Supporting Demanding Wireless Applications with Frequency-agile Radios

Lei Yang, Wei Hou†, Lili Cao, Ben Y. Zhao, Haitao Zheng
Department of Computer Science, University of California, Santa Barbara
†*Department of Electronic Engineering, Tsinghua University*
{*leiyang, lilicao, ravenben, htzheng*}*@cs.ucsb.edu, hou-w05@mails.tsinghua.edu.cn*

Abstract – With the advent of new FCC policies on spectrum allocation for next generation wireless devices, we have a rare opportunity to redesign spectrum access protocols to support demanding, latency-sensitive applications such as high-def media streaming in home networks. Given their low tolerance for traffic delays and disruptions, these applications are ill-suited for traditional, contention-based CSMA protocols.

In this paper, we explore an alternative approach to spectrum access that relies on frequency-agile radios to perform interference-free transmission across orthogonal frequencies. We describe *Jello*, a MAC overlay where devices sense and occupy unused spectrum without central coordination or dedicated radio for control. We show that over time, *spectrum fragmentation* can significantly reduce usable spectrum in the system. Jello addresses this using two complementary techniques: *online spectrum defragmentation*, where active devices periodically migrate spectrum usage, and *non-contiguous access*, which allows a single flow to utilize multiple spectrum fragments. Our prototype on an 8-node GNU radio testbed shows that Jello significantly reduces spectrum fragmentation and provides high utilization while adapting to client flows' changing traffic demands.

1 Introduction

The future is bright for next-generation wireless devices. While current technologies are limited to operating in fixed ranges of increasingly congested spectrum, reforms in spectrum management policy promise to free up spectrum in the near future. The Federal Communications Commission (FCC) has auctioned recently vacated wireless spectrum to service providers [9]. To further democratize the use of this spectrum, online spectrum trading services such as SpecEX (*www.spectrumbridge.com*) now allow small service providers to purchase/rent spectrum directly from regional owners.

Unlike unlicensed bands used by current wireless devices, these new spectrum ranges are large and uncongested. We can take advantage of the opportunity to redesign access mechanisms to support a broader range of wireless applications. For example, current wireless access mechanisms are designed for best effort traffic, and generally rely on spectrum contention as used in CSMA protocols and their variants. The network partitions spectrum into fixed channels, lets each transmission choose a channel and contend in time with its peers. While this approach works quite well for file transfers and interactive applications, past work shows that supporting applications with real-time requirements requires additional modifications that incur significant overheads [25, 27].

In this paper, we reconsider the design of spectrum access mechanisms in dynamic spectrum networks to support applications within more restrictive traffic classes. Specifically, we consider supporting applications with strong quality of service requirements such as high-definition multimedia flows in media rich environments like the home. Traffic demands for these flows can vary significantly over time, but can generally be predicted ahead of time. Unlike best-effort traffic applications, these multimedia flows require dedicated spectrum access to minimize disruptions to their transmissions and to maintain the expected quality of user experience.

We make two observations that make existing contention-based systems unsuitable for these applications. First, *per-packet contention* produces frequent and unpredictable transmission disruptions, which would interfere with our desired traffic delivery constraints. In contrast, if multiple transmissions were allocated isolated frequencies, each flow would obtain necessary dedicated spectrum, while avoiding costly interference that traditionally leads to contention and communication delays [18]. Second, splitting spectrum into *fixed channel partitions* is also unattractive for applications with time-varying bandwidth demands. Fixed partitions prevent flows from using or releasing available spectrum as necessary, and would lead to inefficient spectrum usage [8]. In this respect, new hardware in the form of *frequency-agile radios* can be extremely useful. With these radios, a

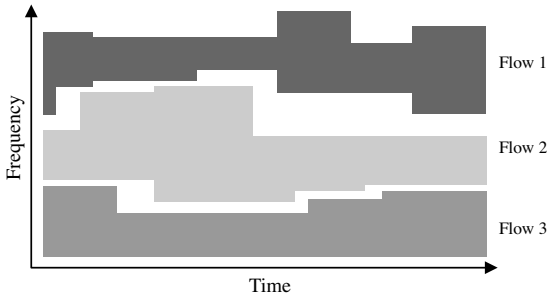


Figure 1: Per-session FDMA: Simultaneous media sessions work in parallel on isolated frequencies, avoiding costly wireless interference while adapting frequency usage to varying traffic demands.

device examines locally available spectrum before each network connection, and directs its radio to operate on a frequency range that not only matches its traffic demands, but also lies orthogonal to existing transmissions. In addition, devices can grab and release spectrum as necessary without being confined by fixed partitions.

Motivated by these observations, we propose a new distributed access technique that lets flows access spectrum in the frequency domain and adapt their spectrum usage based on traffic demands (shown in Figure 1). We refer to this new access technique as "per-session FDMA," where each session refers to a single continuous flow, and build a basic framework where traffic flows can independently select and adapt their frequency usage. First, by detecting "edges" on observed power spectrum maps, each device can accurately and quickly identify free spectrum in its local area. Second, each device can select an available spectrum range based on its present traffic demands, using classical algorithms such as *best fit*, *worst fit*, and *first fit* [19]. Finally, we propose a distributed coordination procedure to synchronize sender and receiver pairs in their spectrum usage.

Several recent proposals describe systems that adapt spectrum usage based on bandwidth demands [15, 20, 32]. In this context, our work builds an efficient framework that determines how device pairs sense and coordinate their access in open spectrum ranges. Our system is MAC-agnostic: once devices obtain spectrum using our primitives, they can use any MAC.

Spectrum Fragmentation. Efforts to evaluate our basic design reveal another fundamental challenge. Over time, as individual transmissions enter and exit the network or adjust their spectrum usage, available spectrum becomes increasingly divided into a collection of discrete fragments. This "spectrum fragmentation" means that a significant portion of spectrum, while free, is effectively unusable because its fragments do not provide the minimum contiguous spectrum range required by new flows. Our experiments show that this artifact does exist in practice, and leads to significant performance degradation even for networks with very few parallel transmissions.

We propose two distinct, but complementary mechanisms to address this fundamental problem: *online spectrum defragmentation* at the spectrum access layer, and *noncontiguous frequency access* at the physical layer. With online spectrum defragmentation, each pair of communicating devices voluntarily defragment spectrum by moving to alternative frequencies, thereby optimizing spectrum availability for other sessions. These frequency moves occur periodically in a session or as flows adapt to changing spectrum demands. They are nearly instantaneous and transparent to neighboring flows. Given our emphasis on minimizing disruptions, however, this technique cannot completely remove spectrum fragmentation. As a complementary mechanism, we offer non-contiguous frequency access, where a radio can utilize multiple spectrum ranges in a single transmission. This provides support for high-bandwidth transmissions even in the presence of moderate levels of spectrum fragmentation. Our approach implements non-contiguous frequency access using a "distributed OFDMA" mechanism, which differs from prior approaches like SWIFT [24] that rely on CSMA to share spectrum among frequency-agile radios.

These two techniques work best in unison. Non-contiguous frequency access requires "frequency guard bands" between allocated frequency boundaries to eliminate cross frequency interference, similar to guard bands between WiFi channels. Since they are not usable for communication, guard bands represent spectrum overhead that increases as flows make use of more fragmented spectrum ranges. Online spectrum fragmentation, on the other hand, effectively suppresses the level of fragmentation.

The Jello Overlay. Based on these two complementary techniques, we design and implement Jello, a MAC overlay to support high-bandwidth real-time applications. Jello does not require centralized spectrum controllers or dedicated radios for control traffic, making it a low-cost and easily deployed solution. Jello radios sense, identify and occupy usable frequencies based on traffic demands while minimizing spectrum fragmentation. Where low levels of fragmentation remain, devices accommodate high-bandwidth transmissions using non-contiguous frequency access. We deploy a prototype of Jello on a 8-node USRP GNU radio testbed, and evaluate the benefits of online spectrum defragmentation and non-contiguous frequency access, both individually and together. Measurements show that Jello reduces disruptions to applications by as much as a factor of 8.

Our work makes three key contributions. First, we explore spectrum access techniques for real-time wireless applications with low tolerance for traffic disrup-

tions, and propose mechanisms for frequency-agile radios to sense, occupy, and synchronize spectrum usage. Second, we identify the spectrum fragmentation challenge, and propose two complementary solutions to maximize spectrum utilization. Finally, we implement and deploy a prototype of Jello, a complete MAC overlay encompassing our techniques. We evaluate the effectiveness of Jello mechanisms using both detailed measurements of an 8-node GNU-radio testbed and simulated experiments. Jello provides interference-free access to demanding applications while maximizing utilization of available radio spectrum, and can be deployed on hardware available today.

2 A Case for Per-session FDMA

The expected arrival of new wireless spectrum is an opportunity to redesign spectrum access protocols to support a richer set of network applications. In particular, available spectrum can be used to support "soft real-time" applications, *i.e.* applications such as multimedia streaming that have very low tolerance for data loss, delays and jitter.

Given their strong demands on the underlying wireless network, these applications do not perform well on CSMA protocols that require parallel flows to perform per-packet contention. Recent experimental results show that such contention leads to unpredictable network delays and disruptions [25, 27], ultimately resulting in visible disruptions to the application-level user experience. Quality of Service extensions such as IEEE 802.11e can prioritize traffic, but does not prevent contention between multiple flows in the same traffic class, *e.g.* video streams in neighboring houses. An alternative for predictable traffic delivery is to employ Time Division Multiplexing (TDM) to obtain a collision-free transmission schedule. However, this requires fine-grain network-wide time synchronization and scheduling, which are difficult to implement in practice.

Assumptions. Our focus is on supporting demanding wireless media applications. We assume that these applications operate in a dedicated spectrum band, generate continuous traffic with time-varying load, and have strong quality of service requirements. In environments where they must co-exist with legacy systems using best-effort traffic, we envision that local wireless spectrum can be partitioned into two ranges for isolation. One range is dedicated to legacy applications using 802.11 CSMA, and the other is dedicated for media-streaming applications running our proposed protocols.

Frequency-agile Radios. Recent hardware advances have produced "frequency-agile radios," wireless radios capable of operating across a wide range of frequencies and jumping between them in milliseconds. Currently available hardware includes the WARP [30], USRP [21], AirBlue [16] and SORA [28], with more expected in the next few years. With these radios, we can now consider *per-session FDMA*, or Frequency Division Multiplexing Access. In this approach, parallel sessions occupy orthogonal spectrum ranges, thus completely avoiding cross-flow interference. When a media session starts, the two end-devices involved choose a free frequency block to set up packet transmissions. As shown in Figure 1, flows can adapt their frequency usage over time as their bandwidth demands vary, thus using time multiplexing to make the best use of radio spectrum. Recent work [15] shows that adapting spectrum on demand leads to 75% improvement over 802.11b.

Our approach differs from the concept of adapting frequency bandwidth on conventional 802.11 devices [8], where 802.11 channels can change their width to 40, 20, 10 or 5MHz by adjusting clock cycles. Our experiments show that scaling up traffic to fixed channel widths can reduce utilization up to 30% in our application scenarios. In comparison, per-session FDMA operates across wider spectrum ranges at fine granularities to ensure high utilization, completely eliminates CSMA traffic contention. Furthermore, each link now can flexibly combine multiple spectrum ranges to form high bandwidth transmission. The proposed per-session FDMA can work on any of the current frequency-agile radio designs [2, 16, 21, 24, 28, 30]. Since our approach operates directly on frequency bands, and uses frequency selection to avoid access conflicts, we also differ from prior work [15] that uses pseudo-random spreading codes to implement random spectrum access.

Challenges. A practical per-session FDMA system for wide-spread deployment needs to support soft real-time applications without relying on centralized spectrum controllers or costly dedicated radios for control traffic. Such a system must address several key challenges. First, to avoid disrupting ongoing transmissions, devices must be able to accurately and quickly identify free frequencies. Second, each transmission pair needs to select a free spectrum block based on their traffic demand while minimizing spectrum fragmentation. They also must do so without disrupting other ongoing transmissions, and without the help of any control radio. Similarly when a transmission pair needs to change frequency usage to accommodate variations in traffic demand (those cannot be handled by MAC rate adaptation), they also need to make the process transparent to others.

3 Jello Framework

To address these challenges, we propose Jello, a light-weight MAC overlay system that realizes distributed per-

session FDMA. Jello radios sense, identify and occupy usable frequencies to support time-varying traffic demands and to avoid interfering with each other. Each Jello device has a single half-duplex frequency-agile radio for wireless communication, and does not require any central control or dedicated control radio.

3.1 Identifying Usable Spectrum

When accessing spectrum, Jello devices must avoid conflicting with other ongoing sessions. Jello achieves this by performing spectrum sensing to quickly and accurately identify usable spectrum ranges. Unlike the time-domain sensing approach [4], Jello uses a frequency-domain mechanism, benefiting from its radio hardware's frequency-agility. Unlike WiFi devices that sequentially scan channels, a frequency-agile radio can listen to the entire spectrum span, as demonstrated by several available radio platforms [24, 30]. Using the frequency-domain signal, each radio constructs a power spectral density (PSD) map [13] that measures the energy level on each small frequency range.

To identify usable frequency blocks, conventional approaches perform energy detection on the PSD map [10]. For a given threshold Γ_{energy}, each radio treats frequency ranges with energy higher than Γ_{energy} as busy and the rest as unoccupied. The detection accuracy, however, is shown to be highly sensitive to the choice of Γ_{energy} and finding a uniformly optimal Γ_{energy} is unrealistic [24]. Recent work proposes to cross-validate the detection result by "poking" transmissions on "busy" frequency ranges and observing their reactions [24]. Each poking event disrupts existing transmissions, forcing them to move to other frequencies or change their transmission parameters. Thus while this solution works for transmissions that are highly resilient to frequent disruptions, it would cause serious performance issues for the media sessions our system targets.

Sensing via Edge Detection. We exploit a unique property of radio transmissions in the frequency domain for accurate detection. To avoid interference to other transmissions, OFDM based transmitters use filters to limit the radio energy within certain frequency bands. As a result, the PSD profile of each transmission has clear edges on the frequency band boundaries, regardless of energy levels (shown in Figure 2). We can reliably identify usable frequency blocks by identifying these edges.

Our edge detection mechanism works as follows. First, as a pre-processing step, we smooth the PSD map by averaging it over multiple consecutive observations and applying two coarse power thresholds to filter out obvious frequency ranges. Frequency ranges with very high power are treated as busy and very low power ones as occupied. This pre-processing aims to filter out most noises in the PSD map before trying to locate edges. This technique has been sufficient in our experiments without using sophisticated smoothing algorithms like [5].

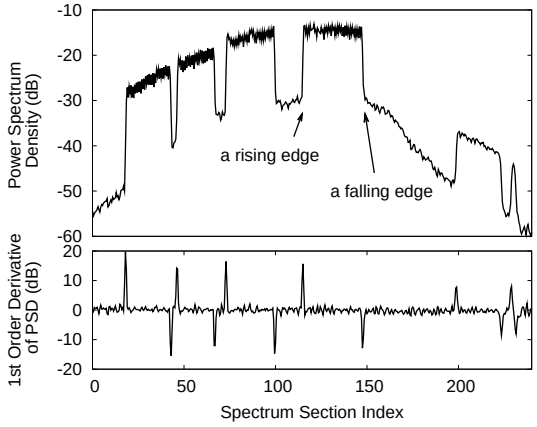


Figure 2: A sample PSD map and its first-order derivative. Jello identifies occupied frequency blocks using edge detection. While the absolute signal strength varies significantly across the frequency, the rising/falling edges are easier to detect.

Second, we apply search-based edge detection [14] and measure the edge strength by the first-order derivative of the PSD map. Let $P(k)$ represent the energy value of a spectrum section with index k, and let $P'(k)$ represent its first-order derivative. To decide whether edges are present, we choose a detection threshold Γ_{edge}. If $P'(k) > \Gamma_{edge}$ then k has a rising edge and if $P'(k) < -\Gamma_{edge}$ then k has a falling edge. A frequency block with a rising edge to its left and a falling edge to its right is declared as busy and the rest as free.

Compared to the energy detector, the edge-detection based sensing is less dependent on the choice of detection threshold. As shown in Figure 2, while the absolute signal strength varies significantly over the frequency, the rising/falling edges are easy to detect. This design works well in OFDM-based systems where the PSD map can capture frequency usage accurately and on-demand. While other forms of interference such as wireless microphones might not display the similar edges in the PSD map, we can incorporate other mechanisms such as feature detection based sensing for improved accuracy [4, 11]. In our target scenario, we focus on a homogeneous setting with radios all using OFDMA and within a short distance, thus our proposed sensing mechanism works well.

Calibrating Sender/Receiver Sensing Results. Each sender/receiver pair must synchronize their sensing results to identify mutually available frequency ranges. While the sender must pause its transmission to sense spectrum, the receiver senses while receiving at no extra

cost. Therefore, the receiver constantly monitors spectrum usage and locates occupied frequency ranges based on its maximum tolerable noise and interference level. When new spectrum blocks become available, it piggybacks the information via data or control packets to signal the sender to sense.

3.2 Choosing Frequency Blocks

After identifying mutually available frequency ranges, a sender/receiver pair needs to choose a frequency block to occupy. Such decisions usually occur when sessions start. It can also happen during a session when traffic changes cannot be handled by MAC rate adaptation. The device pair determines the amount of frequency needed based on estimated traffic demands and estimated MAC transmission rates on available frequency ranges. They can expand/shrink the current frequency usage, or move to a different frequency block. The ultimate goal is to obtain desired spectrum while maximizing system-wide usage efficiency.

The frequency selection problem is analogous to the online task scheduling problem [19]. Due to the unpredictable dynamics of spectrum demands, optimal solutions are hard to find. Similar problems have also been studied extensively in the context of CPU, memory and storage allocations. The most efficient known solutions apply heuristics-based algorithms [19], which have been shown to perform very well in most cases. In particular, we consider the well-known *best fit* strategy that selects the smallest available frequency block that can accept the current spectrum request, the *worst fit* strategy that uses the largest available block, and the *first fit* strategy that uses the first large enough block. When no block is large enough to satisfy a session's demand, we choose the largest block to accept the session partially. We found in our experiments that *best fit* outperforms others.

Propagation-aware frequency selection. In some cases, radio propagation conditions differ significantly across frequency ranges, *i.e.* due to channel fading. Information on received signal and interference strength, if available, can be integrated into Jello's frequency selection algorithm to select high-quality blocks that provide better reliability and higher bandwidth [4, 23]. In the current Jello prototype, the propagation quality is flat across the frequency span considered, thus the receivers use the measured interference strength in their selection process.

4 Suppressing Spectrum Fragmentation

Efforts to evaluate our basic Jello design reveal another fundamental challenge. Over time, as individual transmissions enter and exit the network or adjust their spectrum usage, available spectrum becomes increasingly divided into a collection of fragments (Figure 3(a)). This is because each radio must access spectrum contiguously, *i.e.* using a single frequency block. In this case, although a significant portion of spectrum remains unoccupied, it is effectively unusable because no individual fragment is large enough for a new request. A similar fragmentation problem appears in disk and memory allocation [19]. In this section, we examine the severity and impact of spectrum fragmentation, and propose two distinct but complementary techniques to minimize it. We provide high-level descriptions of our proposed techniques and delay the detailed implementation issues to Section 5.

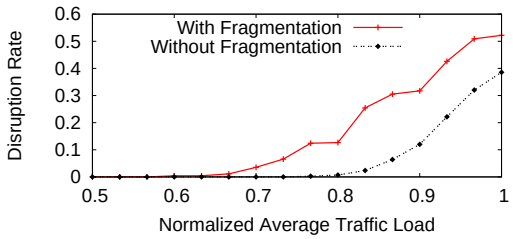


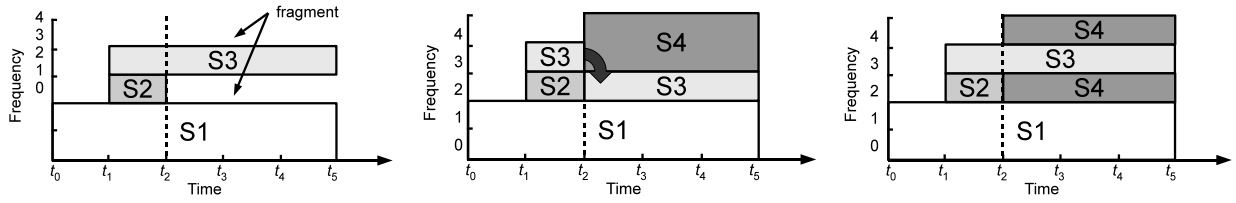
Figure 4: The impact of spectrum fragmentation with 4 streaming media sessions, using VBR video traces from the ASU trace database [3]. We compare the basic Jello system (with fragmentation) to an oracle system that eliminates all fragments.

4.1 Impact of Spectrum Fragmentation

To understand the severity and impact of spectrum fragmentation, we perform a detailed simulation using video traces from an online database [3]. Using a number of frame traces of H.263 video sessions, we simulate a scenario of multiple media sessions within close proximity. We measure the impact of spectrum fragmentation by *application disruption rate*, defined as the percentage of time a session cannot obtain enough spectrum to support X% of its present traffic demand.

We compare two possible frequency access systems: (1) an oracle system that rearranges sessions' frequency usage to defragment the spectrum completely; and (2) a basic Jello system where sessions "claim" their needed spectrum when they start, and do not change frequencies unless their spectrum demands change.

Figure 4 plots the application disruption rate for 4 variable bit rate (VBR) video sessions for $X = 90\%$. On the x-axis, we show the ratio of the total average traffic load of all 4 videos to the spectrum capacity. Clearly, the oracle system that fully defragments the spectrum performs significantly better when the 4 videos present a significant portion of all available spectrum. To guarantee that the disruption rate never rises above 3%, the basic system can only support traffic equal to 67% of the total



(a) Spectrum Fragmentation (b) Online Defragmentation (c) Non-contiguous Frequency Access

Figure 3: Spectrum fragmentation and ways to mitigate its impact. (a) When sessions share spectrum by accessing contiguous frequency, they can create spectrum fragments. (b) After $S3$'s self defragmentation, the same spectrum can now support more spectrum requests. (c) Session S4 uses two spectrum fragments for a single transmission.

spectrum capacity, while the oracle system can support traffic up to 83% of the spectrum capacity*. This is a significant boost in allowed traffic volume, and underlines the significant impact that fragmentation has on system performance.

4.2 Online Spectrum Defragmentation

The above results motivate us to improve Jello's basic design to suppress spectrum fragmentation. The first and most direct solution is to perform online defragmentation. A naive strawman version is to periodically perform global defragmentation where all sessions pause their transmissions, rearrange their frequency usage so that unoccupied frequency blocks are merged into a large contiguous range. This, however, is infeasible in our problem context because we must minimize disruptions to ongoing traffic flows. There is also no central controller to perform global defragmentation.

Instead, we propose an online, distributed approach to defragmentation: ongoing transmissions periodically consider moving to an alternative spectrum block using the *best-fit algorithm* to optimize overall spectrum availability. Each sender/receiver pair periodically senses local spectrum usage, and if possible, coordinates to switch to a frequency block that better optimizes the overall spectrum availability. For example, Figure 3(b) follows our earlier scenario where a session S_2 terminates and leaves a spectrum fragment. If S_3 voluntarily moves to spectrum block 2–3, the new request S_4 can be fulfilled and the overall spectrum utilization increases. Finally, using spectrum sensing to identify unoccupied frequency ranges, each device pair independently defragments spectrum without coordinating with other pairs.

Cost. The cost of our online defragmentation includes (1) the sensing and coordination overhead spent by device pairs to identify unoccupied spectrum and rearrange their frequency usage, and (2) possible conflicts when two sessions simultaneously defragment and make conflicting frequency adjustments.

Focusing on minimizing disruptions to ongoing sessions, Jello uses the following mechanisms to minimize defragmentation cost:

• *Minimizing Sensing/Coordination Overhead*: To minimize sensing overhead, Jello receivers constantly monitor spectrum to identify possibly opportunities for defragmentation. They signal their senders to perform sensing only after identifying possible opportunities themselves. To minimize coordination delay, each sender/receiver pair uses their present frequency block to exchange handshakes and schedule frequency adjustments. At initialization or during a unlikely event of lost synchronization or link failure, Jello devices enter a SYNC state to recover and resume communications.

• *Avoiding Defragmentation Conflicts*: Multiple devices can simultaneously detect a defragmentation opportunity and make conflicting frequency adjustments. Jello minimizes such conflicts by randomizing defragmentation efforts to avoid simultaneous adjustments.

4.3 Non-contiguous Frequency Access

Our second solution is to enable radios to combine multiple spectrum pieces to form a single transmission. Shown in Figure 3(c), S_4 now combines frequency block 2 and 4 together in a single transmission as if it uses a single frequency block.

Non-contiguous frequency access is now widely used in centralized wireless networks such as WiMAX and cellular LTE systems. It is implemented in the form of Orthogonal Frequency-Division Multiple Access (OFDMA). Existing designs, however, require global synchronization, and fail when applied to distributed networks without global synchronization. A key contribution of Jello is to identify and address the challenges of implementing *distributed OFDMA* and to prototype our design on USRP GNU radios.

Cost. To minimize interference, frequency guard bands

*The oracle cannot support 100% traffic load because the flows are VBR and the peak load occasionally exceeds the spectrum capacity.

must be placed at link boundaries [17]. Guard bands are not usable for transmissions, and are essentially spectrum overhead. Frequency guard bands are not an artifact of non-contiguous frequency access: they are required for contiguous access including 802.11 channels (which use 16% of frequency bandwidth as guard bands). On the other hand, the amount of guard bands increases when links start to use non-contiguous frequency blocks.

4.4 The Case for a Unified Approach

With the above two solutions, we ask the question: "*Is one solution sufficient enough to address spectrum fragmentation?*"

First, consider a scenario where only online spectrum defragmentation is available. While this technique improves spectrum utilization overall, each sender-receiver pair is acting independently, and cannot disturb other ongoing transmissions. Therefore, only a limited level of spectrum defragmentation is possible, and this technique cannot achieve the same effectiveness as a global, synchronized defragmentation approach. Thus a low level of fragmentation might remain.

Next consider a network using noncontiguous frequency access, but no online defragmentation. While this technique allows devices to utilize spectrum fragments as if they were a single contiguous fragment, it comes at the cost of multiple guard bands between link boundaries. Without online defragmentation, spectrum fragmentation will continue to degrade over time. Spectrum lost to guard bands will continue to increase, lowering overall spectrum utilization.

Clearly, neither technique by itself can fully address the challenge of spectrum fragmentation. Together they form a more complete solution. Online defragmentation limits spectrum fragmentation to a low level, and non-contiguous access makes all of the spectrum available without incurring significant overhead to guard bands.

5 Implementing Jello

We have implemented Jello on USRP GNU Radios. Despite having limited frequency bandwidth and large processing delays [12], USRP radios are widely available and fully reconfigurable across various protocol layers. We use the USRP implementation as a "proof-of-concept" evaluation of Jello. We modified GNU radio software to implement spectrum sensing, distributed contiguous and noncontiguous frequency access, online defragmentation, and sender/receiver coordination.

Figure 5 presents a high-level structure of Jello. At the physical layer, each Jello device operates on non-contiguous frequency ranges using distributed OFDMA.

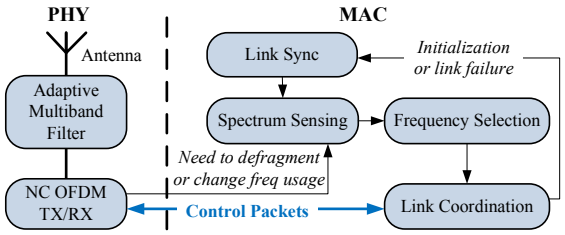


Figure 5: Jello system architecture.

At the MAC layer, Jello devices sense spectrum to identify usable frequency, and adapt their frequency usage when the application demand changes or when an opportunity to defragment appears. We now describe our implementation in detail.

5.1 Physical Layer

At the physical layer, Jello's key contribution is to implement spectrum sensing and distributed frequency access, both contiguous and non-contiguous, on today's common off-the-shelf hardware. Jello implements frequency access using OFDMA, which partitions the spectrum span into many small subcarriers. OFDMA has been widely used in centralized systems such as WiMAX, which divides a 20MHz frequency range into 2048 subcarriers of 10KHz each. Each sender can transmit on any subset of the subcarriers, either contiguously or non-contiguously aligned in frequency. Each receiver can listen to the *entire* set of subcarriers at once. Simultaneous transmissions can occur at different subcarriers without interfering with each other.

Implementing OFDMA on distributed networks, however, is hard. Existing designs in centralized networks rely on global synchronization to maintain subcarrier orthogonality, so that transmissions on isolated subcarriers do not interfere with each other. In distributed networks, where global synchronization is infeasible, OFDMA transmissions fail. To understand the causes, we perform an experiment by configuring 4 links on different frequency subcarriers. Our results show that significant link failures occur. The failures are not caused by the inherent propagation impairments, but by the following two reasons:

(1) *Unable to detect packet preamble*: In many cases, the receivers cannot detect any preamble that marks the beginning of a packet. This is because OFDMA detects preambles using a time-domain "delayed correlation" property from a signal placed at the head of each packet [29]. Because preambles from multiple transmissions are no longer synchronized, the delayed correlation property no longer holds in time-domain signals, preventing any successful preamble detection.

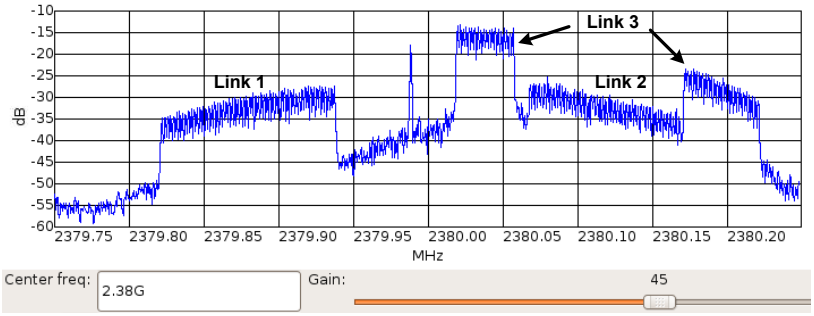


Figure 6: An example of Jello's flexible distributed spectrum access, implemented on USRP GNU radios. Three transmissions access and share radio spectrum in the frequency domain. Among them, link 3 operates on two non-contiguous spectrum blocks to form a single transmission.

(2) *Unable to decode data packet*: Even after fixing the preamble detection, significant losses still occur during packet decoding. This is because while multiple transmissions operate on different subcarriers, they leak energy to adjacent subcarriers, creating inter-carrier interference and destroying the subcarrier orthogonality at receivers. Compared to the preamble, packet data is much more vulnerable to interference because it is sent fewer error protections.

This motivates us to design receivers that "filter" out or minimize unwanted signals to restore the desired transmission properties. With this concept in mind, we propose two new mechanisms on top of the conventional OFDMA design to restore successful transmissions in distributed networks.

Restoring Preamble Detection. To restore the delay correlation property required for preamble detection, we apply an adaptive filter at receivers to remove signals from unwanted subcarriers. To support non-contiguous frequency access, we use a multi-band filter bank. Given the knowledge of the subcarriers used by its transmitter, the receiver first applies a low-pass filter to eliminate signals outside of its lowest and highest indexed subcarriers, and then uses multiple band-stop filters to remove signals from other unwanted subcarriers within the range. This design allows receivers to adapt filter ranges on-the-fly.

Without global synchronization, devices also experience frequency offset [13], defined as the frequency skew between devices' central carrier frequency. The presence of frequency offset could lead to errors in signal filtering. To suppress its impact between sender/receiver pairs, Jello receivers dynamically adjust their carrier frequency and filter width based on the result of preamble detection. At initialization it starts from a loose filter and gradually shrinks the filter to suppress interference. If the filter becomes too tight and fails to detect any preamble in a period, the receiver expands the filter to capture more subcarriers. After each successful preamble decoding, it estimates the frequency offset from its sender and refines the filter parameters.

Restoring Reliable Packet Receptions. While the use of receiver filters significantly improves preamble detection, packet losses can still occur due to out-of-band emissions among transmissions [17]. This work also shows that placing frequency guard bands between transmission boundaries is the most effective solution. To minimize these overheads, Jello devices directly measure interference power levels from the PSD map, and avoid using severely affected frequencies. This technique, combined with the adaptive filtering, allows Jello devices to correctly determine and minimize the usage of guard bands.

GNU Radio Implementation. We implement Jello's distributed OFDMA at 2.38GHz on a spectrum band of 500kHz. We use 256 subcarriers (or frequency sections), each of size 1.953kHz. To carry adequate signals for reliable preamble detection, each transmission must use at least 28 subcarriers, which can be non-contiguously aligned. We implement the receiver filter using the *hamming window* approach [22]. To compensate the frequency offset between sender and receiver, we initially extend the filter by 5 subcarriers and then adjust its central frequency and width on-the-fly. We found in our experiments that adding the receiver filter helps to reduce the amount of guard bands. Overall, placing 2 subcarriers at each link boundary is sufficient to protect all the links in our experiments.

Figure 6 illustrates an example PSD map of a system with three links. In this example, both link 1 and 2 occupy a contiguous block while link 3 utilizes two blocks simultaneously to build a high bandwidth transmission. Small guard bands were placed at link frequency boundaries to minimize cross-link interference.

We implement the spectrum sensing directly over OFDMA. Each device performs the Fast Fourier Transform (FFT) on collected frequency signals, and averages

the results over 50 OFDM symbols[†] to produce a PSD map. It computes the first-order derivative and uses a threshold of Γ_{edge} =5dB to locate edges. We chose these parameters because they work well in our experiments.

5.2 Access Layer

At the access layer, each Jello device will select frequency blocks to set up its communication session. During the session, it adapts its frequency usage when its traffic demand changes or when an opportunity for defragmentation appears. Without any dedicated radio for control, Jello addresses the following challenges: (1) each sender/receiver pair needs to synchronize on their frequency usage to ensure reliable transmissions; (2) to avoid hidden terminal problem, each sender/receiver pair needs to coordinate and choose proper frequency block(s) that are available to both of them; (3) simultaneous transmissions need to avoid using overlapping frequency blocks; and finally (4) devices must be able to quickly recover from failures caused by channel impairments and external interference.

Synchronizing Sender/Receiver. Each Jello sender and receiver pair performs handshaking to synchronize the frequency blocks they use for data transmission. This coordination has low overhead and does not involve any contention among sessions. Because GNU radios have large processing delays [12], our current Jello implementation does not include per-packet acknowledgements. The handshaking process is always initiated by the sender.

To change a session's spectrum usage, the sender performs spectrum sensing to see if there is any opportunity for change. If so, it sends a request (REQ) to its receiver indicating its spectrum sensing results. After receiving a REQ, the receiver selects a proper set of blocks and replies with an acknowledgement (ACK) indicating the selection. It also starts to decode signals from the new blocks. Upon receiving an ACK, the sender configures its transmissions on the new blocks. ACK failures could lead to discrepancy between sender and receiver's frequency usage. Thus, after failing to decode packets for a period of T_{BOFF}, the receiver "switches" back to decoding on the original blocks.

Choosing Frequency Blocks. Each Jello pair first tries to find a contiguous frequency block using the *best-fit* algorithm. If no such block is available, the pair selects multiple frequency blocks following the "noncontiguous best-fit" strategy: select the largest available blocks until the remaining demand is less than the largest remaining available blocks; then use *best-fit* to choose the final block. This approach minimizes the number of blocks required for the session.

Avoiding Conflicts. When an opportunity to defragment spectrum appears, multiple device pairs could react simultaneously, thus leading to frequency adjustments that conflict. To minimize these conflicts, we incorporate a random delay to both the sender's sensing function and receiver's defragmentation triggering. First, upon detecting a defragmentation opportunity, the receiver waits for a random interval T^R_{sense}, and notifies the sender only if the opportunity still exists. Second, a sender always repeats its spectrum measurement after a random delay of T^S_{sense}. A frequency block is considered free only if it is found to be free during both measurements. Random backoffs reduce the probability of simultaneous defragmentation attempts, similar to the CSMA backoffs in 802.11. Finally, devices can configure their backoff windows based on the projected effectiveness of their frequency shifts, giving priority to those that can provide the maximum benefit to the system. For simplicity, Jello uses a uniform random backoff window.

Recovering from Failures. Despite minimizing link failures through careful coordination of spectrum sensing and selection, link failures are sometimes unavoidable. They can occur from external interference or an unlikely conflict scenario where two links simultaneously move to the same frequency block. Redundancy techniques such as error correction codes [24] can improve the robustness of coordination packets, but are ineffective under complete link failures. If a link fails due to interference or conflict, its sender-receiver coordination messages will also fail to reach their destinations.

To address this, Jello introduces a SYNC state that devices enter at initialization or when they detect a coordination failure. A sender enters the SYNC state after failing to receive any ACK after retransmitting a REQ N_S times, and a receiver enters the SYNC state after not receiving any packets for a time period T_{SYNC}. In the SYNC state, devices communicate on the "SYNC Frequency Set" (SCS), a set of frequency blocks dedicated for performing resynchronization. The sender and receiver perform normal handshakes to reestablish synchronization and move to selected frequency block(s). There are several ways to define SCS, in our current implementation we configure it as a preassigned frequency block known to all devices. Devices try to avoid using the SCS for data transmissions except as a last resort, maximizing the probability that the SCS is idle.

GNU Radio Implementation. We implement Jello's access layer as a user-level program. Because USRP radios have a large random processing delay up to 20ms [12, 21], we use relatively large timing parame-

[†]The typical OFDM symbol duration for 802.11 a/g radios is 4μs, so the sensing time is 0.2ms. In GNU radios, the symbol duration is 2ms and the sensing time is 100ms.

ters in our experiments: T^S_{sense} and T^R_{sense} are uniformly distributed in $[0.1s, 1s]$, $T_{BOFF} = 1s$, $N_S = 5$, and $T_{SYNC} = 3s$. Each Jello device tries to defragment the spectrum once every $10s$. We choose the 28 lowest indexed subcarriers (out of 256) as the SCS. Based on our experience with the experimental platform, we found these to be reasonable parameter values.

5.3 Unexpected Hardware Artifacts

We also observe two unexpected hardware artifacts that may affect Jello's testbed performance.

Amplified Impact of Frequency Offsets. The bandwidth limitation of USRP radios magnifies the impact of frequency offsets, because they are now larger than the subcarrier width. Our 20-day measurements show that the frequency offsets can reach 10KHz (≈5 subcarriers) but have relatively smaller variances (<2 subcarriers). To suppress its impact, we manually correct each USRP's central frequency by its measured average, reducing its frequency offset to < 2 subcarriers.

Artificial Signals. Due to imperfect RF shielding, a USRP radio may leak energy to its receiving path, creating a energy peak of random strength near the central frequency (shown in Figure 6 as a spike near 2.38GHz). As a result, a radio could mistake some free subcarriers as being occupied. In our experiments this artifact leads to a small amount (<2%) of spectrum sensing errors. The impact is minor because Jello uses temporal averaged signals in its sensing, reducing the peak's edge strength to that below the detection threshold.

6 Evaluation

We evaluate Jello using both network simulations and GNU radio experiments. We use simulations to evaluate Jello with various design choices and network configurations. We also run experiments on an indoor network of 8 GNU radios in a 12m×7m room (Figure 7), running 4 simultaneous media sessions. We configure each radio's transmit power so that each link maintains 5% or less packet loss when there is no interference present, and all links interfere with each other. Each GNU radio experiment lasts 10 minutes and is repeated 5 times.

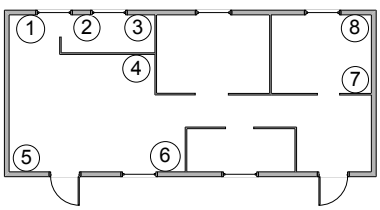


Figure 7: Our Jello testbed: 8 USRP GNU radios are placed in a 12m × 7m room with various walls and furniture.

We use both VBR video traces and synthetic On/Off traffic to generate sessions. We scale the traffic flow as necessary to create a desired load normalized by the frequency bandwidth. Sessions carrying video traffic have similar average loads, and sessions carrying On/Off traffic have different traffic volumes. For video traces, we assume a 10s application buffer so that each session's demand changes every 10s. For synthetic traffic, the On and Off periods are randomly generated from a uniform distribution. Each session determines the amount of frequency required based on its traffic demand and the average data rate achievable on each frequency subcarrier. If the current available frequency cannot fulfill the entire demand, the session will take what is available.

We evaluate Jello by comparing four systems:

• **Static**: partitioning spectrum equally by the number of sessions; each session has a dedicated frequency block.

• **Jello-C**: Jello with contiguous frequency access.

• **Jello-NC**: Jello with non-contiguous access enabled.

• **Optimal**: an "oracle" solution with perfectly accurate sensing that removes fragmentation by assigning spectrum using knowledge of all future requests.

We collect two performance metrics that measure application performance and spectrum usage efficiency:

Application disruption rate: the proportion of time that a session experiences packet losses higher than a maximum threshold X, and thus cannot sustain satisfactory media quality. For example, prior work shows that streaming video sessions can only tolerate up to 10% packet loss [26]. We examined Jello using $X = 5, 10, 20\%$, and arrived at similar conclusions. Due to space limitations, we only show results using $X = 10\%$.

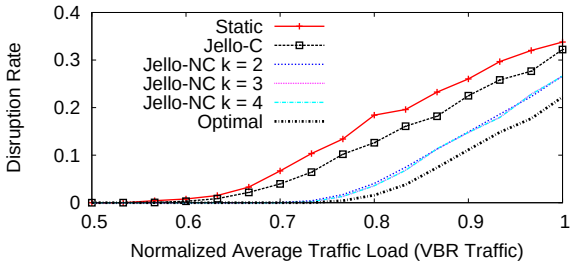
Residual usable spectrum: given a traffic load, the amount of spectrum left for a new media session, averaged over time and normalized by the spectrum capacity.

6.1 Simulation Results

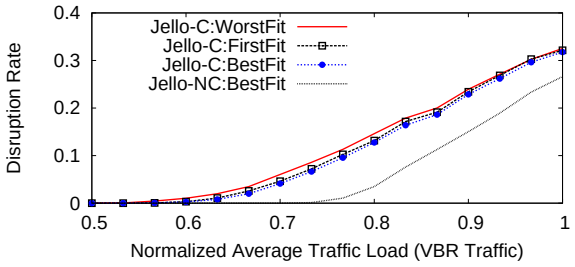
We first simulate Jello under general network configurations. We use these experiments to examine and verify Jello's design concept without any sensing/transmission error or coordination overhead. Figure 8 shows that Jello-NC, by enabling dynamic non-contiguous frequency access, significantly outperforms Jello-C and Static. There is a small distance between Jello-NC and Optimal because Jello-NC uses periodic defragmentation so that a low-level of fragmentation still remains, leading to some loss in spectrum from frequency guard bands.

We also make several key observations:

Impact of k (the maximum # of frequency blocks each radio can use). Since hardware complexity scales with k, it is interesting to understand its impact. In Figure 8



(a) Impact of max. frequency blocks allowed



(b) Impact of frequency selection algorithms

Figure 8: Simulated Jello performance using video traces: (a) when allowing each radio to access $k = 1..4$ frequency blocks; (b) when using different frequency selection algorithms.

	Normalized average traffic load				
	0.6	0.7	0.8	0.9	1
P(1 block)	98.9%	87.7%	70.3%	58.2%	52.1%
P(2 blocks)	1.1%	11.8%	25.1%	31.1%	33%
P(3 blocks)	0	0.4%	4.2%	9.1%	12.2%
P(4 blocks)	0	0.1%	0.4%	1.5%	2.4%

Table 1: Probability distribution of the number of frequency blocks each session uses, using Jello-NC with $k = \infty$.

we examine the application disruption rate of Jello-NC by varying k between 1 and 4. We see that raising k from 1 to 2 leads to a significant performance leap, but after that the benefit of raising k becomes marginal. To further examine this, we list in Table 1 the probability distribution of the number of frequency blocks each session uses when k is unlimited. We see that the need for non-contiguous access does increase with the traffic load, but each session uses no more than 3 blocks with a 97+% probability. We repeated our experiments using different traffic models and network sizes, and arrived at a similar observation. Although inconclusive, this shows that adding 1 or 2 bands to a radio's frequency access capability will significantly boost the overall performance. We also prove this trend analytically in a separate study [6].

Impact of the frequency selection algorithm. Figure 8(b) plots the application disruption rate of Jello-C with *Worst Fit*, *First Fit*, *Best Fit*, and Jello-NC with *Best Fit*. We see that Jello-C with *Best Fit* outperforms Jello-C with the rest but only slightly. In our testbed experiments, we use *Best Fit* for Jello.

Impact of network topology. We examine this impact using a network of 50 sessions. By varying the transmit power we create networks of different conflict conditions, represented by the average conflict degree D. Higher D means each session conflicts with more peers. Table 2 lists the application disruption rate for Jello-C, Jello-NC and Optimal. The same conclusion applies. One interesting observation is that Jello-NC leads to more gains as the conflict level decreases. This is because non-contiguous access provides more opportunity for spatial reuse where non-conflicting sessions can reuse the same frequency blocks.

	Average conflict degree D				
	3.9	5.1	6.3	7.5	8.5
Jello-C	0.025	0.057	0.107	0.155	0.207
Jello-NC	0.005	0.019	0.054	0.095	0.147
Optimal	0.001	0.005	0.018	0.037	0.065

Table 2: Application disruption rate with different conflict degrees using a large network of 50 sessions.

6.2 Testbed Results

We now evaluate Jello using the GNU radio testbed. All the results now include the impact of channel impairments, but those of Jello-C and Jello-NC also include the impact of coordination protocol overhead and spectrum sensing errors. For Jello-NC, we use $k = 3$ in our hardware implementation.

6.2.1 Jello's Overall Performance

Media Quality Measurements. Figure 9 summarizes the application disruption rates using both video and synthetic traffic. Due to channel impairments, all disruption rates are slightly higher than those of simulations. We see that Jello-NC can effectively utilize a large portion of the spectrum (up to 75%) while keeping disruption rates below 5%. It outperforms Static and Jello-C significantly and is within a reasonable distance from Optimal.

Jello-C also outperforms Static, except in the VBR case when the traffic load is lower than 68%. This unexpected degradation comes from Jello's coordination overhead, hardware artifacts (discussed in Section 5.3) and sensing errors (recall that Static has no such overhead). As the traffic load grows, the gain of dynamic spectrum multiplexing overcomes the system overhead. For On/Off traffic, Jello-C consistently outperforms Static. This is because traffic burstiness is higher than that of the VBR traffic, thus dynamic spectrum access leads to significant gains.

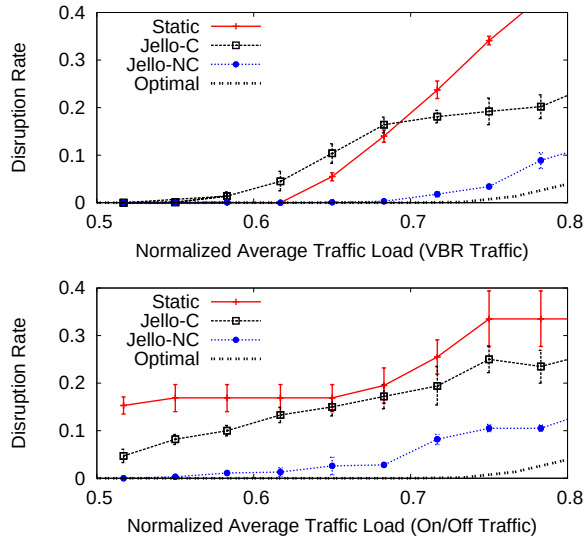


Figure 9: Testbed results: application disruption rate vs. average traffic load. Jello-NC consistently outperforms Jello-C and Static, and is within a small gap from Optimal.

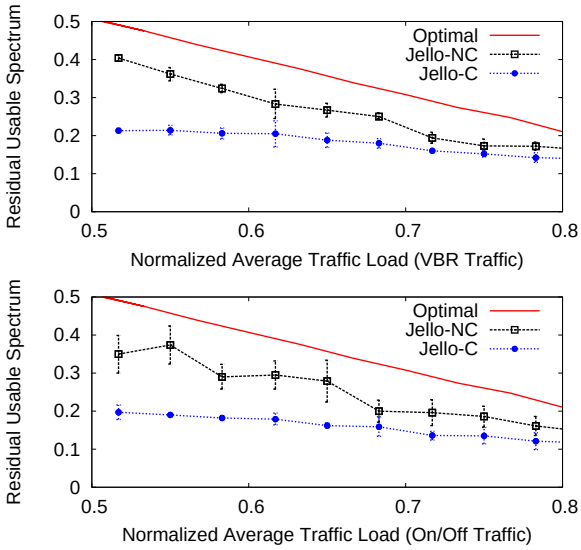


Figure 10: Testbed results: comparing Jello-C, Jello-NC and Optimal in terms of the residual usable spectrum.

Spectrum Usage Efficiency. As another measure of Jello's spectrum usage efficiency, we measure the *residual usable spectrum* as a function of the normalized average traffic load. Figure 10 shows the results for Jello-C, Jello-NC and Optimal. The result of Statis is not shown because the entire spectrum is used by existing sessions.

Compared to Optimal which completely removes all fragments, Jello-NC only sacrifices 10-15% of the total spectrum bandwidth. Among those, 3% comes from the extra guard bands associated with the non-contiguous frequency access (due to infrequent defragmentation), and the rest is from sensing errors and the fact that each new flow can only at most 3 frequency blocks.

For Jello-C, however, the overhead increases to 20-30% of the total spectrum bandwidth. In this case, the impact of residual fragmentations is amplified by the limitation that each new flow can only use 1 frequency block. For the same reason, its residual spectrum is insensitive to variations in traffic loads. An alternative way to interpret the results is that, compared to Jello-C, Jello-NC offers up to 45% more free spectrum to new sessions.

6.2.2 Where Does The Gain Come From?

The improvement of Jello comes from both non-contiguous spectrum access and online defragmentation. In the following, we evaluate their gains separately by comparing the performance of Jello with contiguous and non-contiguous frequency access, and by enabling and disabling online fragmentation.

Benefits from Non-contiguous Frequency Access. For a fair comparison, we assume both access mechanisms use online defragmentation. Figure 9 already shows that allowing non-contiguous access keeps the disruption rate below 10%, while contiguous access may suffer more than 25% disruptions. Another way to interpret the result is that, to keep a 10% or less disruption, non-contiguous access achieves 22–32% improvement in spectrum utilization over contiguous access.

Benefits of Online Defragmentation. Using On/Off traffic, we compare the performance of Jello with and without online spectrum defragmentation. From Figure 11, we see that online defragmentation reduces spectrum disruptions for both contiguous and non-contiguous Jello. For example, with 68% load, defragmentation reduces disruptions from 18% to 15% for contiguous and 5% to 3% for non-contiguous access. Compared to enabling non-contiguous access, online defragmentation has a smaller gain. This is because in our implementation, Jello devices defragment infrequently (at most twice per On period) due to hardware limitations.

6.2.3 Jello's Overhead

Having examined Jello's application-level and spectrum usage performance, we now look into the overhead that separates Jello-NC from Optimal. We quantify the impact of each element that contributes to Jello-NC's application disruption rate. These include: (1) the inherent traffic dynamics where the total spectrum cannot support all the sessions (the same applies to Optimal); (2) the frequency guard band overhead from non-contiguous

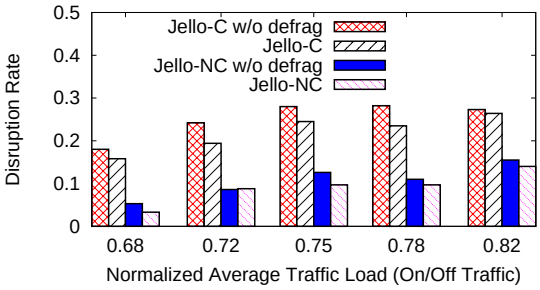


Figure 11: Testbed results: benefits of Jello's online defragmentation using On/Off traffic.

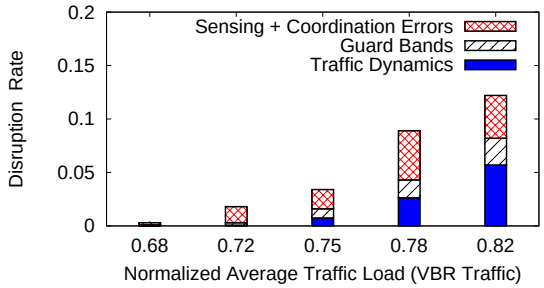


Figure 12: Testbed results: breakdown of contributions in Jello-NC's disruptions, using video traces. The impact of traffic dynamics is unavoidable and also applies to Optimal.

frequency access (for being unable to defragment spectrum completely); (3) the sensing error and coordination overhead caused by channel impairments, and (4) conflicting defragmentation. Results in Figure 12 show that the guard band overhead has a relatively small impact compared to the other two, which confirms that Jello produces a very low-level of spectrum fragmentation. The probability of defragmentation conflicts is 0.5% in our experiments and its impact is absorbed in the coordination errors in Figure 12.

Frequency Guard Bands. In Figure 13(a), we compare Jello-NC and Optimal in terms of their guard band overhead. Without any fragment, Optimal uses a fixed number of guard bands (6 subcarriers out of 240 usable subcarriers), or a 2.5% of overhead. For Jello-NC, the guard band overhead increases with the traffic load because each session uses more frequency blocks to fulfill its demand. However, similar to the results in Table 1, in our experiments more than 85% of time a session uses only 1 or 2 frequency blocks. Thus the overall guard band overhead is less than 5% even at 80% traffic load.

Spectrum Sensing Errors. Figure 12 shows that sensing errors could be a major contributor to the disruptions. In our current implementation, the average false positive (treating available blocks as occupied) and false negative (treating occupied blocks as available) rates are 5–10%. Figure 13(b) shows the results of two sample topologies. These errors are due to the time-varying channel impairments and heterogeneous signal strengths commonly found in indoor environments.

On the other hand, Jello's edge-detection based sensing is much more accurate than the energy-detector, and is relatively insensitive to the choice of detection threshold. To quantify this benefit, we plot in Figure 14 the detection false positive and false negative rates from energy-detection based sensing, using the same topologies in Figure 13(b). We see that energy-detection sensing leads to much higher detection errors, and is highly sensitive to the choice of its detection threshold (-32dB for topology 1, -48dB for topology 2). In addition, it suffers from high false positives (e.g. 40%) in order to maintain a reasonable rate of false negatives (e.g. 10%).

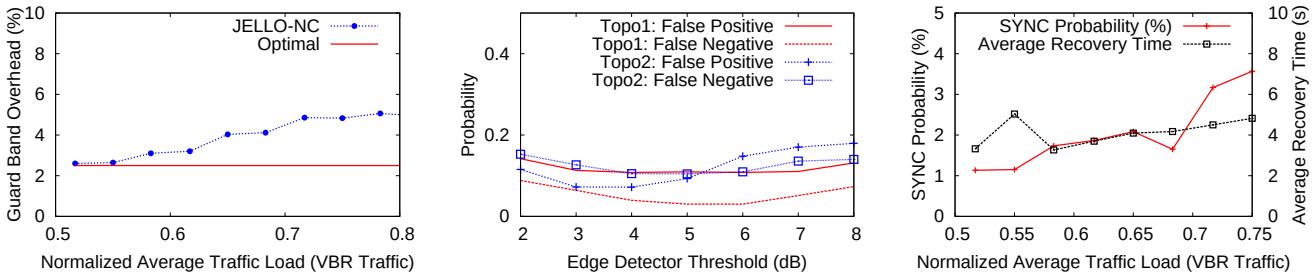
Coordination Overhead. The majority of Jello's coordination overhead is due to links falling back to the SYNC state to resynchronize. In our experiments, these occur from external interference, or an unlikely conflict in frequency adjustments. From Figure 13(c), we see that the probability of entering SYNC is only 2-3%, and the average recovery time is 4-5s. Both the SYNC probability and the recovery time increase with the traffic load because as more sessions start to adapt frequency for additional spectrum, they create slightly more conflicts and more traffic on the SCS. Now a session could wait longer before starting resynchronization. However, because links leave the SCS immediately after locating free spectrum, the SCS utilization stays low.

7 Discussion

We can extend Jello in the following directions.

Integrating with Other MAC Functions. Due to USRP Radios' large processing delay, current Jello implementation does not include several MAC functions. These include (1) rate adaptation (Jello uses BPSK); (2) channel-aware frequency selection (in our experiments the channel quality is flat across the frequency range due to limited bandwidth); (3) power control (we use uniform transmit power across all the subcarriers in use); and (4) packet retransmission. Using powerful radio platforms, Jello can add these functions. A key issue is to investigate the interaction between Jello's frequency selection and these functions and to jointly optimize them together.

Optimizing Frequency Selection. Jello's frequency selection algorithms focus on minimizing network-wide spectrum fragmentation and conflicts. Additional information about each spectrum section such as received signal strength can allow Jello to choose a good set frequency blocks to achieve reliable transmissions match-



(a) Overhead from Frequency Guard Bands (b) Sensing Errors (c) Coordination Overhead

Figure 13: Testbed results: Examining Jello-NC's overhead in terms of the frequency guard band overhead, sensing errors, and coordination delay.

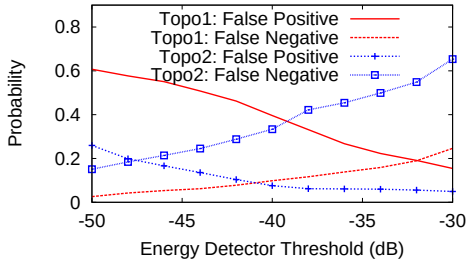


Figure 14: Testbed results: detection reliability of energy detection-based sensing as a function of its detection threshold. Compared to Jello's edge-detector, it leads to much higher detection errors, and is highly sensitive to the choice of its detection threshold (-32dB for topology 1, -48dB for topology 2).

ing its traffic demand and minimize frequency usage. Jello can also use this information to configure a proper amount of guard bands at link boundaries instead of using a uniform configuration. An interesting issue is how to obtain such information reliably and efficiently.

Porting Jello to Other Radios. Jello can be ported onto advanced hardware platforms [16, 21, 24, 28, 30], to benefit from their increased frequency bandwidth and processing speed. For best performance, Jello requires radios that can support fine-grained frequency access, and quickly scan spectrum to identify available ranges.

8 Related Work

We divide the related work into two categories: contiguous and non-contiguous frequency access.

Contiguous Frequency Access. The majority work on dynamic spectrum networks assumes contiguous frequency access [1, 4, 8, 16, 20, 31, 32]. This access pattern has the advantage of being readily implemented on conventional 802.11 devices [8]. In this context, prior works have developed centralized algorithms for load balancing [20], distributed protocols for spectrum contention [32] and for utilizing UHF whitespaces [4].

Jello differs from these works in three aspects. First, Jello's per-session FDMA design is more general in that it operates across wider spectrum ranges at a fine granularity and completely eliminates CSMA traffic contention. Second, unlike [32], which requires a separate control radio to reserve spectrum, Jello devices self-sense spectrum to avoid access conflicts, and defragment spectrum while staying transparent to others. As a result, Jello provides dedicated frequency usage to demanding applications. Finally, Jello's spectrum sensing differs from SIFT [4] which detects any contiguous frequency usage using time-domain signals. Instead, Jello uses wide-band sensing in the frequency domain that can quickly identify multiple active frequency blocks instead of single blocks at a time.

Non-contiguous Frequency Access. Most works in this area assume either centralized control [23] or a dedicated radio for control. Others are limited to simulations [7] without considering practical artifacts such as sensing and guard bands. Jello, on the other hand, implements distributed non-contiguous frequency access and deploys a USRP prototype.

SWIFT [24] is a distributed wideband spectrum access system that can use a large frequency band even when a narrowband signal is present. SWIFT nodes share spectrum in the time domain using CSMA. Jello differs from SWIFT by using per-session FDMA to avoid costly packet contentions and by using an non-intrusive edge-detection based mechanism to identify usable frequency. ODS [15] implements on-demand spectrum access using spread-spectrum codes, focusing on adapting spectrum allocation to bursty traffic. It applies a random policy for selecting codes and uses adaptive receiver feedback to regulate code allocations. Jello differs from ODS by operating in the frequency-domain, using spectrum sensing to avoid access conflicts.

9 Conclusion

Jello provides a new distributed spectrum access technique for demanding wireless applications. High-quality delay-sensitive media sessions can now access and share wireless medium in the frequency domain and adapt their spectrum usage to varying traffic demands. Jello utilizes frequency-agile radios to sense, identify and occupy unused spectrum, allowing multiple sessions to work in parallel on isolated frequencies. To maximize spectrum usage efficiency, Jello devices self-defragment spectrum on-the-fly, and scavenge multiple frequency fragments for use by single, high-speed transmissions. Jello is also MAC-agnostic and does not require any dedicated radio for control. Despite USRP radio's limited bandwidth and large processing delays, our measurements on an 8-node testbed confirm that Jello can provide reliable spectrum access for media applications and significantly improve spectrum usage efficiency.

Acknowledgments

We thank Geoff Voelker, our shepherd Venkat Padmanabhan, and the anonymous reviewers for their helpful suggestions. We also thank Peter Steenkiste and Songwu Lu for their comments on earlier versions of this work. This work is supported in part by NSF Grants CNS-0916307, IIS-0847925, CNS-0832090, CNS-0546216. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] AKYILDIZ, I. F., LEE, W. Y., VURAN, M., AND MOHANTY, S. NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer Networks Journal (Elsevier)* (2006).

[2] AMINI, P., ET AL. Implementation of a cognitive radio modem. In *Proc. of SDR* (2007).

[3] MPEG-4 and H.263 video traces for network performance evaluation. http://trace.eas.asu.edu/TRACE/trace.html/.

[4] BAHL, P., CHANDRA, R., MOSCIBRODA, T., MURTY, R., AND WELSH, M. White space networking with Wi-Fi like connectivity. In *Proc. of SIGCOMM* (2009).

[5] CANNY, J. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence 8*, 6 (1986), 679–698.

[6] CAO, L., YANG, L., AND ZHENG, H. The impact of frequency-agility on dynamic spectrum sharing. In *Proc. of IEEE DySPAN* (2010).

[7] CAO, L., AND ZHENG, H. Spectrum allocation in ad hoc networks via local bargaining. In *Proc. of SECON* (2005).

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

[9] CRAMTON, P., SKRZYPACZ, A., AND WILSON, R. The 700 MHz spectrum auction: An opportunity to protect competition in a consolidating industry. *Report for Frontline Wireless* (2007).

[10] DIGHAM, F. F., ALOUINI, M.-S., AND SIMON, M. K. On the energy detection of unknown signals over fading channels. *IEEE Transactions on Communications* (2007).

[11] FEHSKE, A., GAEDDERT, J., AND REED, J. A new approach to signal classification using spectral correlation and neural networks. In *Proc. of IEEE DySPAN* (2005).

[12] GE, F., YOUNG, A., BRISEBOIS, T., CHEN, Q., AND BOSTIAN, C. W. Software defined radio execution latency. In *Proc. of SDR* (2008).

[13] GOLDSMITH, A. *Wireless Communications*. Cambridge University Press, New York, NY, USA, 2005.

[14] GONZALEZ, R. C., AND WOODS, R. E. *Digital Image Processing (3rd Edition)*. Prentice-Hall, 2006.

[15] GUMMADI, R., AND BALAKRISHNAN, H. Wireless networks should spread spectrum based on demands. In *HotNets* (2008).

[16] GUMMADI, R., NG, M. C., FLEMING, K., AND BALAKRISHNAN, H. AirBlue: A system for cross-layer wireless protocol development and experimentation. In *MIT Report* (2008).

[17] HOU, W., YANG, L., ZHANG, L., SHAN, X., AND ZHENG, H. Understanding the impact of cross-band interference. In *Proc. of ACM Coronet Workshop* (2009).

[18] JARDOSH, A. P., RAMACHANDRAN, K. N., ALMEROTH, K. C., AND BELDING-ROYER, E. M. Understanding congestion in IEEE 802.11b wireless networks. In *Proc. of IMC* (2005).

[19] KNUTH, D. E. *The Art of Computer Programming, Vol. 1 (3rd ed.): Fundamental Algorithms*. Addison-Wesley, 1973.

[20] MOSCIBRODA, T., CHANDRA, R., WU, Y., SENGUPTA, S., BAHL, P., AND YUAN, Y. Load-aware spectrum distribution in wireless LANs. In *Proc. of ICNP* (2008).

[21] NYCHIS, G., SESHAN, S., STEENKISTE, P., HOTTELIER, T., AND YANG, Z. Enabling MAC protocol implementations on software-defined radios. In *Proc. of NSDI* (2009).

[22] OPPENHEIM, A. V., SCHAFER, R. W., AND BUCK, J. R. *Discrete-time signal processing (2nd ed.)*. Prentice-Hall, 1999.

[23] RAHUL, H., EDALAT, F., KATABI, D., AND SODINI, C. Frequency-aware rate adaptation and MAC protocols. In *Proc. of MobiCom* (2009).

[24] RAHUL, H., KUSHMAN, N., KATABI, D., SODINI, C., AND EDALAT, F. Learning to share: narrowband-friendly wideband networks. In *Proc. of SIGCOMM* (2008).

[25] SINGH, S., ACHARYA, P. A. K., MADHOW, U., AND BELDING-ROYER, E. M. Sticky CSMA/CA: Implicit synchronization and real-time QoS in mesh networks. *Ad Hoc Network 5*, 6 (2007), 744–768.

[26] STOCKHAMMER, T., HANNUKSELA, M. M., AND WIEGAND, T. H.264/AVC in wireless environments. *IEEE Trans. on Circuits and Systems for Video Technology 13*, 7 (2003), 657–673.

[27] SUN, Y., SHERIFF, I., BELDING-ROYER, E. M., AND ALMEROTH, K. C. An experimental study of multimedia traffic performance in mesh networks. In *Proc. of WiTMeMo* (2005).

[28] TAN, K., ET AL. SORA: High performance software radio using general purpose multi-core processors. In *Proc. of NSDI* (2009).

[29] TIMOTHY M., S., AND DONALD C., C. Robust frequency and timing synchronization for OFDM. In *IEEE Transactions on Communications* (1997).

[30] Wireless open-access research platform. http://warp.rice.edu/.

[31] YUAN, Y., BAHL, P., CHANDRA, R., MOSCIBRODA, T., NARLANKA, S., AND WU, Y. Allocating dynamic time-spectrum blocks in cognitive radio networks. In *Proc. of MobiHoc* (2007).

[32] YUAN, Y., ET AL. KNOWS: Kognitiv networking over white spaces. In *Proc. of IEEE DySPAN* (2007).