/software/rtptools

[6]http://iphome.hhi.de/suehring/tml/

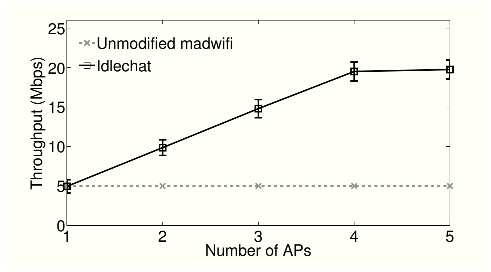


Figure 5: Aggregate capacity in testbed.

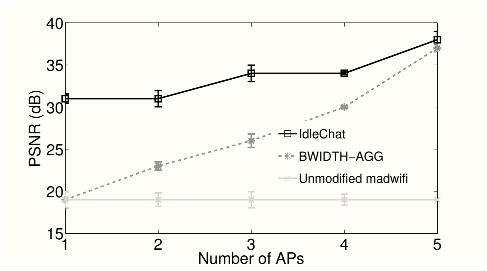


Figure 6: Variation of PSNR with # of APs.

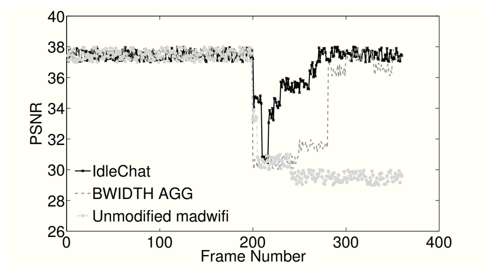


Figure 7: Adaptation to bandwidth change.

We next evaluate the benefits of aggregation in terms of video quality. In the same testbed setup, we stream a MPEG4 encoded video with a bitrate of 30Mbps and length 5 minutes. We use video PSNR as a metric of video quality. Figure 6 shows the PSNR results from 10 such streaming experiments. The client with unmodified madwifi driver achieves 5 Mbps capacity from connecting to one AP and hence sees poor video quality. The benefits of only link aggregation is seen in BWIDTH-AGG where the video quality improves as the number of APs increase. The IdleChat client is able to achieve better video quality than BWIDTH-AGG due to its bit-rate adaptation mechanism described in § III.A.3, wherein the video bitrate is scaled based on bandwidth available . In presence of a single extra AP, IdleChat gets a 10 dB improvement over unmodified client, and a 6 dB improvement over BWIDTH-AGG.

IV.B. Can IdleChat improve timely video packet delivery?

We would like IdleChat to quickly adapt its video delivery quality to any changes in bandwidth. This quick adaptation is critical not only when the client is mobile, but also because broadband links can have high variation in their bandwidths [16].

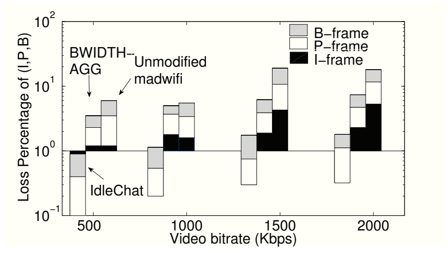
In this experiment setup, there are 2 APs to which the client can connect. AP1 and AP2 have capacities of 2Mbps and 1Mbps respectively, and a 3Mbps video is being streamed. In the middle of the 6th minute (which corresponds to about the 200th frame in this 30fps video), the capacity of the first AP is throttled to 1Mbps and the capacity of the second AP is increased to 2Mbps within 3 seconds (about 100 frames worth of time).

Figure 7 shows that IdleChat is nimble in adapting to bandwidth changes and the PSNR degradation is for a much shorter duration. We see two steps in this adaptation. First, when the capacity at AP1 changes, the queues build up, and because of the increase in jitter values at AP1, the driver notifies the application to reduce the send rate. Quickly, the driver estimates that the capacity of AP2 has increased and the 3Mbps bandwidth is achieved. The unmodified madwifi driver can only collect 1Mbps from AP1. BWIDTH-AGG sees a greater degradation in the video quality till it can collect the bandwidth from AP2.

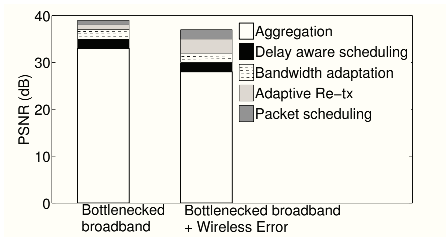
IV.C. Is IdleChat quick in adapting to bandwidth changes?

Figure 8(a) shows that IdleChat improves packet delivery ratio and succeeds in delivering packets with higher utility. Explanation for the figure: The colors stand for (I,P,B) frames. Each group stands for loss at a particular video bitrate as indicated on the x-axis. Within each group, we have 3 bars and each one represents IdleChat (left), BWIDTH-AGG (middle) and unmodified madwifi(right). The y-axis is on a log scale and shows the breakdown of the loss percentages based on the frame type.

We examine the contribution of each of the different mechanisms implemented in IdleChat and described in § III. Figure 8(b) shows the PSNR contributions of each of the techniques in two settings. First, we stream video in a setting where the uplink is the bottleneck but the wireless channel is good. In the second setting, we stream video when the uplink is bottlenecked, and then introduce wireless interference. In both cases, large gains in PSNR are seen due to link aggregation.

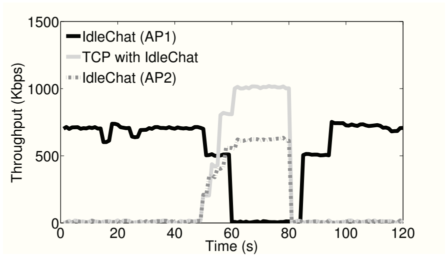


(a) Percentage of (I,P,B) packets delayed.

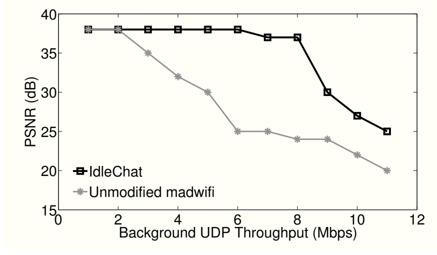


(b) Breakdown of PSNR gains

Figure 8: **Breakdown of gains.**

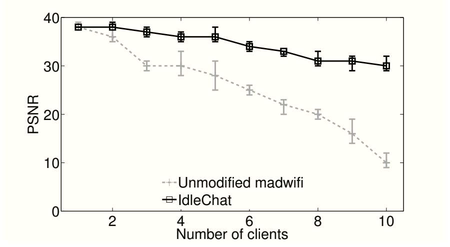


(a) Coexistence with TCP flow.

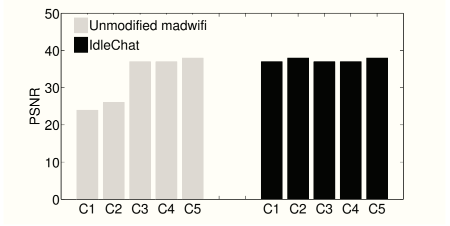


(b) Coexistence with UDP flow.

Figure 9: **Coexistence with flows.**



(a) Client throughput with 5 APs.



(b) Client throughput with 2 APs, one of them being bottlenecked.

Figure 10: **Coexistance with IdleChat clients.**

When the wireless channel is good, we find that the bandwidth adaptation and delay-aware scheduling provide the noticeable benefits. In the presence of wireless losses, the selective retransmission provides the most noticeable gains.

IV.D. How do IdleChat clients behave with other flows?

IV.D.1. Cross traffic

Figure 9(a) shows the behavior of a single IdleChat session in the presence of a TCP flow. In this experiment setup, there are two APs, AP1 with 1Mbps capacity and AP2 with 800 Kbps capacity. We use the IdleChat client to stream an 800Kbps video, and the scheduler picks AP1. We then introduce a saturating TCP flow. We expect the TCP flow to fill up buffers on the link and cause a significant increase in delay. IdleChat flow, which is not designed to compete fairly with TCP flows, should detect this increase in delay and not use the AP at all. When the TCP flow begins to ramp up, IdleChat flow begins to experience delay and it finally stops using that AP. During this time, the entire IdleChat traffic is seen on AP2. At a later time, when the TCP flow is stopped, the IdleChat flow detects when the bandwidth becomes available again and starts to use that AP1 again (since AP1 provides better throughput than AP2). Note that this behavior is in contrast to behavior of FatVAP clients [14] that share the bandwidth equally with TCP flows.

We find a similar behavior in presence of UDP flows. As high bitrate UDP flows start to saturate the link, IdleChat flows start seeing an increase in delay and back off. Figure 9(b) shows the performance of an IdleChat flow with a video bitrate of 3Mbps. The client can connect to 4 APs each with a capacity of 3Mbps, providing an aggregate of 12Mbps. When UDP flows are introduced, we see that the IdleChat client gets the spare capacity in the medium as long as it can find the 3Mbps bandwidth (indicated by the stable PSNR), but quickly PSNR quickly deteriorates when the spare capacity falls below the video bitrate. Unmodified madwifi clients start seeing PSNR deterioration at the instance one 2Mbps flow is introduced since it reduces the capacity on the single AP to which the client is connected.

IV.D.2. Other IdleChat Clients

We show how multiple IdleChat clients compete with each other for bandwidth. In the first experiment setup, we have 5 APs each with 5 Mbps backhaul capacity. We increase the number of clients that are streaming video traffic. Each client streams a video with bitrate 2 Mbps. Figure 10(a) shows the PSNR results over all the clients for this setup over 10 runs. In case of IdleChat clients, the scheduler picks APs that are not loaded, and as a result the clients load balance across APs such that each of them achieve 2 Mbps throughput, resulting in good PSNR values. However, an unmodified clients connect to the AP that is closest to them, leading to unfair load distribution and hence deteriorating PSNR values.

In the next experiment setup, we have two APs and

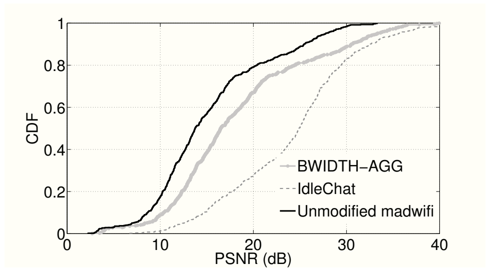


Figure 11: CDF of video PSNR from streaming in residential networks.

multiple clients but one of the APs is now made to be the bottleneck. That is, AP1 has 8 Mbps capacity while AP2 has 2 Mbps. We now stream 2 Mbps bitrate video from each of the client laptops. Figure 10(b) shows the PSNR results for each client both for the case of IdleChat driver and unmodified madwifi driver. In case of unmodified drivers, the two clients that associate with AP2 do not obtain sufficient bandwidth and hence obtain poor PSNR values. In case of IdleChat clients, the clients estimate the load on each AP such that four clients associate with AP1 and one with AP2 such that they obtain 2 Mbps bandwidths such that all clients obtain good PSNR values.

IV.E. IdleChat in Residential Deployments

We demonstrate that IdleChat can be used in residential deployments to improve real-time video quality. To do so, we experiment with IdleChat in two residential areas in Santa Barbara, CA. Each of these locations has two APs, all of them belonging to residences with either cable or DSL connection and an 802.11 AP. We run 1 minute streaming sessions using encoded videos of bitrate 2Mbps with the client located on a desktop connected to a university network. Each session is run at intervals of 10 minutes. Figure 11 shows the CDF of video PSNR values taken over all the streaming sessions from both the residences. The figure shows that IdleChat increases the median PSNR by upto 10 dB. This improvement is the result if bandwidth aggregation as well as the content-aware packet delivery mechanisms implemented in IdleChat.

V. Conclusion

Aggregating multiple backhaul links using 802.11 Access Points has been proposed as a solution to increase backhaul capacity limit and facilitate fast file transfers. Because residential links have unstable bandwidths and high latencies, along with low upload speeds, these solutions do not work well when used for streaming of real-time video. We propose IdleChat, an end-to-end link aggregation scheme for high bitrate real-time video traffic streaming applications such as video chat. To the best of our knowledge, IdleChat is the first system that aggregates backhaul links in a traffic-aware manner to enhance video quality. Experimental evaluations and residential deployment results show that using IdleChat, we can stream videos with bitrates beyond a single broadband link speed and adapt quickly to changing bandwidth and delay conditions.

References

[1] "Cisco Visual Networking Index: Forecast and methodology 2008-2013," http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf.

[2] D. Han, A. Agarwala, D. G. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan, "Mark-and-Sweep: Getting the "inside" scoop on neighborhood networks," in *Proc. of IMC*, Vouliagmeni, Greece, Oct. 2008.

[3] M. Ihmig and P. Steenkiste, "Distributed Dynamic Channel Selection in Chaotic Wireless Networks," in *13th European Wireless Conference*, Paris, France, Apr. 2007.

[4] "FON," http://www.fon.com/en/.

[5] X. Ai, V. Srinivasan, and C. K. Tham, "Wi-Sh: A Simple, Robust Credit Based Wi-Fi Community Network," in *IEEE INFOCOM 2009*. IEEE, Apr. 2009, pp. 1638–1646.

[6] "Meraki Wireless Network," http://meraki.com/.

[7] "The Wi-Fi community built by you," http://www.beta.bt.com/apps/openwifi.

[8] D. Giustiniano, E. Goma, A. L. Toledo, I. Dangerfield, J. Morillo, and P. Rodriguez, "Fair wlan backhaul aggregation," in *Proc. ACM MobiCom*, Chicago, IL, Sep. 2010.

[9] "Community Wireless Solutions," http://www.cuwin.com/.

[10] "FreeNetworks," http://freenetworks.org/.

[11] "OpenWrt supported hardware," http://wiki.openwrt.org/TableOfHardware, Jul. 2008.

[12] "Mushroom networks," http://www.broadbandbonding.com/.

[13] "WiBoost," http://www.wiboost.com.

[14] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi, "FatVAP: Aggregating AP backhaul capacity to maximize throughput," in *Proc. 5th USENIX NSDI*, San Francisco, CA, Apr. 2008.

[15] S. Jakubczak, D. G. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan, "Link-alike: using wireless to share network resources in a neighborhood," *ACM SIGMOBILE MC2R*, vol. 12, no. 4, Oct. 2008.

[16] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, "Characterizing residential broadband networks," in *Proc. of IMC*, San Diego, CA, USA, Oct. 2007.

[17] M. Ihmig and P. Steenikiste, "Distributed dynamic channel selection in chaotic wireless networks," in *13th European Wireless Conference*, Paris, France, Apr. 2007.

[18] "Self-Managing Chaotic Wireless Deployments," http://www.cs.cmu.edu/\~prs/wireless-research/index.html\#chaotic/, 2007.

[19] V. Bychovsky, B. Hull, A. K. Miu, H. Balakrishnan, and S. Madden, "A measurement study of vehicular internet access using in situ Wi-Fi networks," in *Proc. ACM Mobicom*, Los Angeles, CA, Sep. 2006.

[20] A. Akella, S. Seshan, and A. Shaikh, "An empirical evaluation of wide-area Internet bottlenecks," in *Proc. of IMC*, Miami, FL, Oct. 2003.

[21] E. Tan, L. Guo, S. Chen, and X. Zhang, "CUBS: Coordinated Upload Bandwidth Sharing in Residential Networks," in *Proc. of ICNP*, 2009.

[22] R. Raghavendra and E. M. Belding, "Characterizing High-bandwidth Real-time Video Traffic in Residential Broadband Networks," in *Proc. of WiNMee*, 2010.

[23] A. Akella, G. Judd, S. Seshan, and P. Steenkiste, "Self-management in chaotic wireless deployments," in *Proc. ACM Mobicom*, Cologne, Germany, Sep. 2005.

[24] X. Xing, S. Mishra, and X. Liu, "ARBOR: Hang together rather than hang separately in 802.11 wifi networks," in *Proc. IEEE INFOCOM*, San Diego, CA, Mar. 2010.

[25] Y. Wang and Q. Zhu, "Error control and concealment for video communication: A review," in *Proceedings of the IEEE*, 1998.

[26] M. Vishwanath and P. Chou, "An efficient algorithm for hierarchical compression of video," in *Proc. IEEE International Conference on Image Processing*, Nov. 1994.

[27] P. C. Alex, A. Mohr, A. Wang, and S. Mehrotra, "Fec and pseudo-arq for receiver-driven layered multicast of audio and video," 1999.

[28] Abhik Majumdar and Daniel Grobe Sachs and Igor V. Kozintsev and Kannan Ramchandran, "Enhanced Ethernet for Data Center: Reliable, Channelized and Robust," in *15th IEEE LANMAN*, Jun. 2007.

[29] M.-H. Lu, P. Steenkiste, and T. Chen, "Video Streaming Over 802.11 WLAN with Content-Aware Adaptive Retry," in *Proc. ICME*, 2005.

[30] D. Chung and Y. Wang, "Multiple description iamge coding using signal decomposition and reconstruction based on lapped orthogonal transforms," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 9, no. 6, pp. 895–908, Sep. 1999.

[31] M. Jain and C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," in *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.

[32] V. Paxson, "Measurements and Analysis of End-to-End Internet Dynamics," Ph.D. dissertation, U. C. Berkeley, May 1997.

[33] J. Wang and B. Ravindran, "Time-utility function-driven switched ethernet: Packet scheduling algorithm, implementation, and feasibility analysis," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, pp. 119–133, 2004.