Softspeak: Making VoIP Play Fair in Existing 802.11 Deployments

Patrick Verkaik, Yuvraj Agarwal, Rajesh Gupta, Alex C. Snoeren
University of California, San Diego
{pverkaik,yuvraj,gupta,snoeren}@cs.ucsd.edu

ABSTRACT

Voice over IP (VoIP) in 802.11 wireless networks (WiFi) is an attractive alternative to cellular wireless telephony. Unfortunately, VoIP traffic is well known to make inefficient use of such networks. Moreover, we demonstrate that increasing handset deployment has the potential to cripple existing hotspot and enterprise WiFi networks. In particular, our experiments show that VoIP halves the available TCP capacity of an 802.11b hotspot when six to eight VoIP stations share the medium, and effectively extinguishes TCP connectivity when ten VoIP stations are present. Further, we show that neither the higher data rates of 802.11a/g nor the 802.11 standard for quality of service, 802.11e, fully ameliorate the problem. Instead, the problem is rooted in WiFi's contention-based medium-access control mechanism and considerable framing overhead.

To remedy this problem, we propose Softspeak, a pair of backwards-compatible software extensions that enables VoIP traffic to share the channel in a more efficient, TDMA-like manner. Softspeak does not require any modifications to the WiFi protocols and significantly reduces the impact of VoIP on TCP capacity while simultaneously improving key VoIP call-quality metrics. Results show improvements in TCP download capacity of 380% for 802.11b and 50-200% for 802.11g.

1. INTRODUCTION

Voice-over-IP (VoIP) technology is now pervasive in wire-line networks, embodied by wildly successful applications like Skype. Wireless deployment, in contrast, has so far been limited to certain niche products. Recently, however, WiFi-capable consumer phone handsets such as T-Mobile's UMA and the Apple iPhone have been released to the US market in large numbers, portending a huge influx of WiFi VoIP users once third-party applications like iCall [1] become widely available for these platforms. In the near future, it may not be unusual for a dozen active WiFi VoIP handsets to be in range of a single WiFi hot-spot, for example at a local Starbucks.

One might imagine such a scenario would be easily supported by existing installations, as VoIP is a relatively low-bandwidth protocol. For example, given an 802.11b channel with 11 Mbps of capacity, a G.729 VoIP codec rate of 6.4 Kbps, and a combined header size of RTP, UDP and IP of 40 bytes, one might expect a single AP to support over 70 bidirectional VoIP calls and still leave half of the channel capacity for data traffic. It is well known, however, that nothing could be further from the truth; previous researchers have shown that an 802.11b network supports as few as six simultaneous VoIP sessions [4, 7, 16], depending upon the particular characteristics of the network and codecs in use. This counterintuitive result is due to the large per-packet overhead imposed by WiFi for each VoIP packet—both in terms of protocol headers and due to WiFi contention.

Call quality has traditionally been a major concern for WiFi VoIP deployments, since real-time audio traffic has stringent requirements in terms of loss rate, delay and jitter, and needs to be sent at a high rate (e.g., 50–100 packets per second for many VoIP codecs) to maintain acceptable audio quality. In mixed-use cases, best-effort traffic can cause excessive queuing of VoIP traffic at access points and may increase packet loss rate due to contention for the medium. Since a VoIP call occupies only a very small amount of bandwidth (possibly as few as eight bytes of voice data per packet), many researchers [4, 21], and commercial providers [2] have proposed prioritizing VoIP packets, with the unstated assumption that the impact on overall network performance will be minimal. However, we demonstrate experimentally that as few as six VoIP calls remove over half of the TCP capacity in 802.11b. Moreover, prioritizing VoIP sessions runs the very real danger of drowning out all competing best-effort traffic, such as Web browsing and email messaging. Somewhat surprisingly, our experiments show that neither the increased speed of 802.11a/g nor the quality-of-service mechanisms of 802.11e change this reality.

In this paper, we address the impending potential disaster: that widespread VoIP usage will cripple hotspot and enterprise WiFi networks. In addition to quantify-

ing and explaining the impact of VoIP on the capacity of WiFi, we propose backward-compatible modifications to 802.11 that aggregate multiple VoIP clients into the equivalent of a single VoIP client, thus reducing the impact on the network's data-carrying capacity.

Previous work in this domain has proposed the concept of 'downlink aggregation' in simulation [19, 20], which encapsulates multiple VoIP packets into a single packet at the AP, addressed to all VoIP stations associated with the same AP. Our experiments demonstrate, however, that downlink aggregation is insufficient to fully address the problem. We present a complimentary uplink aggregation technique that effectively serializes channel access in the uplink direction by establishing a TDMA-like schedule. We show that this can be done in a distributed manner by independent VoIP stations. We combine uplink and downlink aggregation mechanisms to develop a system called Softspeak that simultaneously improves VoIP call quality while preserving network capacity for best-effort data transfer. Our techniques are independent and complementary in the sense that each provides distinct benefits.

We implement and evaluate Softspeak on a testbed of Linux-based 802.11b/g/e devices within an operational enterprise WiFi network. We show that Softspeak improves residual download TCP capacity of the network substantially, e.g., by 380% in the presence of ten VoIP calls in 802.11b and by 200% in 802.11g (protected mode). We also achieve significant improvements in UDP and TCP uplink capacity, as well as 802.11g unprotected mode. Furthermore, we show that Softspeak improves key VoIP call-quality metrics across all deployments, providing an important incentive for client deployment. To the best of our knowledge, our work is the first to present a system based on commodity hardware that performs both uplink and downlink aggregation to improve the performance of multiple, simultaneous VoIP sessions while increasing the residual data-carrying capacity the WiFi network.

2. THE IMPACT OF VOIP ON WIFI

In this section we empirically demonstrate the degradation of WiFi network capacity as well as VoIP call quality in the presence of an increasing number of VoIP clients. We then employ a detailed simulation of the 802.11 DCF algorithm to determine the precise source of the problem.

2.1 Sources of overhead

The 802.11 protocol is designed to allow clients to access the channel in a distributed manner. Uncoordinated approaches are known to be inefficient under heavy load, however, as collisions become more frequent and the total air-time utilization of the wireless channel reduces dramatically due to air time wasted on garbled frames. This problem is particularly relevant in the case of VoIP traffic, since VoIP clients contend often due to the real-time nature of the traffic. The resulting increased collision rate increases loss and jitter, which in turn degrade TCP performance and harm VoIP call quality.

Furthermore, given the small data payload of VoIP packets the overhead of transmitting the various headers in a VoIP packet becomes considerable: each VoIP packet in a WiFi network is typically encumbered with RTP, UDP, IP, MAC and PHY headers as well as a synchronous 802.11 ACK frame. For example, a G.729 packet may take 157 μs to transmit at the maximum rate in 802.11b, or 273 μs if we include the ACK frame (and assume it is sent at maximum rate). Of this time, the eight bytes of voice data carried inside the packet take up only six microseconds; and even the entire IP packet requires only 35 μs of air time, resulting in a 680% overhead.

Although 802.11g can reduce this overhead to 240%, in protected mode—required when a legacy 802.11b device is present—the overhead remains substantial at over 400% (again optimistically assuming maximum rates are used). Additionally, this overhead may increase in response to loss rate, given the poorly designed rate control algorithms that lower the transmission rate in response to loss, regardless of whether the loss was due to RF or collision. Finally, we note that the resulting increase in air-time scarcity in turn tends to increase collision probability and loss rate as more stations attempt to seize the channel at once, thereby completing a vicious circle.

2.2 Experimental observation

To quantify the impact of VoIP traffic on background data transmissions, we have configured a testbed to reflect a realistic scenario for VoIP usage in the enterprise: stations sending and receiving VoIP traffic are spread out over several offices and are connected to an operational building-wide wireless network. For controlled experimentation we ensure that all stations associate to the same AP and do not roam between different APs. We use wireless cards from two different manufacturers to ensure our results are not artifacts of a particular piece of hardware and consider 802.11b, g and e. (Full details of the testbed are included in Section 4.1.)

2.2.1 Residual capacity

We use our testbed to measure the residual WiFi capacity as well as VoIP call quality in the presence of a varying number of VoIP stations. Here, we measure the residual capacity by simultaneously running a bulk flow and measuring its throughput. We conduct separate experiments for uplink and downlink bulk flows, using both TCP and UDP. Our experiments with UDP flows mea-

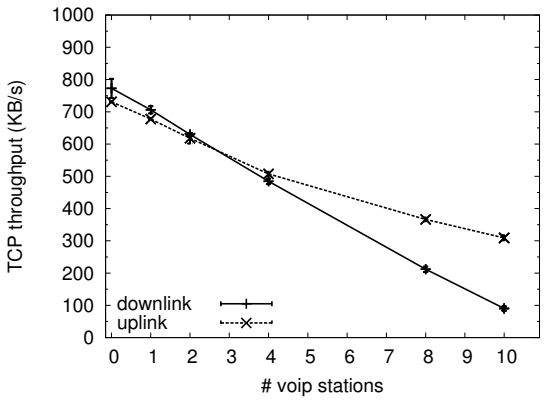

Figure 1: TCP throughput versus the number of active VoIP streams.

sure the raw channel capacity available, while TCP flows measure the effective capacity available for flows that are sensitive to loss and delay. For simplicity, we restrict our discussion to experiments using a single data flow at a single client. We find that increasing the number of data clients does not significantly change the results for TCP capacity or VoIP call quality.[1]

Figure 1 shows the performance degradation of TCP in the presence of a varying number of VoIP stations in a 802.11b network. As we increase the number of VoIP streams, the throughput of a TCP uplink flow (where "uplink" refers to the direction of the TCP data packets) degrades, halving at around eight VoIP streams. Throughput degradation is far worse for a TCP downlink flow, which can be explained as follows. TCP's congestion control mechanism attempts to use the maximum bandwidth available given the loss rate and the RTT. For both cases, the TCP sender needs to share the AP with other traffic for its downlink traffic (data packets for TCP downlink or ACK packets for TCP uplink), and it is therefore at the AP that losses, if any, are expected to occur. Losing a data packet is far worse than losing an ACK packet, however. Therefore, TCP is able to achieve a higher throughput and tolerate a higher loss rate at the AP when sending data uplink. As a result, TCP downlink throughput halves at six VoIP streams and degrades by over 85% in the presence of ten VoIP streams.

The degradation of bulk UDP throughput is less severe than that of TCP: UDP is less sensitive to loss and delay. Nevertheless we observe a significant throughput degradation (over 55% with ten VoIP sessions). We further note that the behavior of UDP uplink and TCP bulk traffic and their impact on VoIP traffic appears quite similar, indicating that in our testbed the TCP uplink behavior is characterized mostly by channel capacity, rather than by loss and delay.

[1]While the degradation in uplink TCP capacity is less significant when multiple data clients contend for the channel, we observe that, instead, the loss rate experienced by the VoIP clients on the downlink increases dramatically (reaching 50% when only three VoIP stations and four TCP stations are active).

2.2.2 Call quality

As we increase the number of simultaneous VoIP sessions, the individual call quality also decreases. Call quality is a function of packet loss rate as well as jitter (as per RFC 3550 [14]). In the presence of TCP uplink traffic and bulk uplink and UDP downlink, the loss rate increases to 50% and jitter increases from one to over six milliseconds as the number of VoIP stations increases from one to ten.

In these cases VoIP traffic undergoes severe drop-tail queuing at the AP queue where it competes with bulk data or TCP acknowledgments. Conversely, TCP downlink traffic is suppressed by the presence of VoIP traffic to an extent that the VoIP loss and jitter remain relatively unaffected. A major challenge is thus to improve TCP downlink performance without sacrificing call VoIP quality.

2.2.3 802.11 protocol extensions

To evaluate whether higher bit-rate protocols alleviate problems of contention and overhead we perform the same set of experiments using 802.11g. We find that in pure 802.11g degradation of bulk traffic is less severe than in 802.11b. For example, TCP downlink performance does not drop as sharply as it does in 802.11b, but degrades in a similar way to TCP uplink and UDP performance. The loss in capacity when ten VoIP clients are present is still substantial, however, ranging from a 32% reduction in the case of UDP bulk downlink traffic, to 39% for TCP downlink traffic. Similarly, while VoIP loss rate and jitter are lower in 802.11g, the loss rate is still unacceptably high: 15–40% when ten VoIP sessions compete with TCP uplink or UDP downlink traffic.

In practice, however, our enterprise WiFi deployment almost never supports only 802.11g clients. For backwards compatibility, 802.11g specifies a "protected mode" to be used when 802.11b stations are detected. In protected mode an 802.11g station precedes each transmission by a CTS frame, thus increasing per-frame overhead.[2] We observe that the capacity degradation caused by 802.11g VoIP clients in an 802.11g protected-mode network is comparable to that of native 802.11b. Thus, the presence of a single legacy 802.11b client (VoIP or otherwise) alongside ten VoIP clients removes 85% of TCP downlink capacity. In addition, we find that whereas VoIP uplink loss rate is negligible in 802.11b, it varies from 10–40% in 802.11g protection mode.

[2]We have not observed the use of full RTS/CTS hidden-terminal protection in our environment.

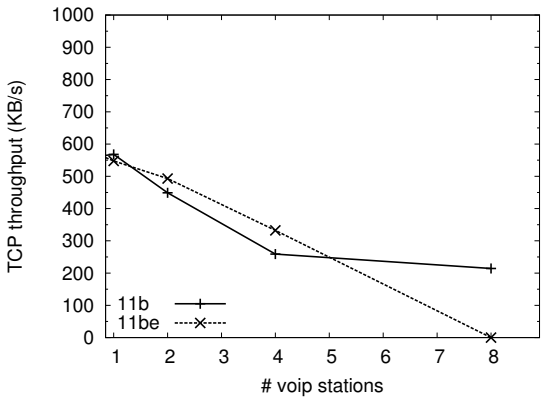

Figure 2: Residual TCP uplink capacity in 802.11b and 802.11b+e.

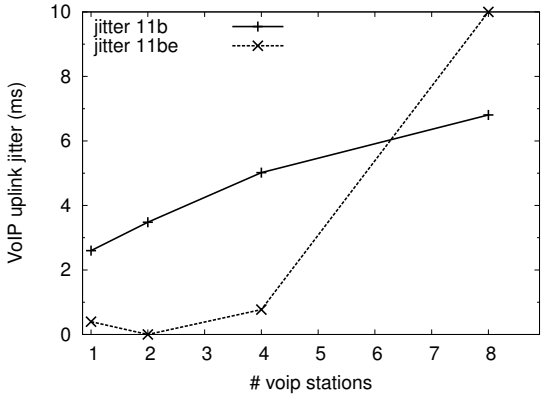

Figure 3: Uplink jitter in 802.11b and 802.11b+e in the presence of TCP uplink traffic.

The 802.11e protocol is specifically designed to allow real-time and data traffic to co-exist efficiently by prioritizing real-time traffic. We compare the performance of 802.11b and 802.11b+e using a popular 802.11e capable access point (a Linksys WAP4400N, different from the Avaya AP-8 used in the previous experiments), with VoIP traffic configured to be classified and prioritized over other traffic at both the AP and the clients. We verify that all stations and the AP do indeed adapt their contention parameters to the type of traffic that is being sent. In the presence of TCP uplink traffic, we observe that compared to 802.11b, 802.11e does indeed decrease loss rate and jitter of VoIP downlink traffic for larger numbers of VoIP stations. However, as shown in Figure 2 this improvement is achieved at the expense of TCP throughput, which degrades far more severely than is the case for 802.11b. Additionally, far from improving VoIP uplink call quality metrics, 802.11e in fact *degrades* uplink jitter for larger numbers of VoIP phones (Figure 3). We conclude that while 802.11e (at least as implemented by a popular AP vendor) is able to improve call quality for a small number of VoIP sessions, it performs poorly when more than a few VoIP clients are combined with background traffic.

2.3 802.11b simulator

While our experiments clearly demonstrate real-world performance problems, it is often difficult to determine to what extent the degradation measured is due to the 802.11 protocol rather than interference, fading, hidden terminals, or other environmental factors. In order to cleanly separate these factors, we have implemented an 802.11 protocol simulator that allows us to evaluate how aspects of the standard distributed coordination function (DCF) algorithm impact performance. We specifically omit the simulation of RF properties, rate adaptation, background broadcast traffic (e.g., DHCP and ARP), and hardware imperfections, in order to show that the DCF algorithm by itself explains our experimental observations.

2.3.1 Configuration and validation

The simulator contains objects representing the AP and wired and wireless stations that send UDP traffic (bulk traffic or based on the traffic characteristics of a VoIP codec). Wired stations are modeled as directly connected to the AP. The wireless stations and AP contend for access using the standard 802.11 DCF algorithm. We parameterize the simulator to mimic the behavior of our testbed hardware (particular settings are detailed later in Table 1) and use a MAC rate of 11 Mbps. We configure an AP queue length of 500 and station queue lengths of 10, but note that our simulation results are not sensitive to the queue length parameters.

We simulate the 802.11b experiment described earlier and find that the results are very similar. For example, simulated throughput degradation is within 10% of the experimental results. The largest difference between the simulated and experimental results is seen in the uplink VoIP loss rate which is 0.8–2.3% versus less than 0.02% on the testbed (for ten VoIP stations).

2.3.2 DCF's share of VoIP impact

Having established that our simulation exhibits the same behavior as the testbed in 802.11b, and that a DCF-based model is sufficient to explain the degradation of our testbed behavior under VoIP, we now proceed to analyze the simulation data to determine what aspect of DCF causes the observed behavior and to motivate our Softspeak architecture. We focus on the air time metric (percentage of time spent using the medium), since it not only directly reflects bulk UDP throughput, but also indirectly reflects loss rate: in a DCF-based model losses are caused by colliding packets, which in turn occupy air time. In Figure 4 we show the wireless air time used as computed by the simulator for each of the follow-

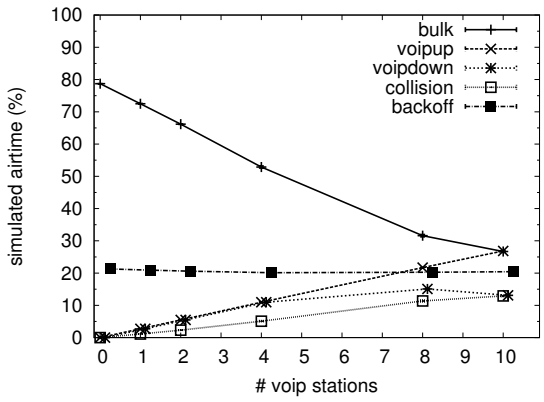

Figure 4: Simulated air-time usage versus the number of active VoIP streams, in the presence of UDP uplink bulk traffic.

ing components: non-colliding bulk traffic (*bulk*), non-colliding VoIP uplink and downlink traffic (*voipup*, *voipdown*), colliding packets (*collisions*), and time during which all stations are backing off or sensing the medium (*backoff*). The results account for all airtime.

Figure 4 confirms that VoIP takes up a large portion of the air time, e.g., 40% for ten sessions, vastly exceeding the air time used by bulk traffic. Most of the VoIP air time (35%) consists of framing overhead. Additionally, 33% of total air time is overhead due to contention (20% backoff plus 13% wasted on collisions). Our techniques presented in the next section are capable of reducing a significant portion of overhead, specifically the framing overhead of downlink VoIP traffic (11%) and the collision time (13%). Based on these numbers there is potential to almost double the bulk-traffic capacity.

3. SOFTSPEAK

Softspeak targets the key challenges of excessive contention and framing to build a software-only solution that can be deployed on existing commodity hardware. The main idea is to aggregate voice traffic by combining many small packets into larger ones. Others have observed that all *downlink* packets must pass through the AP; hence, the opportunity to aggregate exists at either at the AP itself or just before the packets are sent to the AP [19, 20]. However, physically aggregating *uplink* VoIP packets is challenging since there are multiple, independent VoIP senders. Instead, we propose a time-division multiple access (TDMA) scheme that approximates uplink aggregation to the extent that it provides a similar reduction in contention overhead. Our uplink aggregation scheme can function independently of the downlink scheme and requires only client-side modifications. Downlink aggregation, on the other hand, also requires either modifying the AP, or, more realistically, adding a separate "VoIP aggregator" device upstream from the AP. Note that both our mechanisms conform to the existing 802.11 specification and will inter-operate with VoIP stations that do not use Softspeak.

3.1 Uplink aggregation

Our approach to uplink aggregation focuses on reducing the amount of contention created by VoIP clients. Since reducing header overhead by physically aggregating uplink VoIP packets requires extremely tight time synchronization, we instead *logically* aggregate the VoIP clients to appear as a single VoIP client that contends for the channel with non-VoIP clients. Specifically, we alter the contention behavior of the VoIP clients so that they no longer contend with the non-VoIP clients, and then devise a distributed mechanism to schedule the VoIP clients in a TDMA fashion themselves so that they no longer contend with each other either.

We remove the VoIP clients from the standard contention process by modifying their backoff behavior. Instead of sensing the medium for the 802.11-mandated DIFS (DCF interframe spacing) followed by a backoff before sending, a VoIP client only senses for a shorter period of time and does not perform backoff, thus preventing collisions with non-VoIP traffic.[3] This behavior effectively prioritizes uplink VoIP traffic and improves call quality. (A similar mechanism is employed by a commercial product, SVP [2].) By itself, however, this alteration inhibits DCF's ability to prevent collisions among the VoIP stations. In fact, when we simulate two VoIP stations that sense for SIFS (short interframe spacing) without backoff in combination with bulk traffic that uses standard contention, we find that neither VoIP station is able to sustain a viable VoIP session.

To prevent VoIP stations from colliding with each other, we introduce coarse-grained time slots and construct a TDMA schedule for the VoIP clients. When used in combination with downlink aggregation, the downlink aggregator node can assign TDMA slots as well as perform admission control, since it has knowledge of all the clients using our scheme. In the absence of a centralized scheduler, we devise a distributed mechanism (Section 3.1.1) that leverages management frames within the 802.11 protocol to allocate slots.

To implement uplink aggregation, we modify the Ralink RT2560F wireless card protocol stack in Linux 2.6.21 without modifying the WiFi hardware or firmware. Scheduling transmissions is a challenging problem since we do not have control over the WiFi interface at a fine granularity: we can control when the driver sends the frame to the hardware but do not control the physical transmission nor the timing of retransmissions

[3]In the absence of hidden terminals, collisions with ACKs are prevented by 802.11's NAV mechanism.

in case of loss. We address this problem by adjusting 802.11 parameters like inter-frame spacing and backoff as described later in this section.

3.1.1 Slot allocation and admission control

In an ideal deployment, the network operator will have installed a Softspeak VoIP downlink aggregator to do centralized TDMA slot assignment for uplink aggregation. If all available slots are full it can also deny access to a new Softspeak client, in which case that client resorts to normal 802.11 DCF. We expect, however, that in most scenarios it is much easier for individual clients to install Softspeak software than to convince network operators to install new hardware. Moreover, uplink aggregation is useful by itself, i.e., without down-link aggregation, since it reduces contention by uplink VoIP stations. Hence, if clients are unable to locate a VoIP aggregator (the registration process is described in Section 3.2), they proceed with a distributed allocation process.

Independent of how TDMA slots are allocated to clients, VoIP stations need to be synchronized in order to correctly use their assigned slots. Each client in our scheme uses the periodic beacon frame broadcast by an 802.11 AP to synchronize with other VoIP clients. Beacons are sent at fixed intervals (usually 100 ms), and since they are sent by the AP at a low rate, they are typically received by all clients. It is important to note that a VoIP client may also hear beacons from an AP other than the one to which it is associated. To use beacon-based synchronization, VoIP clients need two important pieces of information: a) The AP to whose beacons the other VoIP clients are synchronizing, and b) which TDMA slots they are using. The slot allocation process provides both pieces of information . In the case of distributed slot allocation we encode the information by spoofing the MAC address (6 octets) of a VoIP client as:

- The first three octets (known as the OID) are taken from the reserved or unassigned OID address space to ensure the resulting address is valid and unique.
- The next two octets are the same as the *last two* octets of the BSSID of the AP to whose beacons the VoIP station is synchronizing.
- The last octet is used to denote the particular real time slot the VoIP station is using or wants to use.

The main concern when coordinating clients is that there is no guarantee they can hear each other's transmissions. Hence, Softspeak clients coerce the AP into generating specially crafted packets that the other clients can hear. VoIP stations using uplink aggregation periodically (e.g. once a second) send directed active Probe-Requests on the channel and to the AP that they are currently using using the modified MAC address. The destination (unmodified) AP will respond with a Probe-Response packet whose destination is the VoIP station's modified MAC address, which is heard by all associated clients.

A new VoIP station that wants to use uplink aggregation scheme first goes into promiscuous mode for a few seconds to sense the channel to check if there are any special Probe-Response packets (easily identifiable by the first three octets of the destination MAC address), and, thus can determine which slots are being used at each AP. The VoIP client then picks an unused slot and starts to periodically broadcast a Probe-Request with its source MAC address denoting which AP and slot it is using along with the other clients. As before, the AP sends Probe-Responses which can be heard by new VoIP clients wanting to join. Finally, when a VoIP station finishes its session it stops sending the Probe-Requests for the particular slot it was using and any new VoIP station can subsequently use it.

Our slot assignment scheme seamlessly supports dynamic node arrivals and departures. Moreover, this scheme works even when considering neighboring APs on the same channel as the crafted MAC address of the VoIP stations also has the last two octets of the BSSID that the VoIP station is associated to. Finally, our scheme works even if APs use various 802.11 security features since Probe-Request and Probe-Responses are always sent unencrypted. We were able to successfully test our scheme with an AP that had multiple security features enabled: MAC-address-based access control, WPA2 or WEP encryption, and broadcasting SSID disabled.

3.1.2 Synchronizing TDMA slots

Once slots are allocated, each station ideally contends for the channel in its assigned slot and refrains from contending outside its slot. By default, the Linux 2.6 kernel timer interrupt is programmed to fire every millisecond; we show later that this also happens to be close to the optimal granularity for VoIP slotting in 802.11b. Using one-millisecond slots, a TDMA scheme can support ten simultaneous VoIP stations using a codec with 10-ms inter-packet arrival rate, or 20 stations using a 20-ms codec. Since in 802.11a/g the frames for these codecs take less air time, Softspeak is able to use smaller slots, allowing a larger number of VoIP stations to be admitted. We have not yet implemented sub-millisecond slotting, however.

A straightforward implementation of one-millisecond slotting is to suspend and resume transmission from within Linux's timer interrupt handler, in accordance with a station's assigned slot. However, the naive approach faces two problems: clock skew and timer inaccuracy. Figure 5 plots a time series that illustrates both of these issues. In this experiment, a single station uses `iperf` to emulate a G.729 VoIP codec with a 10-ms inter-packet arrival rate. We manually assign the station

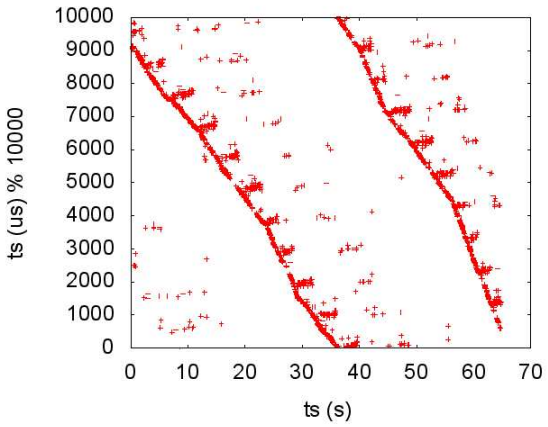

Figure 5: Time series of transmission times by single station, no synchronization.

a static TDMA slot; there is little to no background traffic on the same AP during the experiment

In the figure, the x axis plots time in seconds, and the y-axis shows the start time of each transmission modulo 10,000 μs (10 ms). The figure shows the effect of the timer interrupt firing faster than 1,000 times per second as well as `iperf` sending slightly slower than the configured rate of 100 packets per second. If the timer interrupt and `iperf` operated at their correct rate, we would expect to see a single horizontal band corresponding to the station's assigned slot. Instead, `iperf` schedules packets at a rate slower than the timer interrupt, and as a result iperf and the implemented TDMA slot drift with respect to each other. When `iperf` happens to send inside the slot, a short almost horizontal line appears starting at the bottom of the slot (the slight upward slope of this line is the clock skew). Once transmissions reach the top of the slot, the packets get buffered until the start of the next slot, causing the downward sloping lines. The slope of the latter lines is caused by the timer interrupt firing too fast.

Since there is a distinct possibility that different stations exhibit different degrees of skew, possibly even varying across time, it is important to synchronize the timing of the stations. We resolve this issue by effectively slaving each station's clock to an AP. Specifically, we reset the timer every time a station hears the periodic beacon frame from the AP that was assigned during the slot allocation process. On the Soekris net4801 in our testbed, Linux uses the programmable interval timer (PIT) as its time interrupt source. Therefore, we modify the driver to reset the PIT automatically every time it hears a beacon, which we have measured to be roughly once every 102–103 ms for the APs in our network.[4]

[4]The unit of inter-beacon time in 802.11 is 102.4 ms, rather than 100 ms.

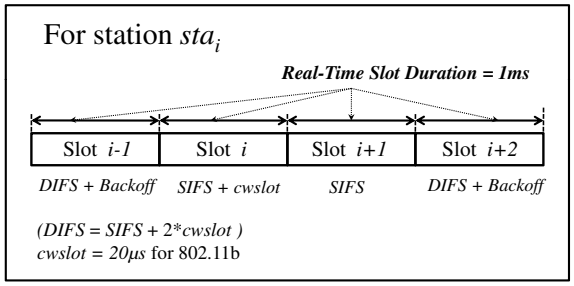

Figure 6: Illustration of dynamic IFS showing the various contention parameters, depending on the slot sta_i is contending in.

Ideally the first slot following a beacon should start exactly at the beacon time. However, there are inevitable delays between when the beacon is generated by the AP and when it is received and processed by a station, and also between the time that the station driver generates a packet for a particular slot and the time that it is transmitted. We have measured this delay to be about 0.5 ms for our hardware. We expect the delay to vary across different stations, but as subsequent figures show it is consistent enough across multiple stations with the same hardware configuration that a station's synchronization can be tuned for that hardware.

3.1.3 Controlling transmission timing

An obvious complication with our scheme is that when a TDMA slot starts, a station other than the station that has been assigned the slot may already be transmitting a frame. At 11 Mbps a maximum-sized IP packet (1,500 bytes) together with ACK will take 1,376 μs, potentially delaying the station by that time from the start of its slot into the next slot[5]. In addition, the VoIP station may repeatedly fail to capture the channel even while actively contending. We address this challenge by letting the WiFi card driver adjust the way VoIP station contends for the channel during its assigned slot, a mechanism we term *dynamic IFS* (dynamic inter-frame spacing).

In DCF, stations contend using an inter-frame spacing of DIFS=SIFS $+(2 \cdot \text{cwslot})$ followed by a backoff[6]. We use the two, 20-μs cwslot intervals starting at SIFS and (SIFS+cwslot) to (a) prioritize the VoIP traffic over non-VoIP traffic and (b) distinguish among different VoIP stations to avoid collisions. Accordingly, we let each station contend as follows. Figure 6 considers a station sta_i which is assigned TDMA slot i. During the station's TDMA slot it contends with IFS=(SIFS+cwslot) (and no backoff). In slot $i+1$, it contends with IFS=SIFS. In any other slot it contends as specified by DCF.

[5]We assume short preambles throughout the paper.

[6]By cwslot we denote an 802.11 contention window slot, 20 μs in 802.11b. It is unrelated to the Softspeak's 1-ms TDMA slot.

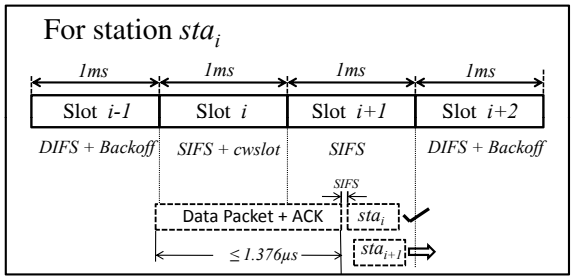

Figure 7: Dynamic IFS in the presence of other data traffic. In slot $i+1$ sta_i wins over sta_{i+1} because it uses IFS=SIFS rather than IFS=SIFS+cwslot

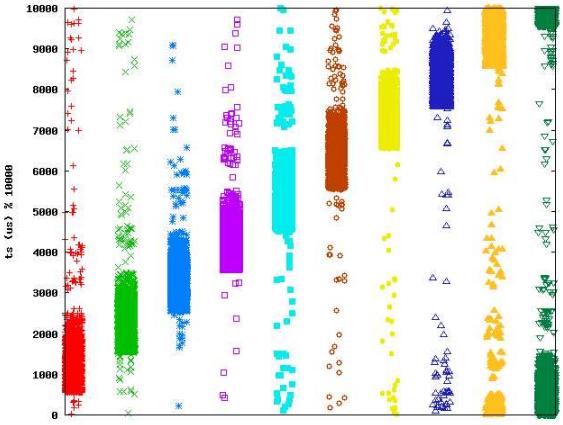

Figure 8: TDMA slotting by ten VoIP stations using dynamic IFS in the presence of UDP down-link background traffic.

Now, let us consider the scenario as illustrated in Figure 7, in which a station sta_i in TDMA slot i is delayed into the next TDMA slot $i+1$ by an ongoing transmission and assume for the moment that sta_i's packet was ready at the start of the slot i. After the transmission has ended, stations sta_i and sta_{i+1} contend for the channel. However, due to the assigned contention parameters, sta_i is guaranteed to win over station sta_{i+1}. Furthermore, after sta_i has finished transmitting and received its ACK (after 430 μs for a large payload G.711 codec), there is still at least 2 ms - 1,376 μs - 430 μs = 194μs for sta_{i+1} to commence its transmission and therefore not contend in TDMA slot $i+2$. It can be shown that in the absence of retransmissions, as long as (a) the duration of a VoIP frame is less than one TDMA slot and (b) the duration of a bulk frame is less than two TDMA slots, station i will never contend in slot $i+2$. Note that even if due to retransmissions a station ends up contending in a TDMA slot other than i or $i+1$ it will do so using conventional DCF contention parameters and do no worse than without our improvements.

Figure 8 plots the transmission start times of ten VoIP stations, each assigned a separate TDMA slot, when competing against background traffic. In particular, a UDP sender generates background traffic in the downlink direction to a separate wireless station. Using dynamic IFS, the slotting is clearly defined, with the majority of transmissions not commencing more than one slot away.

3.2 Downlink aggregation

Our current implementation of downlink aggregation introduces an aggregator component that is placed before the WiFi AP (uplink from the AP). The aggregator is on-path and transparently forwards all traffic to and from the AP; non-VoIP traffic is forwarded without modification. The aggregator buffers VoIP frames destined for wireless stations and releases a frame encapsulating the buffered frames at a regular interval (every M ms, where M is the packetization interval of the VoIP codec.) By combining all the VoIP sessions into one packet per codec interval, downlink aggregation can virtually eliminate the marginal header and contention overhead of additional VoIP clients. There is a down side however: when the aggregator buffers a packet, it adds a constant delay of $M/2$ ms (in expectation), e.g., 5 ms given a 10-ms codec.

In our scheme, when a new VoIP session first starts up (or when the station roams to a different AP) it registers with the aggregator node, which we implement on a separate Linux machine. When there is downlink traffic addressed to a registered VoIP client, the aggregator buffers the packet and combines it with all other buffered packets into a single encapsulated packet that it sends out at fixed intervals (e.g., 10 ms for G.729). The aggregator node uses the IP header information from the most recently heard uplink packet (say from station $S1$) to construct a new frame. Addressing the packet to $S1$ increases the likelihood that the packet will be acknowledged by a currently active VoIP client. We define an aggregation header that stores the set of destinations and original IP packet lengths for each station. The aggregation header is prepended to the UDP header and packet payload for $S1$, and then the respective IP and UDP headers and payloads for the remaining buffered VoIP packets are appended.

In contrast to previous proposals, we address the aggregated frame to only one of the VoIP stations; we configure the WiFi interface of each of the VoIP stations to be in promiscuous mode to allow them to receive the aggregated packets regardless of the destination. In either case, the client passes aggregated packets to the Softspeak module that de-encapsulates the packet, extracts the portion meant for the current station, and passes it up the networking stack. Because the aggregated packet is addressed to only one station, there will be at most one

MAC-layer acknowledgment. Wang *et al.* [19], on the other hand, propose the use of multicast in order to eliminate the MAC ACK frame. We preserve the ACK frame for two pragmatic reasons. First, in our experience, while obviously unable to eliminate all loss, the single ACK frame is a cost-effective mechanism to protect the aggregated packet against many collisions. (The situation changes somewhat due to poor channel quality in some 802.11a/g deployments, however; see Section 4.4.) Secondly, and perhaps more importantly, commodity access points typically transmit multicast frames only at DTIM (delivery traffic indication message) intervals—a multiple of the beacon interval (typically 100 ms)—to interoperate with clients in power-save mode, introducing intolerable delay in existing deployments.

4. EVALUATION

We now evaluate the effect of each of our aggregation schemes, down-link and up-link, both independently and in concert. In particular, we show that (a) our schemes significantly increase the available channel capacity for both bulk UDP and TCP streams while maintaining—or even improving—VoIP call quality, and (b) our implementation of aggregation is close to optimal.

4.1 Experimental testbed

The wireless infrastructure in our building is a managed 802.11b/g deployment comprising enterprise-class Avaya AP-8 access points. There are multiple APs per floor which are configured to be on orthogonal channels to increase spatial diversity. Our building is also instrumented with the Jigsaw [5] wireless monitoring infrastructure, allowing us to get a very detailed view of all wireless traffic in our building.

We have configured eleven Soekris net4801 boxes to act as VoIP stations each with two mini-PCI wireless cards: an Atheros AR5212 chipset-based card and an Ralink RT2560F-based interface. The net4801 is a single-board based computer with a 266-Mhz CPU running a fully functional Linux operating system. To simplify our experiments, we emulate VoIP traffic using `iperf`. We use `iperf` to generate UDP traffic that mimics a commonly used VoIP codec, G.729—a bidirectional packet stream, where each direction has a 10-ms inter-packet arrival, an eight-byte voice payload, and twelve additional bytes of RTP header.[7] The Soekris boxes and AP each have RTS/CTS disabled.

We employ ten commodity PCs connected over wired gigabit Ethernet as endpoints for the VoIP traffic generated by the Soekris boxes. Essentially, each PC-Soekris pair serves as a distinct bi-directional VoIP call. Additionally, one PC-Soekris pair serves as the end points of a bulk transfer (TCP or UDP) to determine the available capacity of the wireless channel in the presence of the VoIP traffic. The TCP receive-window size is configured to be large enough that our TCP transfers are never receive-window limited. Unless otherwise noted, bulk transfer is conducted through the Atheros card, while the Ralink interfaces send and receive VoIP traffic.

[7] Variants of G.729 also run at longer inter-packet times and/or increased voice payload sizes.

Card	CWmin	CWmax	Retry limit
Ralink RT2560F	8	256	8
Atheros AR5212	32	32	11
Avaya AP-8	16	16	11

Table 1: 802.11b contention parameters measured for our wireless hardware.

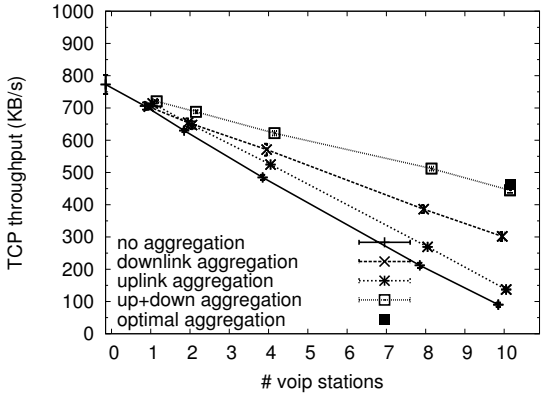

Figure 9: TCP down-link capacity under different aggregation schemes.

We have recorded the default contention parameters for the various device in our testbed using the Jigsaw wireless monitoring infrastructure (Table 1). We note that neither the Atheros card nor the Avaya AP appears to double its contention window size on retries, in contrast with the default behavior specified by 802.11.

4.2 Results for 802.11b

Figures 9-20 compare bulk throughput and VoIP call quality metrics across all combinations of applying uplink and/or downlink aggregation in 802.11b, for the following three cases: TCP up and downlink, and UDP bulk uplink; we discuss the case of UDP bulk downlink in Section 4.3. The plots for VoIP uplink loss are omitted since they are negligible ($< 0.1\%$) for all data points. The most important conclusions are that (a) applying a combination of uplink and downlink aggregation improves residual bulk throughput, in some cases drastically, (b) call quality metrics are mostly insensitive to a varying numbers of active VoIP calls, (c) applying only one of the uplink or downlink aggregation above does not achieve this stability for all three cases of bulk traffic load, and

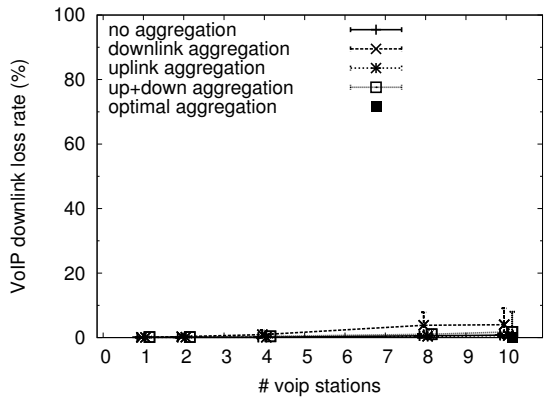

Figure 10: VoIP downlink loss rate in the presence of TCP downlink traffic under different aggregation schemes.

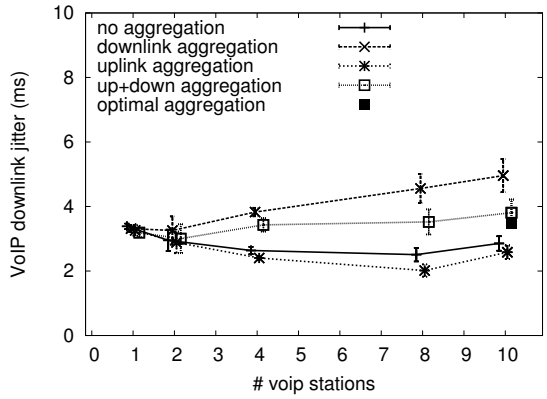

Figure 11: VoIP downlink jitter in the presence of TCP downlink traffic under different aggregation schemes.

(d) our gains approach those of an optimal aggregation scheme.

We summarize the benefits of combined up and downlink aggregation over no aggregation, for the case of ten VoIP sessions, as follows:

TCP downlink: Capacity multiplies 4.8 times from 92 KB/s to 445 KB/s (Figure 9). As a result, call quality is slightly affected, with VoIP downlink loss and jitter increasing slightly—from 0.74% to 1.8% (Figure 10) and from 2.9 ms to 3.8 ms, respectively (Figure 11). Uplink jitter does not change significantly (Figure 12).

TCP uplink and UDP uplink: Capacity increases by around 50% (Figure 13). VoIP downlink traffic improves from being completely unusable for VoIP (over 50% loss rate) to being usable: less than 2% downlink loss rate and less than 2-ms jitter (Figures 14 and 15).

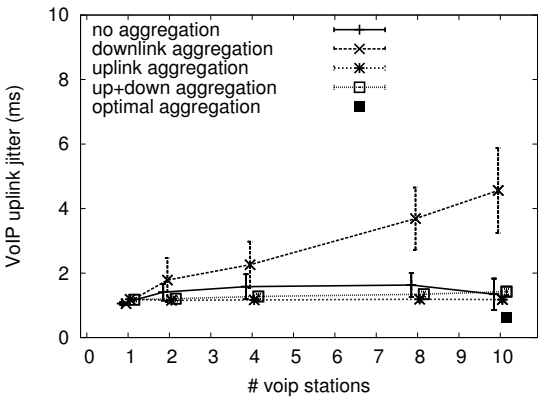

Figure 12: VoIP uplink jitter in the presence of TCP downlink traffic under different aggregation schemes.

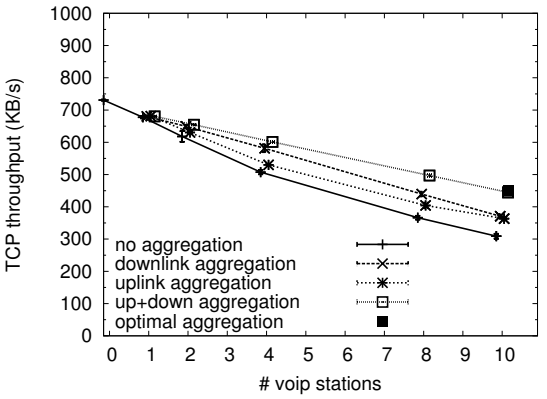

Figure 13: TCP up-link capacity under different aggregation schemes.

While these results show that Softspeak improves the efficiency of 802.11b networks in the presence of VoIP, both in terms of residual TCP capacity and VoIP call quality, an important question is whether further improvements to our implementation could be made. For example, it might be the case that our implementation of uplink aggregation lacks sufficient control of VoIP packet scheduling, causing collisions. An optimal implementation (e.g., one that is implemented in the 802.11 hardware or firmware) might do a better job at controlling the emission of frames according to the TDMA schedule.

To investigate to what extent further improvements may be made to our implementation (but while remaining faithful to Softspeak), we compare our results with those based on an emulation of an optimal implementation. We emulate downlink aggregation by replacing the individual VoIP senders that generate downlink VoIP traffic by a *single* sender that generates packets of the size produced by the downlink aggregator, eliminating any

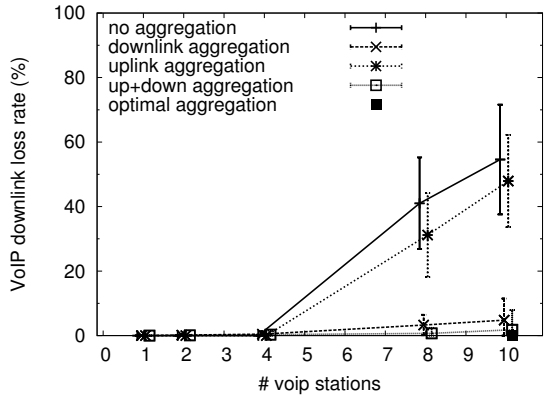

Figure 14: VoIP downlink loss rate in the presence of TCP uplink traffic under different aggregation schemes.

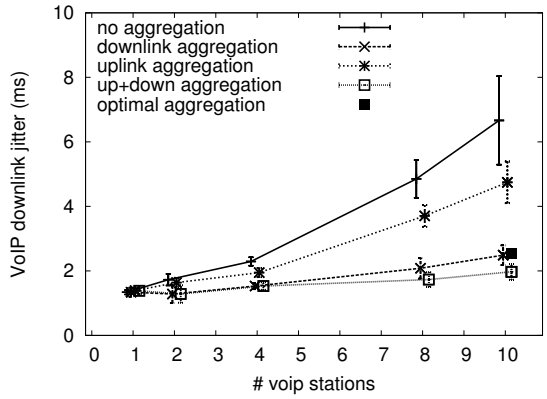

Figure 15: VoIP downlink jitter in the presence of TCP uplink traffic under different aggregation schemes.

jitter and loss potentially caused by the downlink aggregator. We emulate uplink aggregation by replacing the VoIP stations by a *single* VoIP station that sends packets on behalf of all VoIP stations, in other words, it sends packets at ten times the codec rate. The single VoIP station naturally serializes the transmission of uplink VoIP packets, thereby eliminating any collision among VoIP stations. To minimize the probability of colliding with other traffic, it uses SIFS without backoff. In Figures 9-16 the results of the emulation are plotted as an 'optimal aggregation' point for ten VoIP clients. In terms of capacity we achieve close to what is optimally achievable, and jitter is within 1 ms of optimal. The uplink and downlink loss achieved by optimal (not shown) is 0; we achieve $< 0.1\%$ on uplink. On downlink, our scheme reduces loss to 2% which is significantly higher than optimally achievable, but still a vast improvement over the loss rate incurred without aggregation.

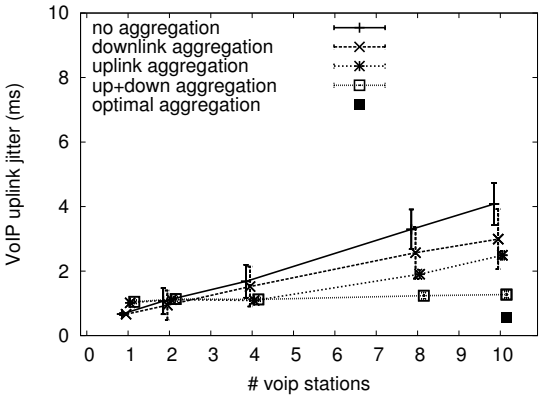

Figure 16: VoIP uplink jitter in the presence of TCP uplink traffic under different aggregation schemes.

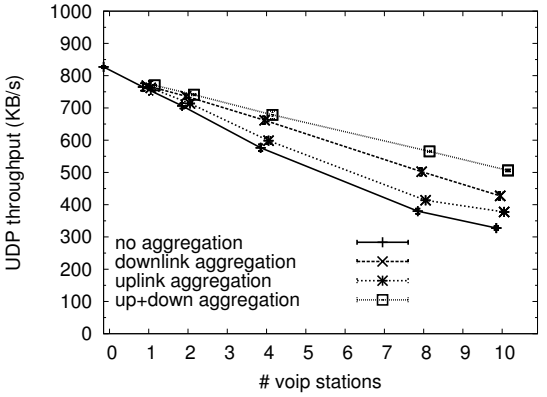

Figure 17: Bulk UDP up-link capacity under different aggregation schemes.

4.3 UDP and 802.11e

While Softspeak can improve the capacity available for bulk UDP downlink traffic in 802.11b networks (Table 2), it cannot simultaneously reduce the high VoIP downlink loss rate that result from competing with a CBR UDP flow. These losses are caused by the AP queue filling with bulk UDP downlink traffic, combined with the fact that UDP does not respond to increasing loss and delay. These losses can only be ameliorated by adding prioritization at the AP: (aggregated) VoIP packets would therefore not be dropped regardless of the amount of UDP traffic buffered at the AP. Luckily, such prioritization is part of the 802.11e standard.

4.3.1 Prioritization

Unfortunately, our testbed hardware cannot simultaneously support 802.11e (supported only by the Atheros chipset) and Softspeak (which is currently only implemented for the Ralink interfaces). We therefore evaluate aggregation combined with 802.11e-like prioritization at

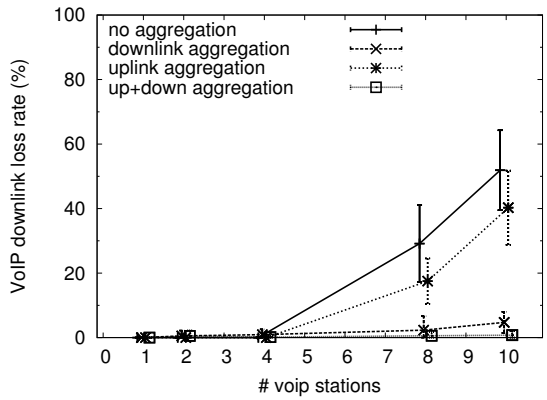

Figure 18: VoIP downlink loss rate in the presence of bulk UDP uplink traffic under different aggregation schemes.

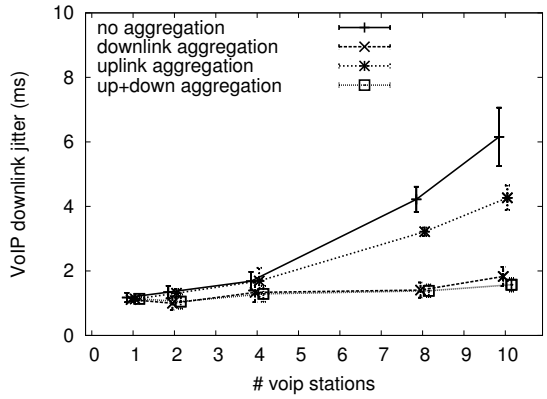

Figure 19: VoIP downlink jitter in the presence of bulk UDP uplink traffic under different aggregation schemes.

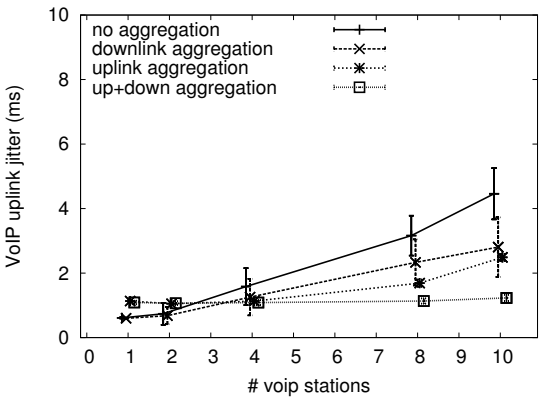

Figure 20: VoIP uplink jitter in the presence of bulk UDP uplink traffic under different aggregation schemes.

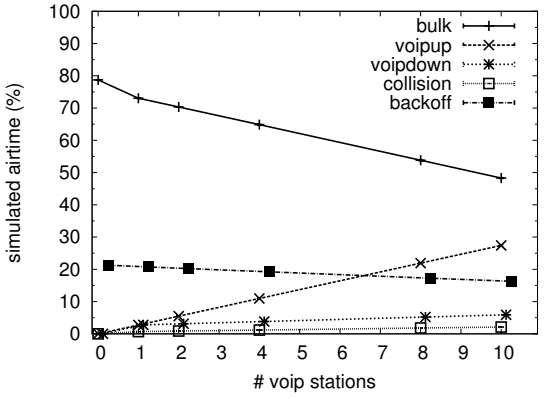

Figure 21: Simulated airtime usage versus the number of active VoIP streams, in the presence of UDP uplink bulk traffic. Both uplink and downlink aggregation are enabled.

the AP using our simulator (described in Section 2.3). We note that, consistent with our results in Section 2.3.2, our simulator produces results similar to those measured experimentally for the case of UDP without prioritization for combination of uplink and downlink aggregation, and we therefore believe that we can extrapolate to the case of AP prioritization. Table 2 shows that when we combine aggregation with prioritization, we not only achieve a 47% improvement on downlink bulk capacity, but also maintain high VoIP call quality (vs. no aggregation, no prioritization).

4.3.2 Airtime utilization

Implementing Softspeak in our simulator also allows us to isolate the source of our performance improvement. Figure 21 shows the simulated airtime plot corresponding to Figure 4, but with uplink and downlink aggregation enabled (and no prioritization). The figure indicates that we have achieved our objective of converting almost all time spent on downlink framing overhead and on collision into bulk data capacity. Consistent with the reduction in collision airtime we have also reduced the collision rate, thereby improving both VoIP loss rate and jitter metrics.

4.4 802.11g

For 802.11g we observe that Softspeak as currently described makes significant improvements in capacity (24–32%, for ten VoIP stations), while maintaining or lowering jitter and VoIP uplink loss to negligible levels. Recall from Section 2.2.3 that when 802.11g runs in protected mode, TCP downlink capacity suffers tremendously in the presence of VoIP. Using Softspeak we are able to triple the TCP downlink capacity for ten VoIP stations. However Softspeak also introduces significant loss of around 5% (unprotected mode) and up to 15% (protected mode), where in some cases virtually none was

Metric	no agg	agg	agg+prio
Downlink bulk throughput (KB/s)	375	605	561
Downlink VoIP loss rate	67%	61%	<0.1%
Downlink VoIP jitter (ms)	6.1	2.2	0.61
Uplink VoIP loss rate	0.82%	<0.1%	<0.1%
Uplink VoIP jitter (ms)	1.8	0.52	0.33

Table 2: The effectiveness of combining aggregation with prioritization in the presence of ten VoIP stations and downlink bulk traffic (simulated). (Bulk capacity without VoIP traffic is 924KB/s).

experienced without enabling Softspeak.

On closer examination, we find that these losses are caused by the fact that at higher 802.11g rates, frames are more likely to be corrupted by bad RF. Since Softspeak's downlink aggregation scheme receives link-layer acknowledgments from only one VoIP client, only frame losses experienced by that client result in retransmission. Frame corruption experienced by other clients remains unnoticed. One solution (similar to the way 802.11 handles broadcast frames) is to transmit aggregated VoIP frames at the lowest data rate, thereby increasing the probability of successful reception. Doing so, however, would incur a considerable cost in airtime and, more pragmatically, requires modifications at the AP.

Instead, we observe that VoIP clients that experience a persistently high frame loss from downlink aggregation can simply opt out of the downlink portion of Softspeak. By deregistering with the aggregator, these clients receive separate VoIP frames as in the non-aggregated case. Note that these stations can still participate in uplink aggregation. To demonstrate that such a scheme can gracefully address this situation in practice, we run an 802.11g protected mode experiment using eight of the VoIP stations. One of the VoIP stations is a poor receiver. When we enable both up and downlink aggregation, we measure a 50% VoIP downlink loss rate for the poor receiver, whereas the remaining VoIP stations experience only a couple percent loss rate. Next we exclude the poor receiver from aggregation, after which the station's loss rate drops to a level comparable with others. The price for using this technique is a decrease in bulk capacity. In this case, we measure a 5% decrease in TCP throughput.

5. RELATED WORK

We are far from the first to observe that WiFi networks are severely limited in the number of VoIP sessions they can support. Researchers have studied VoIP call quality in wireless networks, and attempted to quantify how many VoIP calls traditional WiFi networks can handle while maintaining various quality-of-service (QoS) metrics. These range from analytical and simulation-based studies [3, 10, 18, 21] to those that validate findings by measurements on actual experimental testbeds [4, 7, 16]. While precise findings vary, all studies agree that the effective VoIP capacity of a WiFi network is far less than one might expect given the limited bandwidth usage of typical VoIP streams.

The poor performance of VoIP in WiFi networks is not protocol specific, but is symptomatic of a general issue with any CSMA (carrier-sense, multiple-access) network: channel access and arbitration becomes increasingly inefficient as load (in terms of number of attempted channel accesses, not necessarily bandwidth) increases. Time-division multiple-access (TDMA) can be far more efficient under heavy load. Indeed, 802.11 includes both a point coordination function (PCF) mode and a hybrid coordination function (HCF) mode, in which the AP explicitly arbitrates channel access. Unfortunately, very few deployed 802.11 networks employ these modes.

If one considers modifying the hardware, a variety of options exist. For example, researchers have proposed modifying the way standard 802.11 PCF works [3] as well as alternative ways of implementing 802.11e-like functionality [18]. Of course, non-backwards compatible device modification does not address the issue facing today's networks. Accordingly, researchers have proposed a variety of explicit time-slotting mechanisms, both within the context of the infrastructure-based networks we consider here [8, 11, 12, 15, 17] as well as multi-hop mesh networks [9, 13].

MadMAC [15], ARGOS [9], and the Overlay MAC Layer (OML) [13] each propose to effectively add another software layer on top of the 802.11 MAC to enable time-slotting on the order of 20 ms. Snow *et al.* [8, 17] present a similar TDMA-based approach to power savings where each slot is of the order of 100 ms and requires changes at the access points themselves. The scheduling granularity in all of these schemes is too coarse to effectively support most VoIP codecs. While software TDMA (STDMA) [8] proposes to do TDMA for all traffic, they focus particularly on the performance of VoIP. Their approach is a substantial and backward-incompatible modification to 802.11 that requires very accurate clock synchronization. More significantly, each of the above schemes require the entire network to support the new TDMA architecture with no support for unmodified clients using the default 802.11 DCF.

Over and above TDMA mechanisms, the SoftMAC [11] and MultiMAC [6] projects also suggest modifications to 802.11 MAC behavior, including changing the ACK timing and modifying back-off parameters. The authors do not provide many details about their implementations, however, nor do they evaluate their scheme with deadline-driven VoIP traffic.

Focusing explicitly on improving the performance of VoIP traffic in mixed-use networks, various proposals have suggesting prioritizing VoIP traffic [4, 21], no-

tably a commercial product, Spectralink Voice Priority (SVP) [2]. SVP prioritizes down-link VoIP packets in the AP transmit queue and does not back-off when attempting VoIP transmissions. While we leverage similar optimizations, SVP does not do scheduling, thereby increasing collision rate due to the lack of back-off.

6. LIMITATIONS

We note that independent proposals to use SIFS for contention are fundamentally incompatible, since their frames would collide. We believe that our uplink aggregation scheme can be generalized to include support for other real-time traffic and other schemes. We leave the development of such a generalized TDMA in CSMA protocol as future work.

Softspeak relies on clients overhearing each other's VoIP communication to perform downlink aggregation. Therefore, if a WLAN uses a WiFi encryption protocol such as WPA2, downlink aggregation is no longer possible. (Uplink aggregation, on the other hand, is not affected by encryption.) We note however that many popular hotspots do not enable encryption. Protocols encrypted above the MAC layer, such as Skype, can take advantage of Softspeak's downlink aggregation, as long as they allow some way of being detected as VoIP.

7. CONCLUSION

As WiFi-capable smartphone handsets become more popular, the number of simultaneous VoIP users is likely to increase dramatically in WiFi hotspots and enterprise networks. While previous work has aggregated downlink VoIP traffic, it has focused on improving VoIP call quality in the face of competing best-effort traffic, it has ignored the impact of a large number of simultaneous VoIP sessions on the residual capacity of the network. We find that a relatively small number (five to six) VoIP sessions can effectively halve the capacity of an 802.11b network, and the network becomes essentially unusable once more than a dozen VoIP sessions appear.

We present Softspeak, a set of backward-compatible changes to WiFi that address contention and framing overhead. We develop a software-based design that can be deployed on commodity hardware. We show that our dynamic IFS contention scheme, combined with downlink aggregation, dramatically reduces the impact of VoIP on network capacity yet improves call quality.

8. REFERENCES

[1] iCall for the iPhone (beta). `http://www.icall.com/iphone/`.

[2] Spectralink Inc - Spectralink Voice Priority (SVP). `http://www.spectralink.com/files/literature/SVP_white_paper.pdf`.

[3] J. Al-Karaki and J. Chang. A simple distributed access control scheme for supporting QoS in IEEE 802.11 wireless LANs. *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, 1, 2004.

[4] F. Anjum, M. Elaoud, D. Famolari, A. Ghosh, R. Vaidyanathan, A. Dutta, P. Agrawal, T. Kodama, and Y. Katsube. Voice performance in WLAN networks-an experimental study. *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, 6, 2003.

[5] Y.-C. Cheng, M. Afanasyev, P. Verkaik, P. Benkö, J. Chiang, A. C. Snoeren, S. Savage, and G. M. Voelker. Automating cross-layer diagnosis of enterprise wireless networks. In *Proceedings of the ACM SIGCOMM Conference*, Kyoto, Japan, Aug. 2007.

[6] C. Doerr, M. Neufeld, J. Fifield, T. Weingart, D. Sicker, and D. Grunwald. MultiMAC-An Adaptive MAC Framework for Dynamic Radio Networking. *IEEE DySPAN*, 2005.

[7] S. Garg and M. Kappes. An experimental study of throughput for udp and voip traffic in ieee 802.11b networks. *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, 3:1748–1753 vol.3, 16-20 March 2003.

[8] F. Guo and T. Chiueh. Software TDMA for VoIP applications over IEEE802.11 wireless LAN. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2366–2370, May 2007.

[9] R. R. Kompella, S. Ramabhadran, I. Ramani, and A. C. Snoeren. Cooperative packet scheduling via pipelining in 802.11 wireless networks. In *Proceedings of the Workshop on Experimental Approaches to Wireless Network Design and Analysis (E-WIND)*, pages 35–40, Philadelphia, PA, Aug. 2005.

[10] K. Medepalli, P. Gopalakrishnan, D. Famolari, and T. Kodama. Voice capacity of IEEE 802.11b, 802.11a and 802.11g wireless LANs. In *Proc. IEEE Global Telecommunications Conference*, 2004.

[11] M. Neufeld, J. Fifield, C. Doerr, A. Sheth, and D. Grunwald. SoftMACflexible wireless research platform. *Proc. HotNets-IV*.

[12] G. Paschos, I. Papapanagiotou, S. Kotsopoulos, and G. Karagiannidis. A New MAC Protocol with Pseudo-TDMA Behavior for Supporting Quality of Service in 802.11 Wireless LANs. *EURASIP Journal on Wireless Communications and Networking*, 6:76–84, 2006.

[13] A. Rao and I. Stoica. An overlay MAC layer for 802.11 networks. *Proc. of Mobile Systems, Applications and Services (Mobisys '05)*, 2005.

[14] H. Schulzrinne, S. L. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. RFC 3550, 2003.

[15] A. Sharma, M. Tiwari, and H. Zheng. MadMAC: Building a reconfigurable radio testbed using commodity 802.11 hardware. *Proc. First IEEE Workshop on Networking Technologies for Software Defined Radio (IEEE SECON 2006 Workshop)*.

[16] S. Shin and H. Schulzrinne. Experimental Measurement of the Capacity for VoIP Traffic in IEEE 802.11 WLANs. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2018–2026, 2007.

[17] J. Snow, W. Feng, and W. Feng. Implementing a low power TDMA protocol over 802.11. *Wireless Communications and Networking Conference, 2005 IEEE*, 1, 2005.

[18] P. Wang, H. Jiang, and W. Zhuang. Capacity Improvement and Analysis for Voice/Data Traffic over WLAN. *IEEE Transactions on Wireless Communications*.

[19] W. Wang, S. C. Liew, and V. Li. Solutions to performance problems in voip over a 802.11 wireless lan. *Vehicular Technology, IEEE Transactions on*, 54(1):366–384, Jan. 2005.

[20] W. Wang, S. C. Liew, Q. Pang, and V. O. K. Li. A multiplex-multicast scheme that improves system capacity of voice-over-ip on wireless lan by 100In *ISCC '04: Proceedings of the Ninth International Symposium on Computers and Communications 2004 Volume 2 (ISCC"04)*, pages 472–477, Washington, DC, USA, 2004. IEEE Computer Society.

[21] J. Yu, S. Choi, and J. Lee. Enhancement of VoIP over IEEE 802.11 WLAN via Dual Queue Strategy. *Proc. IEEE ICC*, 4, 2004.