Flashback: Decoupled Lightweight Wireless Control

Asaf Cidon, Kanthi Nagaraj, Sachin Katti
Stanford University
{cidon, kanthicn, skatti}@stanford.edu

Pramod Viswanath
University of Illinois at Urbana-Champaign
pramodv@illinois.edu

ABSTRACT

Unlike their cellular counterparts, Wi-Fi networks do not have the luxury of a dedicated control plane that is decoupled from the data plane. Consequently, Wi-Fi struggles to provide many of the capabilities that are taken for granted in cellular networks, including efficient and fair resource allocation, QoS and handoffs. The reason for the lack of a control plane with designated spectrum is that it would impose significant overhead. This is at odds with Wi-Fi's goal of providing a simple, plug-and-play network.

In this paper we present Flashback, a novel technique that provides a decoupled low overhead control plane for wireless networks that retains the simplicity of Wi-Fi's distributed asynchronous operation. Flashback allows nodes to reliably send short control messages *concurrently* with data transmissions, while ensuring that data packets are decoded correctly without harming throughput. We utilize Flashback's novel messaging capability to design, implement and experimentally evaluate a reliable control plane for Wi-Fi with rates from 175Kbps to 400Kbps depending on the environment. Moreover, to demonstrate its broad applicability, we design and implement a novel resource allocation mechanism that utilizes Flashback to provide efficient, QoS-aware and fair medium access, while eliminating control overheads including data plane contention, RTS/CTS and random back offs.

Categories and Subject Descriptors

C.2 [**Computer Systems Organization**]: Computer-Communication Networks

Keywords

Wireless Network, Wireless Control

1. INTRODUCTION

Due to their distributed and asynchronous nature, Wi-Fi networks, unlike cellular networks, function without a dedicated and separate control plane for node coordination. Wi-Fi uses implicit and explicit control mechanisms that are always coupled with the data plane. For example, CSMA's random back off is a form of implicit coordination between nodes, while RTS/CTS and association requests are explicit control packets that are multiplexed with data transmissions. The lack of a separate and decoupled control plane results in data plane performance problems, and makes it hard to provide desirable features such as QoS, handoffs, access control and power duty cycling. For example, random back offs render packet scheduling inefficient due to collisions and exposed terminals, while explicit control mechanisms such as RTS/CTS consume significant channel time [6, 11, 9, 15].

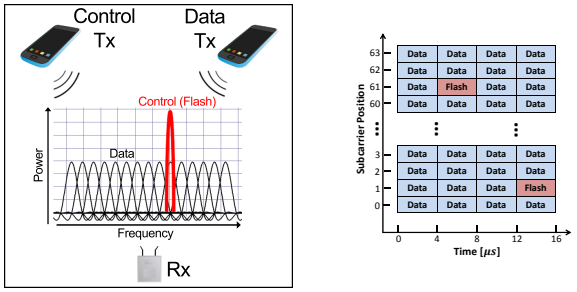

Figure 1: The first figure depicts a node concurrently sending control messages using flashes. The second figure visualizes OFDM packets as a time-frequency grid. Flashes are sent over specific slots on the grid.

In contrast, cellular networks have none of these problems, since they use a separate and decoupled control plane. Cellular networks (e.g. LTE [13, 8]) designate dedicated spectrum for explicit control signaling between the base station and clients. Such a decoupled control plane has many benefits. For example, the base station can leverage the control plane to centrally schedule user's data flows, ensuring both channel efficiency (e.g. no collisions or exposed terminals) and the required QoS for traffic classes (voice, video and data). Several other useful functions, such as client power saving modes and seamless handoffs, are enabled by the separate control plane. However, the cellular control plane adds considerable overhead, because cellular networks have to reserve a significant share of their spectrum for the control channels [13, 8]. Furthermore, since the control channels are accessed using techniques such as OFDMA, which require microsecond level synchronization [1], cellular networks have to rely on accurate centralized time synchronization. Wi-Fi's goal is to be simple, cheap and asynchronous, and therefore the Wi-Fi standards cannot afford to adopt expensive and centrally coordinated mechanisms. This raises a natural question: Can we achieve similar control functionality in Wi-Fi without paying the high overhead of cellular networks, while retaining the simplicity and flexibility of a distributed asynchronous network?

In this paper, we present the design and implementation of Flashback, a novel technique that provides Wi-Fi networks with a *sepa-*

SIGCOMM'12, August 13–17, 2012, Helsinki, Finland.

rate, decoupled and low-overhead control plane, while preserving Wi-Fi's desirable distributed asynchronous nature. The key building block of Flashback is a novel mechanism that allows nodes to send short control messages on the same channel concurrently with data transmissions, without harming the data transmission. Moreover, the control messaging is asynchronous, and does not require precise clock synchronization among all the nodes. Flashback's control messaging adds very little resource overhead since no extra spectrum or time slots are used, yet as our experiments show, provides fast and reliable control plane messaging rates of up to $400Kbps$, with average rates of $175Kbps$. Flashback is general purpose and can be used for a wide gamut of applications, from centralized resource allocation to handoffs and network association.

How can nodes send short control messages concurrently with data transmissions without causing harm to the data? The key insight is to use short high-powered flashes that are localized in frequency and time, which only interfere with a very small number of symbols from the data transmission. When the data receiver detects these flashes, it simply erases the bits that were corrupted from the bits of the data packet. Flashback uses the location of the flashes as a way to modulate bits that represent the control message. To make this clearer, consider the scenario, where a client is transmitting a data packet to the AP using Wi-Fi OFDM. The packet can be visualized as a two-dimensional grid of time and frequency slots, with dimensions equal to the length of the packet and the number of OFDM subcarriers respectively, as depicted by the second picture in Fig. 1. Another client concurrently transmits short flashes that are localized to one of the slots in the time-frequency grid. As Fig. 1 shows, the flash is overlaid on top of one of the subcarriers in the received data packet. In practice, flashing is equivalent to sending a sinusoid that has a time length equal to that of a Wi-Fi OFDM symbol ($4\mu s$), at a frequency equal to the specific subcarrier on which the flash is sent. The client can transform a series of flashes into a small control message (32 bits) by varying the relative location of flashes. The AP can then read the client's message, by detecting the time-frequency slots in which it sees high-powered flashes and transforming them back to the control message.

Flashback exploits two unique properties of OFDM signaling to ensure that data transmissions do not fail due to interference from flashes. First, OFDM divides data into small chunks that are modulated separately on each time-frequency slot. By flashing a small number of specific time-frequency slots, Flashback ensures that any interference is localized and does not corrupt the entire packet. Second, successful packet transmissions invariably have some *link margin*, i.e. the SNR of the received packet is typically greater than the minimum SNR needed to decode the packet. Due to this margin, the loss of a few data bits does not affect decoding: the receiver can simply erase the bits from the slots where the flash was sent and recover the data from the other symbols.

We design and implement Flashback on a Virtex-5 based FPGA software radio test bed, including a full implementation of a $20MHz$ Wi-Fi OFDM receiver and transmitter with convolutional coding. Flashback is implemented by applying minimal firmware modifications to the OFDM PHY receive and transmit chains. Our experimental evaluations show that Flashback supports typical control message rate of about $175Kbps$ and a maximum rate of $400Kbps$ (depending on the environment), while causing minimal impact to concurrent data packets (less than 0.5% overhead). In addition, Flashback's flash detection can be configured so that it rarely misses a flash (less than 1% false negative rate) under almost all channel conditions, ensuring that control messages are received reliably.

The main contribution of Flashback is to exploit its flashing capability to create a novel control plane for asynchronous wireless networks that is decoupled from the data plane. Such a control plane can be used for a variety of network control functions. First, we show how Flashback can facilitate efficient, high throughput and QoS-aware medium access. We then demonstrate that Flashback can be used to design a virtual wireless enterprise network that provides seamless association, flow setup, teardown and scheduling. Finally, we show how Flashback can be used by Wi-Fi clients to efficiently duty cycle.

In order to evaluate these applications we ran trace-driven simulations under several network scenarios. Our simulation results show that Flashback can outperform the traditional Wi-Fi MAC protocols significantly, providing a throughput increase of more than $5\times$ over CSMA/CA and $1.8\times$ over RTS/CTS in congested networks, and 70% energy savings compared to Wi-Fi Power Saving Mode (PSM). Furthermore, Flashback can be used to easily enforce QoS policies. Using Flashback, latency-sensitive traffic (e.g. VoIP) would experience less than $1ms$ of delay even under highly congested networks.

2. OVERVIEW

This section provides an overview of Flashback. We first describe the general motivation for our control channel design. We then provide a short primer on OFDM wireless PHY and SNR link margins, which are fundamental for understanding Flashback's control messaging mechanism. We finally describe an overview of Flashback's design.

2.1 Motivation: Control Plane On Top of the Data Plane

Before we start describing Flashback's design, we would like to answer a simple question: Why does our control plane send flashes on top of concurrent data transmissions? Can't we implement a control plane similar to cellular networks, where spectrum is pre-allocated for a dedicated control channel?

The main reason is that spectrum is becoming scarce in the unlicensed band, especially due to the proliferation of dense wireless deployments. Specifically, if we were to pre-allocate a narrow $2Mhz$ chunk for a control channel, according to Wi-Fi specifications, we would need to reserve an additional $1.875MHz$ as guard band [12]). Consequently, in a $20MHz$ channel, almost 25% of the spectrum would be consumed by the control channel. In contrast, by sending control messages on top of the data plane spectrum, Flashback does not consume any extra spectrum.

Alternatively, we could use a protocol like OFDMA, which lets multiple nodes transmit data and control packets on different frequencies at the same time, without requiring extra spectrum [1]. However, OFDM requires tight synchronization (on the order of a microsecond) among all client nodes. Wi-Fi's current protocols do not support such tight time synchronization. In contrast to OFDMA, our scheme does not require any clock synchronization.

Finally, messages such as RTS and CTS packets, beacons and probes are all designed to implement control functions. However, these messages occupy the network time of the data packets. Furthermore, since these messages have to be delivered reliably, they are encoded using the lowest bitrates and consume significant portions of the channel time. As our own simulations and recent studies have shown [7], such messages can consume up to 40-50% of the channel's time. In addition, this overhead makes it hard for QoS standards like 802.11e [14] to enforce QoS requirements in congested networks, because they multiplex their control messages on the data plane. As a result of the extra overhead, many network managers prefer to disable most of these control functions.

Minimum Required SNR	Bitrate	Modulation	Coding
3.5 dB	6 Mbps	BPSK	1/2
4.5 dB	9 Mbps	BPSK	3/4
5 dB	12 Mbps	QPSK	1/2
9.5 dB	18 Mbps	QPSK	3/4
12 dB	24 Mbps	16-QAM	1/2
17.5 dB	36 Mbps	16-QAM	3/4
21 dB	48 Mbps	64-QAM	2/3
22 dB	54 Mbps	64-QAM	3/4

2.2 OFDM Primer

OFDM is a multi-carrier wideband modulation technique. OFDM divides the entire bandwidth available for transmission into evenly spaced orthogonal bands, which are called subcarriers. The data is split into parallel streams, one for each subcarrier. Specifically, in Wi-Fi, a 20MHz channel is split into 64 subcarriers, each spaced 312.5KHz apart.

To modulate data onto OFDM transmissions, a single constellation symbol is sent on each subcarrier of an OFDM symbol, depending on the bitrate being used (BPSK, QPSK, etc.). The constellation symbols are passed through an IFFT to produce time domain samples, which are then sent over the channel. The typical OFDM symbol length is $4\mu s$. OFDM symbols are sent one after the other. The symbols can therefore be pictured as a two-dimensional time-frequency grid, where each frequency index represents a single subcarrier, and each time index represents a period of $4\mu s$, as depicted by Fig. 1. Packets can be visualized as constellations being mapped to different points on this grid. This abstraction applies to any OFDM based system, including Wi-Fi, LTE and WiMAX.

The Wi-Fi OFDM transmitter sets the constellation modulation and channel coding rate according to the channel conditions, in order to protect against channel distortions such as noise and interference. For example, a dense constellation such as 64-QAM is less robust to errors than a sparser one like 16-QAM. Similarly, Wi-Fi transmitters can increase the data redundancy, by controlling the channel code rate. Wi-Fi has three different coding rates (1/2, 2/3, 3/4), where the rate is defined as the ratio of data bits to actual encoded bits that are sent. Thus, the lower the coding rate, the higher the redundancy.

To maximize throughput, Wi-Fi transmitters perform rate adaptation, i.e. they estimate the channel SNR to the receiver, and for that SNR, pick the densest constellation and highest coding rate that still ensure that the receiver can decode. Table 2.2 shows different combinations of Wi-Fi constellations and coding rates and their corresponding SNR thresholds, as we measured on our Wi-Fi receiver. SNR measurement in current networks is based on packet loss estimates, i.e. if a packet sent at a particular rate is lost, then the sender assumes that the channel SNR is lower than the threshold for that rate, and consequently the sender switches to a lower rate. There are more sophisticated techniques to measure SNR that tradeoff increased measurement overhead for better accuracy.

2.3 Link Margin

The previous discussion raises two observations. First, since SNR fluctuates over time on the order of milliseconds, SNR measurement is inherently error-prone. Consequently, even the most accurate SNR estimation techniques cannot adapt to instantaneous SNR variations, and therefore bitrate adaptation algorithms prefer to err on the conservative side. Algorithms minimize the risk of packet loss by using a lower bitrate than the threshold permits, since packet loss is expensive. Second, to simplify the implementation of Wi-Fi hardware, the bitrates available for transmission are discrete. Therefore, in order to protect against the uncertainty in measuring SNR, and due to the discreteness of bitrates, for every successful

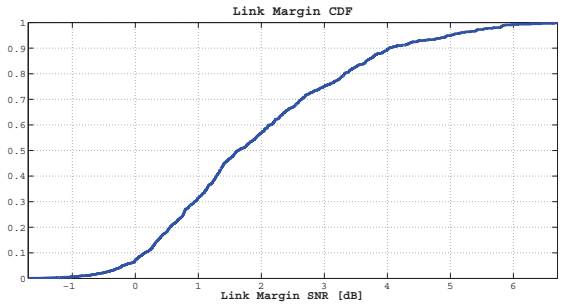

Figure 2: The CDF of the link margin between the actual SNR measured by the channel sounder, and the SNR threshold chosen by SoftRate's rate adaptation algorithm.

Wi-Fi transmission there is some *link margin*. Link margin is the difference between the instantaneous channel SNR and the minimal SNR required to decode the packet. The link margin's value is dependent on the accuracy of the SNR measurement, how conservative the rate adaptation protocol is, the available discrete bitrates and the amount of external interference.

To measure the link margin in a realistic setting, with varying channel SNRs and a practical rate adaptation algorithm, we perform the following experiment. We use RUSK Stanford Channel Sounders [19] to measure a $20MHz$ channel on the $2.4GHz$ ISM band. One sounder is fixed, while the second one is moved at walking speed. Next, we simulate the rate adaptation decisions a state-of-the-art rate adaptation algorithm, SoftRate [20], would make if it were making decisions for the transmitter based on the channel trace. For each bitrate SoftRate picks for transmitting a packet, we measure the difference between the actual channel SNR (from the trace) and the SNR threshold required for the packet to be decoded. We collected a trace of about $70,000$ packets from our sounders. Fig. 2 plots the CDF of these margins, measured over the packets that were transmitted successfully.

The measured median link margin is $1.63dB$ and the average is $1.94dB$. In other words, typically there is a considerable amount of slack between the data rate used by the transmission and the data rate the channel could actually support. Furthermore, the estimate that we measured is most likely conservative. In commercial systems, Wi-Fi transmitters are configured to use higher link margins, due to interference from legacy radios (e.g. microwave ovens) or neighboring APs. Note that the link margin is negative for about 7% of the packets. For these packets, even though SoftRate chose a bitrate that was too high for the SNR of the transmitted packet, the algorithm 'got lucky', and the packet was decoded successfully.

2.4 Flashes: Flashback's Control Signals

The goal of Flashback is to provide a control plane for asynchronous wireless networks with negligible overhead. To this end, Flashback requires a messaging technique, which would allow nodes to concurrently send control messages without harming the ongoing data transmission, and without requiring any synchronization.

Flashback exploits the link margin to create the desired control messaging mechanism. The basic idea is that since typical transmissions have some link margin, any node could send a control packet even while a concurrent data transmission is taking place, as long as the interference from the packet is smaller than the tolerable margin.

However, simply sending regular Wi-Fi packets as control messages on top of concurrent data packets is doomed to fail for two reasons. First, the receiver needs to have a way to detect and decode the control packet. Since the power of the data packet would

typically be much stronger than the control packet, it would be difficult to reliably determine whether a control packet has actually been sent, if it is limited to a signal that is weaker than $1.94dB$ on average. Second, the amount of interference that the control packet would cause is a function of both the power at which it is transmitted, as well as the channel between the flash transmitter and the receiver. To ensure that the interference stays within the required tight SNR margin, the control transmitter would have to precisely estimate the channel and set its transmission power to meet the link margin requirements. Given that the uncertainty involved in link SNR estimation is the raison d'être of link margins, it would be risky to rely on high precision SNR estimation to ensure that the margin is met.

Flashback's insight is to utilize the fundamental properties of OFDM modulation to concurrently send easily decodable control messages, while limiting the interference of the control messages to a level below the link margin. Flashback leverages the time-frequency abstraction of OFDM to localize the interference the control messages cause to the data packets. Instead of sending a regular OFDM packets over the entire band, the control message transmitter sends *flashes*. Flashes are simple sinusoids that have a frequency equal to the frequency of a specific subcarrier, and duration equal to the OFDM symbol time. The flash appears as a power spike, localized to a specific point in the time-frequency OFDM grid, corresponding to the frequency of the flash sinusoid and the time slot in which the flash was transmitted. Since the flash transmitter focuses all its power on a single subcarrier, in most cases the flash has much higher power relative to the data transmission symbols. Consequently, the flash's position on the time-frequency OFDM grid is easy to detect using a simple peak detection algorithm. Note that Flashback doesn't synchronize the flashes to the data symbols, so there might be time and frequency offsets between the grids of the transmitter and the receiver. We address these later in our design.

In order to decode the data transmission, the receiver first detects the flashes, and then instead of reading the erroneous bits from the symbols that were flashed, the receiver *erases them*. In other words, instead of causing bit errors due to concurrent control messaging, Flashback exploits the fact that flashes are easily detectable to convert them to bit erasures. Channel codes can correct twice as many bit erasures as bit errors [16, 18]. An additional advantage of flashing is that it causes bit erasures in short bursts, and wireless channel codes are designed to handle bursty interference. Due to all these factors, as long as the number of flashes is limited, a receiver can use simple algorithms to decode both the data and flash transmissions concurrently.

How can nodes leverage flashes to send control messages? The basic idea is to use the relative subcarrier position of the flashes as a way to signal a small set of bits. For example, a transmitter can send two consecutive flashes at subcarriers 2 and 27 and use the difference (25) as the means to encode information. Using the relative distance between subcarriers has the advantage of being robust to frequency offset errors. In Sec. 3.2 we describe a simple and robust modulation scheme that uses the relative subcarrier distance idea to design a control messaging algorithm. We also describe a protocol that mediates which node can send a control message at any point.

While this discussion provides the basic intuition behind Flashback, several practical questions remain.

- How many flashes can a Wi-Fi data transmission tolerate? How does the link margin relate to the number of allowed flashes?
- How does the receiver reliably detect flashes and decode control messages? How do we prevent the high powered flashes from saturating the ADC?
- How do we ensure that the number of flashes is under the tolerable limit when multiple nodes want to send control packets? What is the protocol for accessing the shared control plane?
- How can the MAC layer leverage this new decoupled control channel?

In the next section, we describe the design of Flashback that addresses these questions.

3. DESIGN

In this section we describe the control plane's three components: the basic flashing transmission and detection mechanisms, the technique that transforms flashes into short binary control messages, and the protocol for nodes to asynchronously access the control plane. We also describe the minor changes the data plane receiver has to implement to decode concurrently transmitted data packets.

Before we proceed, we define several notations for the rest of the paper. We assume that nodes use Wi-Fi OFDM for data packet transmission with 64 subcarriers spread over a $20MHz$ channel, where each OFDM symbol lasts $4\mu s$. As discussed before, OFDM can be abstracted as a time-frequency coordinate system (i, j), where $0 \leq i \leq 63$ identifies the subcarrier and j is the time slot. To send a data packet, a node fills these coordinates with different constellation samples. The magnitude of the constellations is a function of the transmit power P with the following constraint:

$$\frac{\sum_{i=0}^{63} \sum_{j=1}^{L} |x_{ij}|^2}{L} \leq P \tag{1}$$

Where x_{ij} is the complex symbol transmitted on the (i, j) coordinate, and L is the number of symbols in the packet.

We define R, as the maximum number of flashes that all the nodes can send per second over the network, without harming the performance of the data transmission (i.e., without introducing more than 1% of errors). The value of R is a function of the link margin, which depends on the PHY's rate adaptation algorithm, as well as the accuracy of the flash detection algorithm. In our experimental evaluation we found that for Wi-Fi, even a conservative approach that minimizes flash interference with a state-of-the-art rate adaptation algorithm, allows Flashback to use a flash rate R of *50,000 flashes per second.*

3.1 Flashing

Flash transmission is simple to implement. Using the OFDM coordinate abstraction, a flash transmission amounts to transmitting a complex constellation f on a specific co-orindate (i, j), and zeroing all other coordinates on the same time slot (an entire column in the grid). In practice, this requires the node to send a single sinusoid that lasts for $4\mu s$ with frequency corresponding to the i'th subcarrier. The actual time domain message is generated using the standard OFDM modulator (i.e. it is passed through an FFT and the rest of the transmission chain). Flash transmission therefore requires no extra transmit hardware or signal processing logic.

The magnitude of the flash f is larger than the magnitudes used for sending data symbols. To see why, consider the power constraint in Eq. 1. Since for a given j, the node sends zeros on all of the i coordinates except one, the power constraint will be satisfied even if the magnitude of the flash is much higher than the one used for data symbols. The higher magnitude aids detection, as explained in the next subsection. In our current implementation, flashes transmit at $64\times$ greater power than the power used for individual data symbols. Prior work has made a similar observation on how using higher magnitude signals with narrower width Wi-Fi channels helps improve range [5].

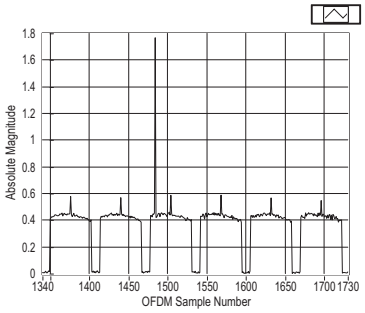

Figure 3: The graph contains a sequence of frequency-domain samples taken from a measurement conducted on our Flashback implementation. The flash is simple to discern from the data packets.

3.1.1 Flash Detection

The relatively high magnitude of flashes relative to data symbols makes them typically easy to detect. This is demonstrated by Fig. 3 where the magnitude of a single flash is clearly visible against a backdrop of regular OFDM data symbols. Flashes are of course even easier to detect when the channel is already idle. Using the same time-frequency OFDM abstraction, if the receiver sees a peak value in one of the time-frequency slots, it declares that a flash is detected in that slot. To reliably detect the flash and reduce false positives that can be caused by practical effects such as frequency-selective fading, Flashback uses a technique that uses the differential rather than the absolute value of the complex symbols.

Specifically, the receiver computes the following equation at every (i, j) coordinate:

$$D(i,j) = \frac{\sum_{m=-1}^{1} \sum_{n=-1}^{1} |(|y(i,j)| - |y(i-m, j-n)|)|}{8} \quad (2)$$

Where $y(i, j)$ is the constellation value received on coordinate (i, j).

The receiver detects a flash at coordinate (i, j) i.f.f. $D(i, j) > T$, where T is an empirically determined threshold. In effect, the equation above is computing the average differential value at (i, j). D will be very high when there is a flash, compared to a normal data symbol. The detection threshold T depends on practical system parameters such as receiver dynamic range and receiver gain.

There are a couple of practical issues with this detection algorithm. First, in order to simplify the implementation, we assumed that the flash transmitter does not synchronize to the packet of the data transmitter. Therefore, their respective time-frequency grids won't align perfectly. However, the receiver will synchronize to the data transmitter's grid since its primary goal is to decode the data packet. Practically, this means that a flash might span two time-slots at the receiver. Since the flash is now split, the flash's constellation value might be half the maximum value. However, since flashes are sent at $64\times$ the data symbol power, they are still likely to be received at a much higher power magnitude than the value of the data symbol and should be easy to detect. Performance can be further improved if the receiver runs two parallel flash detectors offset from each other in time by half a time slot, i.e. $2\mu s$. Intuitively, one of the detectors will usually have a better alignment with the flashing node, and the receiver can pick the grid with the higher flash value, and send its corresponding slots to the flash decoder. Second, in order to reliably detect flashes, we empirically measured that the receiver requires the flash power to be at least $6dB$ higher than the power of the concurrent data packet. In some cases, this might cause flashes by a node that is far away from the AP to be 'drowned' by nodes that are closer to the AP, and have significantly higher link SNRs. To mitigate this issue, the nodes with low channel SNRs might need to delay their flashes until the data channel is not occupied by data packets with high SNRs (e.g. packets with a high bitrate).

3.1.2 Data Packet Decoding

Flashback requires minimal changes to data packet decoding in the presence of concurrent flashes. As discussed before, the receiver erases the data symbols that are present at the same coordinates as the flashes. Practically, the data decoding PHY is instructed to ignore the demodulated bits for those symbols, and inform the decoder that these bits have been erased. It is known that erasures are easier to handle than bit errors, and require half the redundancy to correct [16, 18].

An important benefit of this scheme is that we do not have to change the existing OFDM PHY decoding hardware for decoding data in the presence of flashes, since soft Viterbi decoders support erasures.

3.2 Messaging with Flashes

The goal of the modulation algorithm described below is to communicate a series of bits that represent the control message, and communicate it to the receiver using a sequence of flashes. Note that Flashback offers a generic short messaging system that we can use for implementing any form of control functionality. In this section, we will describe a technique that provides control messaging rates of $175Kbps$ assuming that the maximum flashing rate is $50,000$ per second. However, as well see in Sec. 5 depending on interference conditions, nodes can flash at rates as high as $100,000$ per second, and the scheme below can be appropriately modified to provide a messaging rate of $400Kbps$.

We make three assumptions for the flash modulation algorithm:

- Flashback cannot assume any synchronization between nodes. Specifically, we cannot assume that the time-frequency grids of the data transmitter and the flash transmitter are tightly aligned.
- At any point, only one node can send a flash-based control message on the control plane. This assumption simplifies the messaging design as we show below.
- Nodes have the ability to sense whether the AP is currently transmitting a data packet. The reason for this requirement is that we assume the AP is half-duplex, and therefore any flashes that would be sent while the AP is transmitting cannot be decoded by the AP.

Flashback's modulation scheme transforms a 32 bit control message into a series of flashes, by varying the distance between subsequent flashed subcarriers. We also add an 8-bit CRC, which results in a total message size of 40 bits. Control messages are a series of 9 flashes sent one after the other in fixed time intervals of $20\mu s$. Using a fixed time interval between flashes simplifies the scheme, and helps other nodes sense that a flash message is being sent. The scheme also ensures that on average, nodes do not exceed R flashes per second.

For this entire section, assume that we do not flash on any pilot subcarriers, zero padded subcarriers, as well as the subcarriers adjacent to them. This leaves us with 36 subcarriers, which we logically number from 0 to 35. In addition, the first flash in the control message signals the start of the message, and is always sent on a pre-designated subcarrier (34). We also do not flash one subcarrier on either side of the start-of-message subcarrier (i.e. $33 - 35$).

The relative distance between two consecutive flashed subcarriers is used to encode the control message. We use the relative distance between consecutive flashes, rather than the absolute position of each one of the subcarriers, because in practice the carrier

frequency of the flashing node could be slightly offset relative to the receiver's. Hence a messaging technique that uses the relative positions of the flashes will be robust to carrier frequency offset (CFO).

We take the bits that comprise the control message, and convert the binary number they represent into a number composed of 8 digits using base-32. A 40 bit message can always be represented using 8 digits in base-32. For example, assume the binary message is represented by $x_1, \ldots, x_8$ where x_i is the i'th digit in the base-32 number. Therefore, if the start-of-message flash was on subcarrier f_1, then the first flash (after the start-of-message flash) would be flashed on subcarrier $f_2 = (f_1 + x_1) \mod 32$, the second on subcarrier $f_3 = (f_1 + x_1 + x_2) \mod 32$ and so forth. To decode the message, the receiver detects flashed subcarriers $f_1', f_2', \ldots, f_9'$, and calculates the values of $x_1, \ldots, x_8$. For example, the formula for x_1 is given by $(f_1' - f_2') \mod 32 = x_1$. The same formula is applied to the other flashes. After it recovers the digits in base-32, the receiver can then read the message by converting the base-32 number into its binary representation. Finally, it checks the message's integrity using the 8 bit CRC.

Therefore, using our very simple coding scheme, Flashback can send on average 32 bits every $180\mu s$, which yields an overall bitrate of about $175Kbps$. 32 bit messages can be used for sending short control requests. For example, the first 10 bits in each message can be used to encode the sender's ID, and the last 22 bits can be used for specifying the contents of the control message (e.g. flow identifier, QoS request or a network association request). While $175Kbps$ is low by data transmission standards, our experiments demonstrate that this rate is sufficient support a practical Wi-Fi control channel.

We chose this simple scheme for a couple of reasons. First, it ensures that there is a fixed interval of $20\mu s$ between consecutive flashes, and as a result maintains a fixed flash rate, consistent with $R = 50,000$ flashes per second. Second, by using the relative distance between flashes, the scheme is robust to CFO errors. Finally, it provides receivers a simple mechanism to detect errors, since they know that a flash should be expected only once every $20\mu s$. In practice, we find that flash detection is fairly accurate, and has low false negative and negligible false positive rates. Hence the previous technique performs quite well.

3.3 Protocol for Sending a Flash

Flashback uses a carrier sense and random back off approach to regulate which node flashes at a particular time slot. Each node, before it flashes, performs a random back off, and counts down the back off counter. Meanwhile, it carrier senses for the *start-of-message flash subcarriers* for any other node flashing. If a flash is detected, it stops the countdown and restarts the counter after waiting for $180\mu s$, which is the time it takes to send a control message. If no flashes are detected and the counter goes to zero, then the node flashes the message. The average value of the random back off is set so that nodes do not exceed $R = 50,000$, the maximum number of flashes per second.

The protocol described above has similar problems to standard carrier sense based MAC protocols. However, unlike traditional Wi-Fi protocols, Flashback effectively exports the contention from the data plane to the control plane. Therefore, control plane access problems such as collisions, congestion and hidden nodes do not directly affect the overall performance of the data plane. This architecture is similar to most cellular control planes, where medium access in the data plane is centrally scheduled, while the control plane access is based on contention [13, 8].

Note that nodes can flash only when the AP is not transmitting, assuming the AP is half-duplex. Therefore, nodes have to decode the data packets of data transmitters, in order to verify whether that the AP is not transmitting a message. To acknowledge the control messages, the AP piggybacks on the regular ACK it sends after receiving a data plane transmission. This is an optimization; the AP could also have used Flashback to send the control message ACK.

3.4 Practical Issues

3.4.1 Interaction with Neighboring Networks

A practical concern is whether Flashback may interact destructively with neighboring networks, if they happen to be transmitting on the same Wi-Fi channel (this problem usually occurs in very dense networks). For example, an AP might detect flashes from a neighboring network and obtain incorrect control information. Further, flashing might hurt data plane transmissions on a neighboring network, since the overall flash rate might be higher than the limit picked per network.

Flashback's flash encoding technique naturally combats the problem of neighboring APs mistakenly decoding flash messages not intended for them. Since each control message provides 32 bits, several bits at the start of the message can be used to signal the identity of the sending node. As long as the identity spaces don't overlap, each AP should be able to tell whether the control message originated from its own network.

The second problem of multiple nodes concurrently flashing is largely avoided because of the "start-of-message" flash, and due to the fixed length of control messages. Since before sending the control message flashes, nodes always send a start-of-message flash on special subcarriers (40), any node sensing it will refrain from flashing for $180\mu s$. This applies to nodes across networks too. Of course there are corner cases where such sensing fails, as in any carrier sense mechanism.

Another concern is whether Flashback interacts poorly with narrowband protocols like Bluetooth or Zigbee. Since we designed Flashback to transmit flashes on single Wi-Fi subcarriers (i.e., $312.5KHz$), Bluetooth and Zigbee, which occupy a much wider bandwidth ($1MHz$ and $2MHz$ respectively), do not interfere with the correct decoding of Flashback control messages.

3.4.2 Size of Flash Messages

The messaging system described above assumes packets are long enough to accommodate the nine flashes that are needed to encode a 32 bit message. If we assume a flash rate of $50,000$ flashes per second, it translates to one control message every $180\mu s$. However, Flashback cannot send flashes while the AP is transmitting, since the AP won't be able to decode the control message. Therefore, the flashing node detects if the AP starts transmitting in the middle of sending a flash message. If so, it abandons the incomplete flash message and retries to send it at the next available opportunity.

3.4.3 Automatic Gain Control (AGC)

Another concern is that flashing may interact negatively with automatic gain control at the receiver. Specifically, the AGC tries to exploit the full dynamic range of the ADC, by automatically amplifying the incoming signal by the right amount, so that it fits the entire ADC range. However, when a flash arrives, since its value will be higher than the normal data transmission, the ADC can get saturated. In order to tackle this problem, we use static gain control. Specifically, our current system uses a 14 bit ADC that provides $86dB$ of dynamic range. Assuming that in the best case, we will see data transmissions with an SNR of $35dB$ ($\tilde{6}$ ADC bits), we can set the static gain so that the best case data message will consume

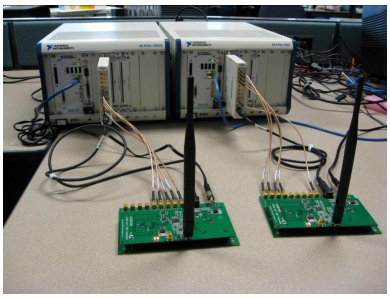

Figure 4: The National Instruments software defined radios we used to implement Flashback and carry our experiments.

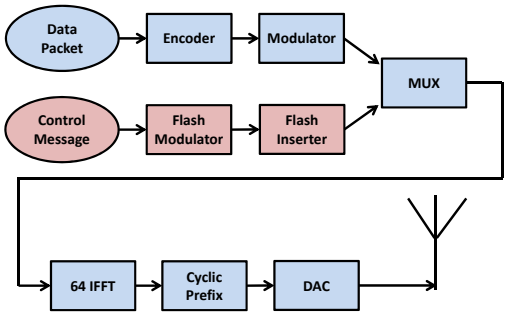

Figure 5: Flashback transmitter implementation.

8 bits of ADC resolution (leaving us with a margin of 2 extra bits). This leaves 6 bits or $36dB$ for decoding flashes, which is more than sufficient in practice.

4. IMPLEMENTATION

We implemented the Flashback receiver and transmitter using Virtex-5 LX30 FPGA based software radios from National Instruments [4], shown in Fig. 4. The FPGA has basic LUT-FF and 32 DSP48E arithmetic unit resources. The Flashback receiver was implemented on two units: the 7695 FPGA implements the FFT logic of the receiver chain, while the flash detection and decoding is performed on a real-time processor (NI PXIe-8133 RT Module). The FPGA is connected to an NI 5781 Baseband Transceiver, which serves as both the ADC and the DAC. The ADC has 14 bits of resolution, and the DAC has 16 bits. We used the National Instruments LabVIEW system design language [3] to implement and debug Flashback on the programmable radios.

4.1 Flash Transmitter Implementation

Fig. 5 presents the implementation of Flashback's transmitter. We added two blocks to the standard OFDM transmit chain: the flash modulator and the flash inserter. The flash modulator transforms a 32 bit binary number into a series of 9 flashes on the time-frequency grid. It does this, by adding an 8 bit CRC to the 32 bit binary sequence, and converting the entire sequence into eight 32-base numbers. The flash inserter then encodes the message using the technique described in Sec. 3.2 into a series of 9 flashes, adds the flashes generated by the encoder onto the transmission in the appropriate slots and zeros out the rest of the slots. Other than these two blocks, the Flashback transmitter is identical to a Wi-Fi transmitter. When a node wishes to flash, it switches on the flash transmitter logic, and when it has to transmit regular data messages, it can simply turn the Flashback logic off.

4.2 Flash Receiver Implementation

The implementation of the receiver is shown in Fig. 6. For the

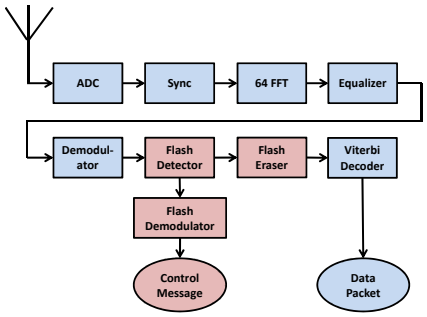

Figure 6: Flashback receiver implementation.

receiver, we added three blocks to the receiver chain: the flash detector, flash demodulator and flash eraser. The flash detector is always turned on by the receiver, because flashes can be received at any time.

The flash detector was implemented by utilizing the simple peak detector described in Sec. 3.1.1. The peak detector runs twice over the same samples, by reusing the same DSP hardware blocks of the OFDM receiver. The first peak detector is synchronized with the data packet, and the second one is offset from the first one by half a time slot, i.e. it is synchronized to $2\mu s$ after the first peak detector. If we detect a flash in the first flash detector, we only erase one subcarrier from the data packet. If we detect a flash in the offset flash detector, we erase the flashed subcarrier from the two data symbols it affected. We found that this 'aggressive' flash detection technique improves our flash detection error rates while reducing data errors.

The flash demodulator is very similar to the transmitter's flash modulator: it transforms the time-frequency flash grid into a binary sequence by converting a base-32 number into a binary number. Before sending the data into the Viterbi decoder, the flash eraser sets the confidence level of all the flashed bits (i.e. bits belonging to the flashed subcarriers) to 0. This signals to the Viterbi decoder that it should erase these bits from the data stream.

5. EXPERIMENTAL BENCHMARKS

In order to test Flashback, we benchmark Flashback's performance with two metrics.

Maximum Flash Rate (R): The maximum number of flashes per second that can be sent without harming data throughput.

Flashback Detection Accuracy: The false positive and false negative rates of detecting flashes at the receiver.

5.1 Maximum Flash Rate

As we discussed in Sec. 2, the link margin depends primarily on two factors: the discrete bitrates and the conservative SNR estimation of the bitrate adaptation algorithm. In practice, it also depends on a third factor, which is the level of interference. Bursty external interference will hurt the SNR estimation of the bitrate adaptation algorithm (i.e., the algorithm will interpret the interference as noise), which will cause the algorithm to behave more conservatively and increase the link margin.

We experimentally evaluate the impact these three factors on the maximum flash rate. All experiments were conducted during nighttime at our lab on the Wi-Fi band of $2.48Ghz$. In order to estimate the SNR for the data packets, we sent known packets. We then estimated the channel's noise by subtracting the known symbols from the received symbols. In order to estimate the effect of interference, we generated a trace of Zigbee and Bluetooth and added it as noise to our channel sounder's trace.

Discrete Bitrates: Our experimental setup consists of three nodes.

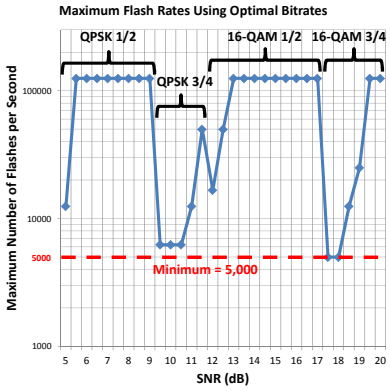

Figure 7: Number of flashes Flashback can support for different SNRs, with 'oracle' bitrate adaptation, i.e. when the transmitter is sending at the optimal bitrate for our receiver. In a real network setting, the transmitter will typically transmit data at a lower bitrate than optimal, and Flashback will be able to support a higher flash rate than the rates depicted here.

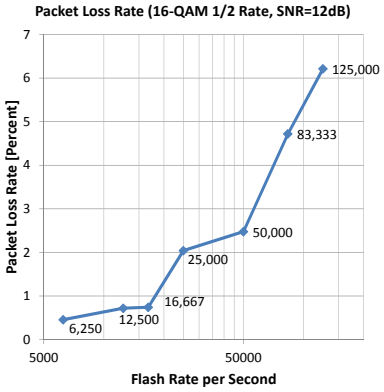

Figure 8: Packet loss rate of a $12dB$ link, as a function of the flash rate.

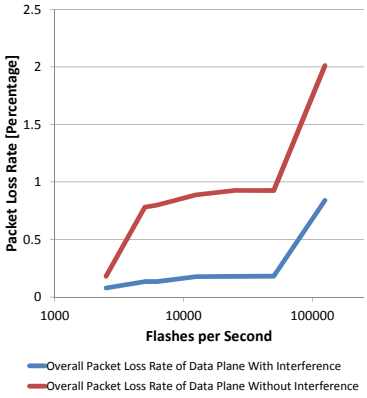

Figure 9: Overall packet loss rate as a function of the flash rate, measured by applying the SoftRate bit adaptation algorithm on a $75,000$ packet trace collected by our channel sounders.

The first node is a regular data transmitter. It keeps sending Wi-Fi packets with a fixed length of 40 symbols. The second node is the flash transmitter. The flash transmitter continuously transmits flashes at a specified rate at the same power. The third node is the receiver. The receiver decodes both the data packet and the flashes. In our experiments, the flashes are received at $6-10dB$ higher than the data transmission. The location of the data transmitter is varied to obtain different SNRs. For each SNR, we first tried transmitting hundreds of packets at all the bitrates, and picked the bitrate which maximizes the throughput for the data transmitter. Next, we varied the flash rate, and picked the highest rate, which doesn't affect the throughput of the data transmitter by more than 1%. Fig. 7 plots the maximum flash rate as a function of the SNR measured between the data transmitter and receiver, when the transmitter is transmitting at the most optimal discrete bitrate (i.e. this is an oracle transmitter; it can estimate the exact SNR for every packet ahead of time).

Analysis: As we can see from Fig. 7, since we are using an oracle transmitter that utilizes perfect SNR estimation, the channel can only support a maximum of $5,000$ flashes per second, because it is the minimum of all the flash rates across all SNRs. However, since the channel SNR is dynamic, and the bitrate adaptation algorithms cannot estimate SNR perfectly as we will show below, we can typically support much greater flash rates of $50,000$ flashes per second.

An interesting feature of the graph is that it looks like a 'chainsaw'; it starts low when Flashback switches to a new bitrate, and increases when the SNR is increased. The reason we see the chainsaw effect, is that every time Flashback switches to a higher bitrate, the data plane loses some redundancy in its channel code, since it is trying to send more data on a given channel. Since there is less redundancy, the flash eraser cannot erase as many bits as it could before, and Flashback can only support a lower flash rate. However, when the SNR increases, the message is received with fewer errors at the receiver, and the Viterbi decoder can tolerate more bit erasures due to flashes. As an aside, it is likely that the line in Fig. 7 could be increased beyond $125,000$ flashes a second at high SNRs. In our experiments, we did not attempt to flash at higher rates than $125,000$, since we only have one flash transmitter.

To further explain the previous results, we focus on one experiment where the SNR of the data transmission is $12dB$ at the receiver. For this SNR, the data transmitter uses a 16-QAM, 1/2 rate encoding for its packet to maximize throughput. Note that this SNR lies right at a transition point for data bitrates. For a slightly lower SNR of $11dB$, the data transmitter would have picked QAM, 3/4 rate encoding. We vary the flashing rate from the flash transmitter and measure the packet loss rate for the data transmission. The results are presented in Fig. 8. As expected, as we increase the flash rate, the packet loss rate of the data packets increases. The reason is that the increasing number of flashes causes the receiver to erase more bits from the bit stream. If the receiver erases too many bits, it will start seeing errors in the Viterbi decoder.

Note that even at the most minimal flash rate, there is a small probability of packet loss (around 0.45%), which most probably occurs because sometimes a flash can interfere with the packet header itself. In a practical system this would be avoided, since the flashing node could utilize the carrier sense mechanism to ensure it does not interfere with the header.

Conservative Bitrate Adaptation: Bitrate adaptation algorithms have to choose the transmitted bitrates conservatively, because of the difficulty in measuring channels under time-varying contention-prone environments. This increases the link margin in practical settings. To evaluate this property, we need to conduct benchmarks on a network where there are time-varying environmental conditions and multiple contending nodes. However, we cannot conduct such experiments with our software radios, since they are large static nodes (as we can see in Fig. 4) and too expensive for emulating an entire network. Hence, we resort to trace-driven experimentation.

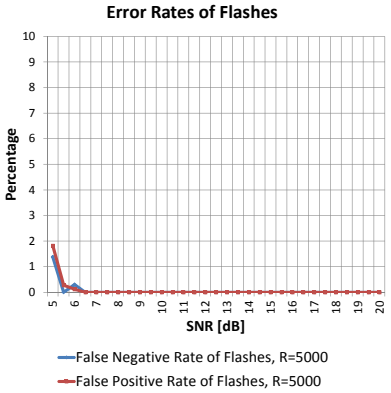

Figure 10: The false negative and positive rates of Flashback when received at roughly $6dB$ higher than the data packets.

We use the traces collected by the RUSK channel sounder, which we described in Sec. 2, to simulate time-varying and contention conditions, and a state-of-the-art bitrate adaptation algorithm, SoftRate [20]. We simulate a Wi-Fi transmitter sending packets over the channel trace, and collect the bitrate decisions SoftRate makes. We then vary the flash rate and measure the effect on the data throughput, by applying the packet loss rates we measured on our Flashback implementation. We also added interference by generating a trace of Bluetooth and Zigbee signals, received at SNRs between $3dB$ and $10dB$.

Fig. 9 plots the overall packet loss rate of the data plane as a function of the flash rate. Without interference, Flashback can send $50,000$ flashes per second. This number is 10 times higher than the number of flashes per second when Flashback was using the optimal discrete bitrate for each SNR. With external interference from sources such as Bluetooth and Zigbee radios, Flashback can send flashes at a rate as high as $100,000$ flashes per second. The reason is that rate adaptation algorithms cannot accurately estimate the channel SNR and interference, and are forced to make conservative bitrate decisions to avoid packet loss.

As we can see, the right flash rate depends on the environment Flashback operates in. In settings where contention, external interference and time-varying conditions are common, Flashback can transmit at a flash rate of $100,000$. However, if the environment is calmer (e.g., a home with several Wi-Fi clients), then Flashback supports a lower flash rate of $50,000$. These correspond to data rates of 400Kbps and 175Kbps respectively. By default, we assume Flashback has a data rate of $175Kbps$.

5.2 Flash Detection Accuracy

Next, we investigate the accuracy of flash detection. We sent flashes at $6dB$ on a known subcarrier over a varying data plane, while modifying the SNR of the received data packet. At each SNR, we calculated the false negative (number of flashes missed by the receiver) and false positive rate (flashes that the receiver detected which weren't actually sent). As depicted by Fig. 10, the error rates are very low for all SNRs. The false negative and positive rates are below 0.1% for all SNRs other than in the range of $5-6.5dB$. This is probably due to the fact that in low SNRs, there is more variation among the power of the different subcarriers and it is difficult to correctly detect the flash.

In the rare occasions that the flash detector produces false positives, it is always on a subcarrier adjacent to the real subcarrier that was sent by the flash transmitter. These false positives are caused by the carrier frequency offset between the flash transmitter and the receiver. This is why the flash messaging technique uses the relative distances between subcarriers to modulate the message, and is therefore resilient to such false positives, as described in Sec. 3.2.

6. APPLICATIONS

Flashback provides a generic decoupled control plane that can be used to improve data plane performance for a variety of applications. The decoupled control plane allows Flashback not only to significantly improve existing scheduling protocols like RTS/CTS and CSMA/CA, but also enable novel applications. This section presents how Flashback can be used to significantly improve existing Wi-Fi MAC protocols. We then show how Flashback enables a virtualized enterprise Wi-Fi network with better handoffs, duty-cycling, load and interference management.

6.1 Flashback-MAC

Flashback Šs control plane enables us to eliminate almost all of the control overhead in the current Wi-Fi MAC protocol, which uses CSMA/CA or RTS/CTS to mediate medium access. In contrast with these schemes, nodes use Flashback Šs control plane to notify the AP of their outstanding transmission requests, while the AP is solely responsible for determining the schedule for the entire network. We describe this simple and generic MAC protocol, which we call Flashback-MAC, and explain how it eliminates a number of chronic problems associated with the current Wi-Fi MAC protocols.

The **Demand Map** is the key novel abstraction that Flashback provides the AP to determine the schedule for the entire network. The demand map is a list of all outstanding transmission requests from all nodes in the network. The list is sorted in order of the arrival of the requests. Each request is associated with a flow identifier at that node, and is also annotated with the SNR of the node requesting a transmission opportunity. Each request can also include a few extra annotations, including bits that signify QoS requests, such as latency or throughput sensitive traffic. Note that the demand map also keeps track of traffic that the AP has to transmit to various nodes.

The demand map for uplink traffic is constructed using flashes. Whenever a node has a packet queued for transmission, it flashes a short control message to the AP. The message contains a flow identifier, and the corresponding amount of outstanding traffic and any associated QoS requests. Our current implementation allows two different classes: latency sensitive and regular data. Latency sensitive traffic is for interactive applications such as VoIP. The first 10 bits in the control message are used to signal the identity of the flashing node, the next 4 are reserved for the flow identifier, the next 8 for outstanding data in multiples of $100B$ for that flow, the next 8 for specifying deadlines, and the final 2 for specifying the QoS level. There are certain bits reserved in the ID and flow identifier fields for association requests for new nodes that wish to join the network.

The control message is sent using the control plane protocol just described. The AP decodes the control message, and also estimates the approximate SNR of the flashing node from the flash message itself. Whenever the AP has the opportunity to send a packet (either data or ACK), it piggybacks a short message at the end of those packets signaling the receipt of the control messages. Hence, if a node does not see an acknowledgement it can resend the control message using flashes.

The demand map decouples the medium access mechanism from the scheduling policy, i.e. it allows the AP to flexibly use any scheduling algorithm on top of the demand map. To demonstrate the demand map's flexibility, we simulated two simple and generic scheduling algorithms.

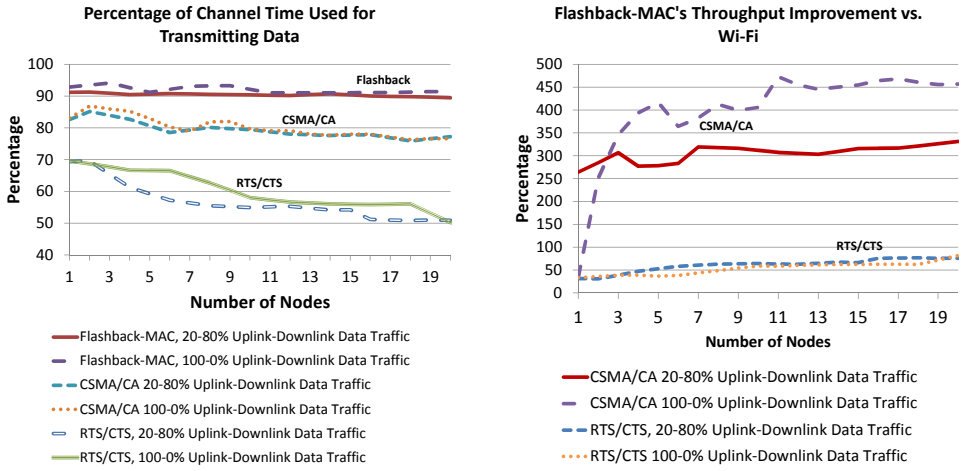

Figure 11: Flashback-MAC using FIFO scheduling compared to CSMA/CA and RTS/CTS with $1,000$ byte packets.

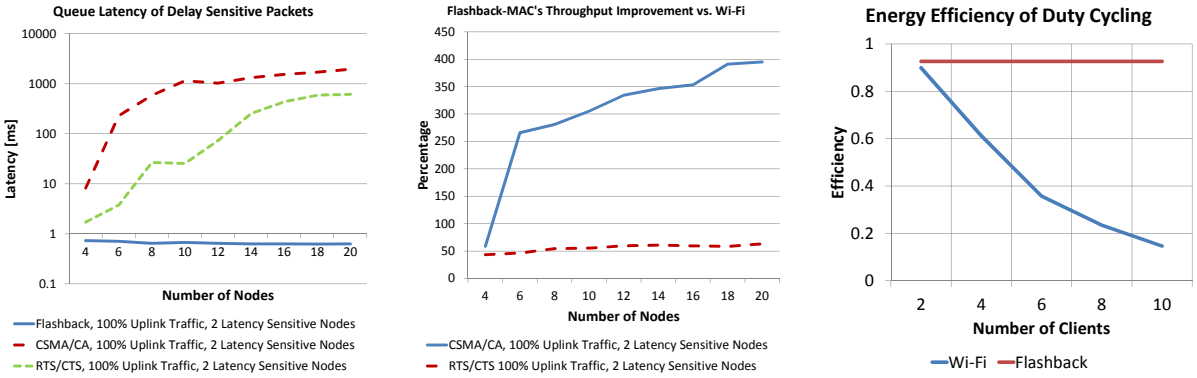

Figure 12: Flashback-MAC using QoS-aware scheduling and duty cycling, compared to RTS/CTS and Wi-Fi PSM. In the two graphs on the left, two nodes are sending latency-sensitive packets, while the rest are sending normal packets. Latency is measured as the delay between the time the node requests to send the packet and when it starts sending the packet. The third figure shows the energy efficiency gains of using Flashback compared to Wi-Fi PSM.

First, we designed a simple FIFO scheme where the AP schedules transmission requests in the order in which they arrive. Second, we simulated a QoS-aware scheduling algorithm that prioritizes latency sensitive traffic ahead of regular traffic.

Once the schedule is determined, the AP uses the next opportunity for a data plane transmission (either data or ACK) to signal which node gets to transmit next, and from which flow identifier.

Flashback-MAC provides significant benefits over the Wi-Fi MAC protocols. First, it eliminates overheads related to medium access. Nodes no longer have to perform carrier sense or exponential back off at the data plane, nor use any control messages such as RTS and CTS. These mechanisms add significant overhead, since they have to be sent at the lowest bitrate ($6Mbps$) compared to the data transmission, which can reach bitrates as high as $54Mbps$. Our simulation results show that such contention overhead consumes nearly 50% of the channel time in congested networks.

Second, it eliminates hidden terminals on the data plane, since nodes no longer have to use the inaccurate carrier sense mechanism to determine if a channel is idle. Instead, with Flashback-MAC the AP explicitly controls which node is transmitting and can actively prevent hidden terminals.

Third, it allows the AP to implement QoS on the uplink and downlink efficiently. Explicit coordination allows Flashback-MAC to guarantee QoS in a centralized fashion, while paying minimal overhead.

Finally, Flashback-MAC is backwards compatible. Even if only the AP and some of the nodes in a network support Flashback-MAC, the AP can still service the requests from the nodes that do not support Flashback, as long as the AP ensures there is enough idle time in the network. The only caveat is that the old nodes may still impact the desired QoS of the Flashback-MAC nodes.

6.1.1 Performance and Latency

We simulated Flashback-MAC using an NS-3 Wi-Fi simulation. Our simulated network includes one AP and a varying number of clients that are randomly uniformly distributed on a 80×80 grid. We applied to the simulation the packet loss and flash error rate values that we collected from our Flashback receiver implementation and from our channel sounder traces. The nodes collectively flash at a maximum rate of $50,000$ flashes per second. We implemented the demand map abstraction for the AP, and simulated the scheduling policies described above.

In order to analyze the amount of control overhead Flashback-MAC can eliminate, we simulated a congested network of nodes, each trying to schedule a large amount of traffic ($80Mbps$ of $1,000$ byte packets). We compare the amount of time the network spends

on transmitting data with the amount of time the network spends on being idle or transmitting control messages. We measured this time for CSMA/CA, RTS/CTS and Flashback-MAC, under two scenarios: full uplink traffic, and a mix of 20% uplink traffic and 80% downlink. Fig. 11 shows the results of the simulation. As we can see, as the network grows larger, the overhead for RTS/CTS becomes greater, up to a level of 50%, while CSMA/CA falls to a level of about 77%. For CSMA/CA, as the number of nodes increases, more collisions occur due to hidden nodes, and nodes experience more packet loss and their bitrates deteriorate, which causes their packets to occupy the channel for a longer time. For RTS/CTS, the channel utilization decreases significantly, because the nodes spend more time contending on the channel in order to send their RTS or CTS packets. Flashback-MAC completely eliminates the hidden node problem of CSMA/CA, since it centrally schedules all the nodes in the network. In addition, Flashback-MAC does not suffer from RTS and CTS packet overhead, because it can send all the node requests over the control channel, and it piggybacks the request acknowledgements and scheduling decisions on the AP's ACKs. The reason Flashback-MAC has an overhead of about 8-10%, is that it still spends the Wi-Fi SIFS inter-packet idle time between sending a packet and receiving an ACK.

Flashback-MAC has significantly higher throughput than CSMA/CA (up to $5.5\times$) in congested networks, because CSMA/CA suffers from collisions due to hidden nodes, and hence nodes transmit at lower bitrates. All the nodes in our simulator use a bitrate adaptation algorithm that is based on packet loss. CSMA/CA's throughput can be improved, if we use a bitrate adaptation algorithm that discerns between SNR based packet loss and collision based packet loss. In comparison with RTS/CTS, the extra time that Flashback-MAC gains by not sending RTS and CTS packets, is translated into an increase of up to $1.8\times$ in throughput.

FIFO is of course a very basic scheduling algorithm. In order to demonstrate a slightly more sophisticated scheme that utilizes the demand map abstraction, we simulated a QoS-aware scheme, where incoming requests are inserted into two queues: a delay sensitive queue and a normal data queue. The delay sensitive queue is emptied at $4\times$ the rate of the normal data queue. Nodes can request to schedule a delay sensitive message by simply flashing a control message with the corresponding QoS bits.

The QoS simulation uses a similar network to the FIFO simulation. The only difference is that only two of the nodes send $1,000$ byte latency-sensitive packets at a relatively low rate ($800kbps$ per node), while the rest of the nodes are attempting to send regular $1,000B$ packets at the high rate of $80Mbps$. Fig. 12 depicts the results. As we can see, Flashback-MAC's QoS aware schedule can enforce that the latency of the delay-sensitive packets will be kept below $1ms$, while maintaining a significantly higher throughput than the Wi-Fi MAC protocols. Note that the low latency access is enforced, despite the fact that the control plane is congested. This is because control messages are small ($180\mu s$) relative to data packets. As a result, the QoS requests are promptly delivered to, and serviced by the AP. In conclusion, this simulation emphasizes the importance of decoupling the control and data planes.

6.1.2 Energy Proportional Data Plane

Flashback-MAC enables clients to operate the radios in an energy proportional manner. Energy proportionality means that a radio has to be awake only when it is sending or receiving useful traffic, and does not have to spend energy contending for channel access or waiting to be served by the AP. The basic idea is simple: the default mode for the client radio is to sleep. If a client wants to send packets to the AP, it wakes up and flashes a control message. The AP replies with the schedule, telling the client when it will have an opportunity to send its packet. The AP also informs the client if it has any downlink packets destined to it. If the client has no packets to send to the AP, it periodically (every $100ms$) wakes up and flashes a control message to the AP asking if there are any outstanding packets. The AP informs the client if it has any outstanding traffic and the schedule for that traffic. The client then wakes up at the appropriate time and waits for its packets to be delivered.

Fig. 12 plots the energy efficiency of a client using this protocol when compared to the state-of-the-art Wi-Fi Power Save Mode (PSM) used to conserve energy. Energy efficiency is defined as the fraction of time a client radio is sending and receiving useful traffic relative to the total time it is awake with both protocols. Wi-Fi PSM allows nodes to sleep with $100ms$ duty cycles, but if they have either uplink or downlink traffic, clients have to stay up until they get served. As we can see, clients obtain near optimal energy efficiency with Flashback, since clients do not have to stay up and contend to access the network like in normal Wi-Fi.

6.2 Centralized Enterprise Wi-Fi

We can use Flashback-MAC to enable an enterprise Wi-Fi network with the following properties.

- Seamless Mobility: Once a client authenticates with the network, it should be able to roam around the entire enterprise network and send and receive packets from the closest AP. It should not have to spend time and energy in re-associating.
- Better Load and Interference Management: The network should be able to assign clients dynamically to different APs to balance load. Within an AP, client performance should not suffer due to interference from other nodes.
- QoS: The network should be able to provide QoS to prioritize latency-sensitive traffic.

Current networks struggle to offer such capabilities. First, if clients are mobile, they are forced to re-associate with new APs as they move around. The association protocol is time consuming, since nodes need to wait for the beacons from the neighboring APs, find the one with the best signal strength, and then go through the association protocol even though they need to connect to the same enterprise network. Further, load balancing is hard, since clients make decisions on which AP to connect to and the network manager has little control over the allocation of clients to APs. Once connected, interference problems are common due to scenarios such as hidden terminals. Finally, the network has no capability of providing QoS. For example, there is no mechanism to provide low latency connectivity for VoIP traffic.

Flashback provides a general primitive to enable this network architecture, namely, a decoupled control plane. Nodes can utilize Flashback with the following protocol. When associating and authenticating with the network for the first time, nodes use Flashback to send control messages that include their address and authentication information. These messages can be sent at any time, without the node having to wait for the beacon and then contend for sending an association request packet to the AP. Further, nodes can just broadcast the Flashback association message, without specifying the AP. Any AP which hears the association message forwards it to a central controller. The controller assigns the client to the best available AP depending on the load on each AP and provides it with an authentication token. When the client wishes to initiate a flow, it flashes a control message with the request, which includes the authentication token. Since the token is recognized by all the APs in the network, any AP can immediately schedule the request.

This protocol can ensure that client association and mobility is seamless. First, clients do not have to associate with a particular AP

to be able to send and receive traffic. Second, clients are presented with a simple flow abstraction: clients request a flow setup, and the controller just assigns the client the best possible AP to relay that traffic. This naturally allows for mobility, since there is no longer a notion of being connected to an AP. Further, since the controller can decide which APs to assign to which clients, it can dynamically adjust to varying load and ensure that no AP is overloaded.

Note that this functionality is very difficult or impossible to obtain with normal RTS/CTS style control signaling that is coupled with the data plane. Any type of control message requires a node to contend for access with other transmissions, whether it's for association or load-balancing.

7. RELATED WORK

Our flashing mechanism and its name were mainly inspired by FlashLinQ [21], a centrally synchronized peer-to-peer cellular protocol that also utilizes the positions of narrowband signals on different frequencies to encode scheduling information. Unlike Flashback, FlashLinQ requires tight and centralized symbol-level synchronization (it uses cellular GPS based time synchronization), which is not available in a distributed asynchronous network like Wi-Fi.

Flashback is related to prior work on pulse based messaging in Zigbee networks [10], which allows a node to concurrently send control messages with data transmissions. However, this technique sends a time domain pulse across the entire bandwidth and corrupts the whole packet. It requires the data transmission to be very resilient to interference, and therefore only works with protocols such as Zigbee, which do not perform rate adaptation and have very robust link margins. Flashback on the other hand, exploits the unique properties of OFDM to flash in order to localize interference. Moreover, Flashback works under Wi-Fi's rate adaptation mechanism, where link margins are much smaller than Zigbee.

Most synchronized or centralized wireless networks use some kind of independent control channel, which in many cases is physically separated from the data channel. For example, LTE has dedicated physical control channels both for downlink and uplink traffic [13, 8]. These control channels have pre-allocated frequencies, which creates a clear demarcation between the control and data channel bands. This type of frequency based separation is not possible for asynchronous distributed networks like Wi-Fi, because such networks are not centrally managed. Hence, in Wi-Fi, the receiver needs to re-synchronize each time to the transmitter's transmission (using the time-based preamble). This makes it very complicated for the receiver to decode transmissions of multiple senders on different bands in the same time slot, as base stations do in LTE. In particular, LTE and WiMAX [1, 2] both utilize OFDMA, a multi-user OFDM scheme that requires tight synchronization, where different nodes can transmit simultaneously on different subcarriers.

In addition, similar to Flashback, Sen et al. [17], utilize a back off arbitration mechanism based on randomly selecting subcarriers. The main difference between the two control techniques is that Sen et al. only use their technique when the data channel is idle, where Flashback allows nodes to flash concurrently with data transmissions and therefore creates a decoupled control plane.

8. CONCLUSION

Flashback follows the classic networking architecture of decoupling the control and data planes. Moreover, it provides this separate control plane at minimal overhead. We believe Flashback can be applied to a number of problems in wireless networks, ranging from fast network association, to mobility and power saving client modes. Further, Flashback can serve as a substrate for distributed coordination to tackle coexistence in dense network deployments. We plan to explore these applications in our future work.

9. ACKNOWLEDGEMENTS

We would like to thank Aditya Gudipati and Steven Hong for their help in setting up the experimental framework, and Manu Bansal, Dinesh Bharadia, Israel Cidon and Jeffrey Mehlman for their valuable feedback. We would also like to thank our shepherd, Krishna Chintalapudi, and the anonymous reviewers for their comments. Asaf Cidon is supported by the Leonard J. Shustek Stanford Graduate Fellowship.

10. REFERENCES

[1] Ieee standard for local and metropolitan area networks, part 16: Air interface for fixed wireless broadband systems.

[2] Ieee standard for local and metropolitan area networks, part 16: Air interface for fixed wireless broadband systems, amendment 2: Physical and medium access control layers for combined fixed and mobile operation in licensed bands and corrigendum 1.

[3] Labview system design software,http://www.ni.com/labview/.

[4] Ni pxie-8133 user manual, http://www.ni.com/pdf/manuals/372870b.pdf.

[5] CHANDRA, R., MAHAJAN, R., MOSCIBRODA, T., RAGHAVENDRA, R., AND BAHL, P. A case for adapting channel width in wireless networks. In *ACM SIGCOMM* (2008).

[6] CHENG, Y.-C., BELLARDO, J., BENKÖ, P., SNOEREN, A. C., VOELKER, G. M., AND SAVAGE, S. Jigsaw: solving the puzzle of enterprise 802.11 analysis. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2006), SIGCOMM '06, ACM, pp. 39–50.

[7] EUGENIO MAGISTRETTI, KRISHNA CHINTALAPUDI, B. R., AND RAMJEE, R. Wifi-nano: Reclaiming wifi efficiency through 800ns slots. In *Proc. of MOBICOM 2011*.

[8] GHOSH, A., RATASUK, R., XIAO, W., CLASSON, B., NANGIA, V., LOVE, R., SCHWENT, D., AND WILSON, D. Uplink control channel design for 3gpp lte. In *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on* (sept. 2007), pp. 1 –5.

[9] JUDD, G., AND STEENKISTE, P. Using emulation to understand and improve wireless networks and applications. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2* (Berkeley, CA, USA, 2005), NSDI'05, USENIX Association, pp. 203–216.

[10] KAISHUN WU, HAOYU TAN, Y. L. J. Z. Q. Z., AND M.NI, L. Free side channel: Bits over interference. In *ACM MOBICOM* (2010).

[11] KHURANA, S., KAHOL, A., AND JAYASUMANA, A. Effect of hidden terminals on the performance of ieee 802.11 mac protocol. In *Local Computer Networks, 1998. LCN '98. Proceedings., 23rd Annual Conference on* (oct 1998), pp. 12 –20.

[12] KUN TAN, JI FANG, Y. Z. S. C. L. S. J. Z., AND ZHANG, Y. Fine grained channel access in wireless lan. In *Proc. of SIGCOMM 2010*.

[13] LOVE, R., KUCHIBHOTLA, R., GHOSH, A., RATASUK, R., CLASSON, B., AND BLANKENSHIP, Y. Downlink control channel design for 3gpp lte. In *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE* (31 2008-april 3 2008), pp. 813 –818.

[14] MANGOLD, S., CHOI, S., MAY, P., KLEIN, O., HIERTZ, G., STIBOR, L., POLL CONTENTION, C., AND POLL, F. Ieee 802.11e wireless lan for quality of service.

[15] NG, P. C., LIEW, S. C., SHA, K. C., AND TO, W. T. Experimental study of hidden-node problem in ieee802.11 wireless networks *, 2005.

[16] PLESS, V. *The Theory of Error Correcting Codes*. Wiley Interscience Series in Discrete Mathematics and Optimization. New York: John Wiley and Sons, 1989. Second Edition.

[17] SEN, S., ROY CHOUDHURY, R., AND NELAKUDITI, S. No time to countdown: migrating backoff to the frequency domain. In *ACM MobiCom* (2011).

[18] SINGLETON, R. Maximum distance -nary codes. *Information Theory, IEEE Transactions on 10*, 2 (apr 1964), 116 – 118.

[19] TELEKOMMUNIKATION, F., CZINK, N., B, B., VAZQUEZ-VILAR, G., JALLOUL, L., AND PAULRAJ, A. Stanford july 2008 radio channel measurement campaign, 2008.

[20] VUTUKURU, M., BALAKRISHNAN, H., AND JAMIESON, K. Cross-layer wireless bit rate adaptation. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication* (New York, NY, USA, 2009), SIGCOMM '09, ACM, pp. 3–14.

[21] WU, X., TAVILDAR, S., SHAKKOTTAI, S., RICHARDSON, T., LI, J., LAROIA, R., AND JOVICIC, A. Flashlinq: A synchronous distributed scheduler for peer-to-peer ad hoc networks. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on* (29 2010-oct. 1 2010), pp. 514 –521.