Spectrum Virtualization Layer

Kun Tan, Yong He, Haichen Shen, Jiansong Zhang, Yongguang Zhang
Microsoft Research Asia
Jan 31, 2011

ABSTRACT

This paper presents an architecture to support *dynamic spectrum access (DSA)* in general wireless networks. Our architecture advocates a new *spectrum virtualization layer (SVL)*, directly below the wireless physical layer (PHY or baseband processing). We call it *Layer 0.5*.

SVL presents a *virtual baseband* to traditional wireless PHY, which is designed for a fixed contiguous spectrum band with fixed width. When spectrum allocation changes under DSA, SVL performs real-time reshaping of baseband signals so that the baseband of traditional wireless PHY maps to the current physical spectrum band allocation, which can be of different width, or even in separate non-contiguous bands. All operations inside SVL are *agnostic* and *transparent* to upper PHY, making it possible to enable DSA without changing the traditional PHY designs.

We have implemented a prototype of SVL on a software radio platform and demonstrated the flexibility and effectiveness of SVL in supporting DSA. We believe this is the right architecture model for enabling DSA for general wireless networks.

1. INTRODUCTION

There is an upsurge of interests in dynamic spectrum access (DSA) recently. Unlike traditional wireless devices that operate on fixed spectrum assignments, DSA requires them to be dynamically reconfigurable in a wide spectrum range according to the network or environmental status. It is widely believed that DSA is the key to solve the existing spectrum inefficiency problem [3]. One example of DSA is White-Space networking, where wireless devices opportunistically use spare TV channels for data communication [4]. Recent work further demonstrates DSA-enabled PHY can significantly improve the communication efficiency in wireless LANs [13, 16], and suggests DSA to be a desirable feature in all wireless networks.

It is a rather challenging task to enable DSA under current wireless network designs. First, DSA discards the widely adopted design principle of having fixed channels. Instead, each device must now handle any possible time/space varying spectrum allocation, which includes changing frequency, varying width, and non-contiguous bands [7]. This presents a substantial challenge to any physical layer (PHY) design that assumes the traditional fixed spectrum model. Although several vertically integrated designs have been proposed to

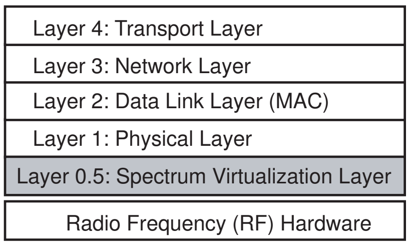

Layer 4: Transport Layer
Layer 3: Network Layer
Layer 2: Data Link Layer (MAC)
Layer 1: Physical Layer
Layer 0.5: Spectrum Virtualization Layer
Radio Frequency (RF) Hardware

Figure 1: Layer 0.5: Spectrum Virtualization Layer.

support such flexible spectrum bonding [13, 16], modifying every possible PHY design case-by-case is not a scalable solution. Further, when multiple wireless designs coexist on a single device, there isn't a good way to coordinate the possibly overlapping spectrum management and access functions implemented separately in each wireless PHY/MAC, especially if they are on the same spectrum band (like WiFi and ZigBee over 2.4GHz ISM band). Ideally, we desire a general architecture solution where DSA can be supported in a single place and enabled for all common wireless systems.

In this research, we investigate such an architecture solution. Our approach is to "virtualize" a non-variant spectrum band out of the dynamic changing spectrum allocation in DSA networks. Thus, we can support various wireless PHY without need to change their design. This can be done in current network architecture by inserting a layer below the traditional wireless PHY layer (*i.e.*, the baseband processing), It intercepts and rewrites digital signals passing between the baseband and the radio frequency (RF) front-end hardware, in both send and receive directions, to hide the spectrum dynamics and to create the effect of a fixed spectrum. We call this the *Spectrum Virtualization* approach and the new layer the *Spectrum Virtualization Layer* (or *SVL*). Since it is beneath Layer 1, the traditional wireless PHY layer, we call it *Layer 0.5* in the standard network layer architecture (Figure 1).

Contrast to the approach that embeds and distributes DSA function inside each wireless PHY/MAC, SVL is the central place to manage spectrum usage and enforce the allocation results. SVL provides a *virtual spectrum band* to each PHY, and derives a spectrum mapping table from spectrum management policies and the dynamic allocation results. At the sending side, when PHY generates digital baseband waveforms, SVL *reshapes* these samples into a different waveform silhouette such that, when RF front-end transmits them as usual, the outcome radio frequency signals will match the designated spectrum allocation. At the receiving side, SVL performs inverse reshaping on the baseband samples received from RF front-end before passing on to PHY.

The signal reshaping operation contains a combination of digital signal processing (DSP) algorithms to support flexible spectrum mapping, including filtering, frequency shifting, signal decomposition and recomposition, bandwidth and sample-rate adjustment, *etc.* (detailed in Section 4). It can map the virtual spectrum to a physic spectrum with different width, as well as multiple non-contiguous spectrum bands.

SVL further allows multiplexing of several wireless PHYs onto single RF hardware in the same node. This helps to reduce the number of wide-band RF front-end needed, and to simplify the radio design and multi-radio integration on mobile devices.

We have implemented a prototype of SVL on a high performance software radio platform [15]. We evaluate our SVL implementation under three widely used wireless PHYs, namely ZigBee, 802.11b, and 802.11g. We demonstrate the power of SVL by showing a DSA network where ZigBee, 802.11b, and 802.11g nodes can co-exist and coordinate to maximize spectrum efficiency.

In summary, the paper makes following contributions: (1) We propose an architecture to support DSA for general wireless designs. Specifically, we define the new Spectrum Virtualization Layer located at Layer 0.5 that enables dynamic spectrum management and access for general PHY without modifying the PHY design. (2) We design novel signal reshaping process for SVL that supports flexibly mapping between virtual and physical spectrum without knowing the detailed modulation schemes inside PHY. (3) We present a novel way to multiplex multiple PHY on single RF front-end. (4) We demonstrate the feasibility of SVL with our implementation on a software radio platform, and evaluate its performance. We strongly believe that SVL is the right layer to implement and enable DSA once for all.

The rest of the paper is organized as follows. Section 2 provides some background on radio transceiver designs and our motivation. We present the SVL architecture in Section 3. The detailed designs are discussion in Section 4. After describing the implementation of a SVL prototype using a software radio platform in Section 5, we evaluate its performance in Section 6. Section 7 discusses a few applications for SVL. Finally, Section 8 discusses related work and Section 9 concludes.

2. BACKGROUND AND MOTIVATION

2.1 Digital Transceiver and DSA

A radio transceiver is generally divided into two components: the radio frequency (RF) front-end and the baseband processing unit. In modern radio design, the baseband processing is generally performed in digital domain with digital signal samples, and the RF front-end mainly contains analog radio circuitry. Thus, analog-to-digital and digital-to-analog

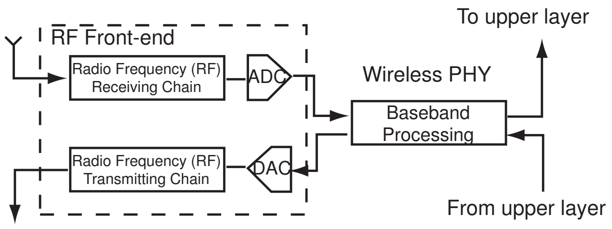


Figure 2: The architecture of a wireless transceiver.

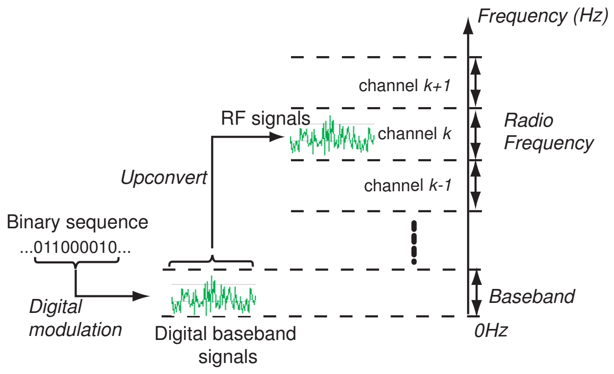


Figure 3: The baseband unit modulates binary sequence into digital baseband signals and the RF front-end further converts them to high frequency analog RF signal.

conversion (ADC/DAC) form the nature interface between the baseband unit and the RF front-end, as shown in Figure 2.

As shown in Figure 3, The baseband unit translates information bits into digital baseband waveforms[1], and vice versa. This function is also referred as *digital baseband modulation* (or *digital modulation* for short). Digital modulation maps a binary sequence to a segment of digital waveform samples, called *symbols*. At the receiver side, these symbols are *demodulated* to retrieve the embedded binary information. The baseband signals are not suitable to transmit directly. Thus, the RF front-end will convert the digital baseband samples into high-frequency analog radio signals. When receiving, the RF front-end selects the desired radio frequency signals, down-converts, and digitizes them to digital baseband samples.

In current radio systems, radio frequency is divided into fix-size channels that equals to the baseband width at predetermined central frequency. The baseband signal may be mapped to one of pre-defined channels, but it cannot change the bandwidth or pick up an arbitrary central frequency.

Such fixed spectrum management shows inefficiency in spectrum utilization in many cases. One such case is when there are multiple heterogenous wireless systems with different channel width, sharing the same spectrum band (like 802.11 and ZigBee in 2.4GHz ISM band). When a narrow-band radio is transmitting inside a channel of a wide-band radio, the wide-band radio has to keep silence not to interfere with the on-going narrow-band transmissions, even though

[1]In communication system, we call the frequency band from close to zero to a cut-off high frequency as *baseband*.

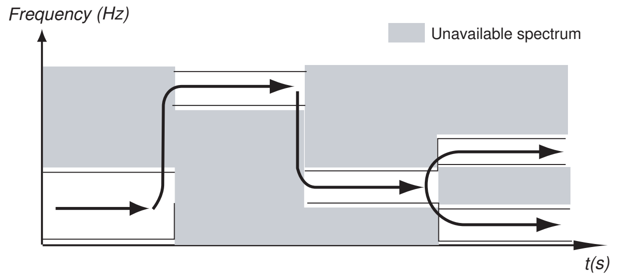


Figure 4: Dynamic spectrum access. Communication frequency bands may change dynamically according to the spectrum availability as well as application requirements.

only a small portion of wide-band channel is in use. Another example is TV White-Space networking. Due to the existence of primary users (incumbents), the available spectrum for a secondary network is varying both with time and space. It is often fragmented into narrow spectrum holes, since the incumbents may operate anywhere in the TV band. Traditional wireless system design fails to utilize these fragmented spectrum holes well [4]. Finally, previous research [14, 16] also shows that spectrum used by a radio should commensurate with the application traffic requirement. Otherwise, the contention overhead may be significantly enlarged and eventually reduce the efficiency.

It is well-recognized that to improve the spectrum usage efficiency, it is necessary to embrace the *dynamic spectrum management and access*. DSA removes the concept of fixed channels. Instead, radio frequency allocated to in a wireless system can be of variable width, at different central frequency, or even non-contiguous, based on the spectrum availability and the application requirements (Figure 4). To efficiently utilize these varied RF bands, it requires wireless PHY to generate matched baseband waveform as well. One direct approach is to modify current wireless PHY design to be DSA-capable. Prior work [13, 16] has presented several DSA systems based on this approach. However, due to the heterogenous nature of different wireless designs, such a case-by-case modification to each PHY would be a non-trivial effort.

2.2 Heterogeneity of Wireless PHY

Different wireless PHY may have completely different ways to modulate baseband signals. Largely speaking, the baseband modulation can be classified into *single carrier modulation (SCM)* and *multi-carrier modulation (MCM)*.

Traditional wireless systems modulate information only on one carrier tone (Figure 5a). Thus, symbols are concatenated in time domain. Spectrum spreading techniques are commonly used with SCM to spread symbols into a wider spectrum band to achieve better diversity as well as handle *inter-symbol interference*. SCM is usually simple and reliable even in very noisy channel. But the spectrum spreading reduces the efficiency, and thus results in low communication speed. ZigBee, 802.11b and WCDMA [1] are typical

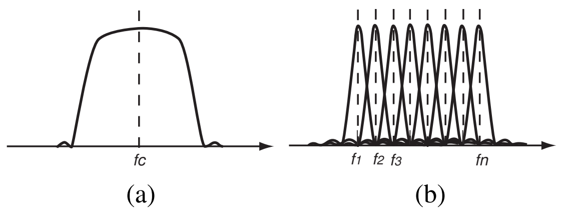


Figure 5: (a) Single carrier modulation; (b) Multi-carrier modulation (OFDM).

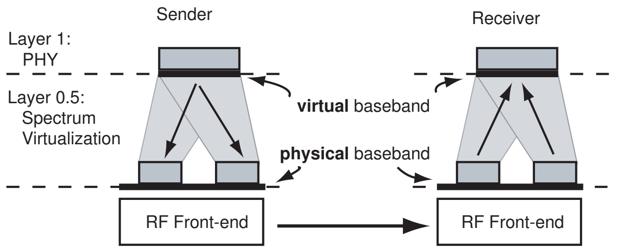


Figure 6: Spectrum virtualization.

examples of single carrier systems.

On the other hand, modern high-speed wireless PHY use MCM due to its ability to achieve high spectrum efficiency (Figure 5b). Unlike SCM, MCM divides the whole band into a number of sub-channels, and modulates a binary stream on one sub-carrier in each sub-channels. These parallel modulated signals are mixed and transmitted simultaneously. In MCM, sub-carriers are usually selected to be orthogonal, so that FFT/iFFT can be used to efficiently generate the finally modulated signals. Many high-speed wireless systems like 802.11a/g and LTE [2] adopt multi-carrier modulation.

Wireless PHY also differ from one another in the way how they handle multi-path fading. While Rake-receiver is commonly used for spreaded SCM signals, MCM relies on cyclic-prefixes (CP) to remove the impacts of multi-path fading. The choice of CP further depends on the design working range for the wireless system. Longer range suffers longer multi-path delays and calls for longer CP, but longer CP may lead lower efficiency because it is an overhead. Evidently, 802.11a and LTE have largely different PHY specifications even though they are both OFDM.

The fundamental tradeoffs in wireless PHY design makes it unlikely to have a one-size-fit-all PHY. We expect to continue seeing many different designs to accommodate various wireless applications. As a consequence, it is desirable to have a general architecture to manage and support DSA for all these wireless designs.

2.3 Spectrum Virtualization

One way to support DSA in today's wireless network architecture is to create a new layer, the *Spectrum Virtualization Layer* (SVL), below the traditional PHY (baseband processing). We term this new layer as *Layer 0.5* (Figure 6).

The goal of this layer is to bridge the gap between the traditional PHY, which is usually designed for a fixed frequency band with fixed bandwidth, and the desirable base-

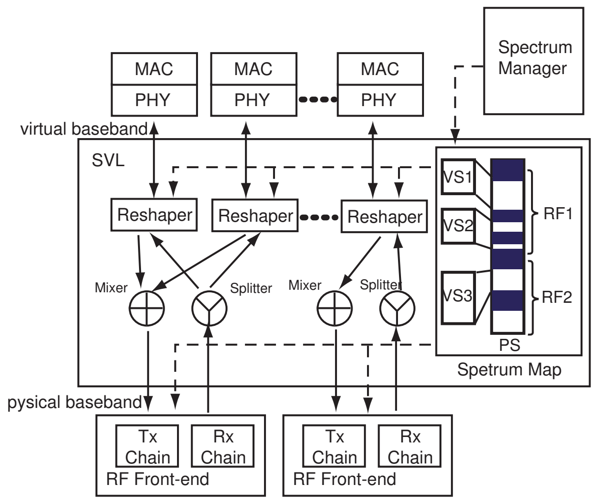


Figure 7: Architecture of Spectrum Virtualization Layer. SVL maintains a mapping between virtual baseband and physical spectrum bands, and dynamically reshapes signals between virtual and physical spectrum. Gray portion of the spectrum map shows the unavailable spectrum bands. The solid arrows show the data paths; while the dashed arrows show the control paths.

band under DSA, which can have any time and space varying spectrum configuration, often over a wider band. The basic idea is the following. First, we decouple the traditionally tight connection between PHY and RF front-end and add a layer of indirection. Here, the baseband that PHY operates on is the *virtual baseband* and the baseband used for the RF front-end is the *physical baseband*. The two differ in the sense that one is fixed and specified by PHY design and the other is dynamic and determined by some unspecified DSA allocation method. At the sender side, PHY generates digital waveforms as if it were connected to a RF front-end. The SVL layer intercepts these samples and *reshapes* them into a different waveform shapes so that, when RF front-end transmits them as usual, the outcome radio signals would match the current spectrum allocation. At the receiver side, SVL performs the inverse reshaping operation on the physical baseband samples to recover the original digital waveform shapes for the PHY layer. For example in the Figure 6, a wider band virtual baseband sample is reshaped into two narrower spectrum band samples and back.

SVL provides an architecture support for DSA for any current wireless PHY. It hides all complex dynamic spectrum management and access from wireless PHY, and become a unique narrow waist to manage spectrum and enforce its usage.

In the next section, we overview the architecture and design principles of SVL. Section 4 discusses the technical details.

3. OVERVIEW AND ARCHITECTURE

The overview architecture of SVL is shown in Figure 7. SVL provides a *virtual baseband* to each PHY and dynamically translate the signals between virtual baseband and the physical baseband on the RF front-end. The width of the virtual baseband is specified by each PHY during the initialization stage. Hereafter in this paper, when we call *baseband*, we refer to this the virtual baseband; We also use the term "*physical spectrum band*" (or simply "*physical band*") to denote a portion of spectrum on a RF front-end's physical baseband that is allocated to the PHY.

SVL maintains a *spectrum map* between virtual baseband and physical spectrum bands. The mapping is quite flexible in SVL. For example, it can map the baseband to a physical spectrum band with the same width (*e.g.*, *VS1* in the figure). Alternatively, it can map the baseband to a narrower contiguous physical band, or several non-contiguous physical bands (*e.g.*, *VS2* or *VS2* in the figure). How spectrum is allocated is controlled by *Spectrum Manager*.

The function of the spectrum manager is to monitor the current spectrum usage (*e.g.*, by sensing or querying a database), allocate free physical spectrum bands for a PHY based on various policies, and update the spectrum map in SVL. There is already abundant work on the dynamically spectrum management, like [4,6,10,16]. In this paper, we do not explicitly address how spectrum should be allocated among different PHY, which is still an active research area. Instead, we assume SVL already has such a spectrum map, and our focus is how to enforce the *spectrum access* of baseband signals to match the physical spectrum specification.

It is worth noting that the separation between dynamic *spectrum management* and *spectrum access* in SVL is analogous to the separation of *packet routing* and *packet forwarding* in the conventional network router design. Such a separation is both efficient and flexible. *Spectrum access* makes decisions and operate on every PHY frame and therefore should be simple and highly optimized for efficiency. While *spectrum management* may involve complex algorithms and interactive signaling protocols to decide spectrum allocation. It should situate outside the critical path, *e.g.* as a separated user-mode daemon, but be allowed to modify the spectrum mapping table dynamically.

The core and challenging part of SVL is to design the *signal reshaper* that translates signals from baseband to physical bands, and vice versa. The reshaper must satisfy the following requirements.

First, the reshaper should be *PHY agnostic*. To support heterogenous wireless PHY, the reshaper must perform signal translation without knowing the specific modulation scheme. It implies that the reshaper can only adopt general digital signal processing algorithms that operate on general baseband waveform.

Second, the reshaper should be *transparent* to upper layer PHY. That means, although the reshaping operation may change the baseband waveform in some way, the upper layer PHY should not be able to tell whether this distortion is coming from the reshaping operation or it is due to a nat-

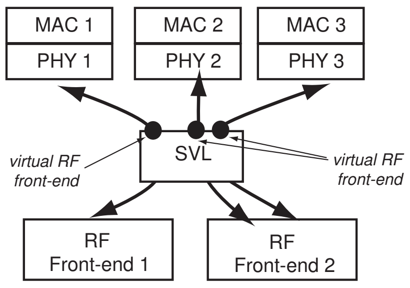


Figure 8: RF front-end virtualization.

ural wireless channel fading. This is to ensure the distortion caused by reshaping can be modeled by an equivalent multipath fading channel, and therefore can be handled with the equalization mechanisms in current PHY.

Finally, the reshaper should be as *simple* as possible, introducing minimal overhead. For example, it should not try to equalize and recover exactly the original transmitted baseband waveforms, because of following two reasons: (1) It may add considerable overhead by sending training symbols; (2) It largely overlaps with similar functions in PHY. Therefore, it can be either redundant or premature, since only the PHY knows the best way to handle fading, based on its modulation scheme as well as application requirements.

We will elaborate our reshaper design in Section 4.1 that satisfies all above three requirements. Our reshaper reveals all distortion of baseband waveform to upper layer PHY and relies on their own ability to correct them.

After reshaping, baseband signals are converted to physical baseband signals. Physical baseband signals from multiple PHY may be mixed (added) together before sending to the RF front-end. When receiving, the incoming signals are passed to a *splitter*, which contains a matched filter for each PHY based on its spectrum map. The filtered physical band signals are fed to the reshaper again, where the inverse operations are perform to recover the baseband signals. Then, baseband signals are sent to wireless PHY, which will further demodulate them and obtain the binary information.

Conceptually, SVL also *virtualizes* RF front-ends for each wireless PHY, with the *mixing* and *splitting* operations. As illustrated in Figure 8, SVL can flexibly map different PHY to different RF front-ends; Also, it multiplex several PHY onto single RF front-end hardware. *RF front-end virtualization* is particularly helpful in DSA networks, as it allows multiple PHY to share a common powerful wide-band agile RF front-end. Thereby, it reduces the required resource for multi-radio integration in terms of space, energy as well as price, on mobile devices.

Next, we outline the Layer 0.5 interfaces to PHY, and we describe in details of SVL operations in Section 4.

3.1 Layer 0.5 Interface

SVL defines three clean and simple interfaces described as follows. Note that for clarity, we use C Language syntax to describe these interfaces, but this does not imply they can only be implemented in C and in software only.

a) Registration. A wireless PHY should register itself with SVL before it can send and receive signal samples. The *registration interface* is shown as follows:

```
vs_handle svl_register ( int bandwidth,
                         int over_sample,
                         int carrier_num,
                         policy * pp );
```

svl_register defines a virtual spectrum band. The *bandwidth* defines the desired width for the baseband. The *over_sample* parameters, together with *bandwidth* determines the sampling rate of baseband. As discussed earlier, the sampling rate should be at least twice of the bandwidth to satisfy Nyquist criteria. So *over_sample* is at least two. It is possible for PHY to specify a higher over-sampling rate, which may give better performance but at a cost of computation as more samples need to be processed in a given interval. *Carrier_num* defines number of sub-carriers used in multi-carrier modulation; For single carrier modulation, *carrier_num* should be equal to one. As we will explain later, SVL needs this information to determine the precision in signal decomposition (Section 4.1).

The last parameter specifies a hint to SVL on what sorts of reshape operations can and should be performed on the baseband signals of this PHY. This parameter will work together with the spectrum manager to determine the mapping between virtual baseband and physical spectrum bands. Table 1 lists some example policies that may be specified by PHY.

SVL will forward the registration request to the spectrum manager. If accepted, the spectrum manager will allocate a portion of physical spectrum and update the map entries in the spectrum map. SVL will return a handle to identify the registered virtual baseband.

Table 1: Example policies specified in SVL.

Policy	Meaning
SVL_VS_NO_SHRINK	Do not shrink baseband signals to a physical band narrower than the specification.
SVL_VS_NO_SPLIT	Do not split baseband signals into non-contiguous physical bands.
SVL_VS_GUARD_BAND	Specify the additional guard-band size at the edge of PHY.

b) Send symbol. A wireless PHY should call *send interface* to output baseband signals to SVL. The *send interface* has the following format:

```
int svl_send_symbol ( vs_handle vs_id,
                      COMLEX_SAMPLE * samples,
                      int length );
```

Wireless PHY calls *svl_send_symbol* to output a baseband *symbol* to SVL. It takes an identification to the virtual baseband, which is used to lookup the spectrum map for the physical spectrum. The parameter *samples* and *length* specify the pointer to the location and the number digital samples to send, respectively.

c) Receive symbol. A wireless PHY should call *receive interface* to receive a symbol worth of baseband samples from SVL. The *receive interface* has the following format:

```
int svl_receive_symbol ( vs_handle vs_id ,
                         COMLEX_SAMPLE * samples ,
                         int length );
```

Similarly, *svl_receive_symbol* takes an identification to the virtual baseband, the pointer to the sample buffer location and the number of digital samples to be received. It returns the actual samples that is passed to PHY.

It should be noted that these interfaces are very general. Only a sequences of time-domain digital samples are passing through the interface between PHY and SVL. Therefore, any PHY can be easily integrated with SVL using these interfaces.

4. SVL OPERATIONS

4.1 Signal Reshaping

The reshaper is the core functional module in SVL that performs digital signal processing to translate signals between virtual baseband and physical bands. Figure 9 shows the processing blocks inside the reshaper.

There are three main operations: *signal decomposition/recomposition*, *bandwidth and sampling rate adjustment* and *frequency band shifting*. Signal decomposition may split a virtual baseband signal into several sub-streams, each of which can be mapped to a separated physical band. At the receiver side, these sub-streams are collected by *recomposition* and converted back to virtual baseband signals. The signal decomposition and recomposition are two core operations to support non-contiguous bonding of physical bands. The *bandwidth and sampling rate adjustment* may change the signal bandwidth and match the signal sampling rate to the RF front-end. This operation allows a virtual baseband to be mapped to a physical band with different width. Finally, the *frequency band shifting* module will move the central frequency of baseband signal ($0Hz$) to that of the physical band when transmitting, and move it back to zero when receiving.

In following discussion, we refer a spectrum band, B, as a set of frequency, which can be represented by its central frequency f and pass-band width b. Therefore, we denote $B(f, b)$ as a frequency band, whose range is $(f - \frac{b}{2}, f + \frac{b}{2})$. We also denote $\Theta = \{B_{p,i}(f_i, b_i)|i = 1..k\}$, the set of allocated physical bands to the PHY.

We often use b_s to denote the *aggregated bandwidth* of all

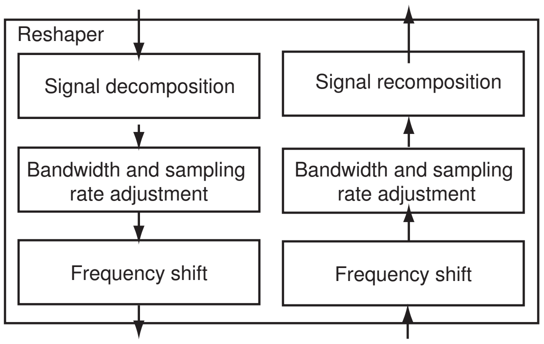


Figure 9: Processing blocks of the reshaper in SVL.

allocated physical bands in Θ, *i.e.*,

$$b_s = \sum b_i.$$

And we denote $B_{span}(f_{span}, b_{span})$ *the span* of Θ, which is defined as the spectrum band between the highest and lowest frequency of all physical bands. Formally, $B_{span} = (f_{low}, f_{high})$, where

$$\begin{aligned} f_{high} &= \max f, f \in B_{p,i}, B_{p,i} \in \Theta; \\ f_{low} &= \min f, f \in B_{p,i}, B_{p,i} \in \Theta. \end{aligned}$$

4.1.1 Signal Decomposition and Recomposition

There could be several ways to split a baseband signal into multiple sub-streams. For example, one naive way is the *time-domain decomposition*, which may cut signal samples into segments, *i.e.* PHY *symbols*, and map each symbol to a physical band alternatively, as shown in Figure 10(a). However, such time-domain decomposition violates the *transparency* principle as discussed in Section 3. This is because symbols passing different spectrum bands may experience completely different multi-path fading, and be distorted in different ways. This will defeat the channel estimation and equalization of existing PHY, as they always assume that all samples should pass a same *coherent* wireless medium. The erroneous channel estimation and equalization will severely reduce the demodulation performance and cause frame losses.

Instead, in this paper, we propose a frequency-domain signal decomposition based on FFT/iFFT, as shown in Figure 10(b). The core idea is to perform an M-point FFT on every PHY symbol passed to SVL. This FFT operation decomposes the time-domain signals into M frequency-domain components. Then, we can re-assign each of the M frequency component to a sub-carrier corresponding the allocated physical bands. After re-assignment, it performs an N-point iFFT to convert the frequency components back into time-domain samples. Here, the aggregated bandwidth of physical bands should be no less than virtual baseband width. Thus, N is no less than M. At the receiver side, the inverse operation is performed. This time, it will first perform an N-point FFT to convert the incoming samples into frequency components. Next, M sub-carriers corresponding

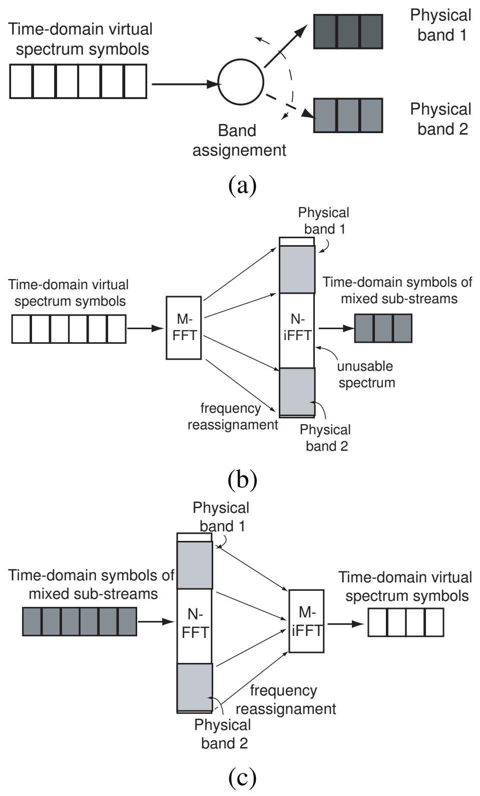


Figure 10: Signal decomposition. (a) Time-domain decomposition; (b) Frequency-domain decomposition; (c) Frequency-domain recomposition.

to the allocated physical bands are collected. Then, an M-point iFFT is performed to regenerate the baseband symbols, as shown in Figure 10(c).

The frequency-domain decomposition/recomposition retrains the transparency property of SVL. Intuitively, it ensures the components at the same frequency of different PHY symbols always pass the same sub-carrier in a physical spectrum band. Thus, it would appear to upper layer PHY as a coherent wireless channel, although the multi-path fading property may be changed by the reshaping operation. Consequently, existing channel estimation and equalization mechanisms can be used to handle this multi-path fading effect. Later in Section 4.1.4, we will give a more formal explanation.

There are several detailed issues that need to be considered.

a) How to choose M? The value M determines the resolution of our frequency decomposition. It should not be too small. For example, for a multi-carrier modulated PHY with C sub-carriers, M should not be less than C. Otherwise, we will introduce inter-carrier interference during our reshaping operations. On the other hand, if M is too large, the FFT/iFFT operation may create much overhead as the computation complex of FFT increases as $O(M \log M)$. In this paper, we take the following rule-of-thumb to pick up a proper M for frequency decomposition,

$$M = \max(C, M_{min}),$$

where M_{min} specifies the minimal resolution. We choose $M_{min} = 64$ in this paper.

b) How to choose N? We should choose N sub-carriers that are large enough to cover all the physical bands (maybe non-contiguous). We denote b_v the width of virtual baseband, b_s the aggregated bandwidth, and b_{span} the width of the span of the allocated physical bands. We only consider the case where $b_v = b_s$. Later, we will discuss the case when $b_v > b_s$ [2]. Then, N should satisfy the following condition,

$$N \geq M \frac{b_{span}}{b_v}. \qquad (1)$$

To ease the computation of FFT, we choose N as the smallest power of 2 that satisfies Equation 1.

c) How to map N sub-carriers to physical bands? Let f_{span} be the central frequency of the span (B_{span}) of allocated physical bands. We will first shift each physical band, $B_{p,i}(f_i, b_i) \in \Theta$, by $(-f_{span})$ to be $\hat{B}_{p,i}(f_i - f_{span}, b_i)$. Then, a subcarrier is *available* if it is covered by any $\hat{B}_{p,i}$. An available sub-carrier can be mapped to a frequency component of a baseband signal. It is easy to see there should be at least M available sub-carriers if Equation 1 is satisfied.

d) What if the number of samples in a symbol is different from M? It is a common case that the samples in one PHY symbol, *i.e.* K, is different from M. For single-carrier PHY, K is usually smaller than M; while for multi-carrier PHY, K may be greater due to the use of cyclic-prefix (CP). By perform M-point FFT and N-point iFFT, the decomposition enlarges the signal bandwidth by a factor of $\beta = \frac{N}{M}$. At the same time, it converts K samples of a symbol to βK samples accordingly.

If $K \leq M$, we simply pad zero to the K sample before perform M-point FFT. As we know from DSP theory, zero padding in time-domain samples does not change the frequency response of a signal. Then, after signal re-arrangement and N-point iFFT, we will obtain N time-domain samples, with which only the first βK samples are significant. So, we output only these βK samples and truncate the rest.

On the other hand, if $K > M$, we perform FFT as follows. We perform M-point FFT for every M samples until to the tail of the symbol where the remaining samples, say L samples, are less than M. Next, we perform an additional M-point FFT from the $(K - M)$th to Kth sample. Since we artificially shift FFT window by $(M - L)$ samples, it will cause a phase rotation in the frequency domain. Thus, we need to compensate it before performing N-point iFFT. The compensation is done by rotating the phase of values on corresponding sub-carriers. Specifically, if a frequency component i has been assigned to sub-carrier j, then we multiply

[2]Note that SVL shall not allocate

the sample at sub-carrier j by a factor $e^{j2\pi\frac{M-L}{M}(j-i)}$. After N-point iFFT, this last group of samples should overlap with its previous group by $\beta(M-L)$ samples. Then, we take an average for these $\beta(M-L)$ samples as output.

Similar operations are performed by recomposition at the receiver side. But with recomposition, it will reduce the signal bandwidth by β. Accordingly, it takes K samples from the physical bands and regenerates $\frac{K}{\beta}$ virtual baseband samples.

e) What if the aggregated bandwidth (b_s) of allocated physical bands is less than the virtual bandwidth (b_v)? In previous discussion, we assume the aggregated bandwidth of allocated physical bands, b_s, equals to b_v. Here, we consider the case when $b_s < b_v$. We artificially scale the frequency of these physical bands by a factor of $\alpha = b_v/b_s$. Thus, the aggregated bandwidth of the scaled physical bands, say $\hat{b}_s$, is equal to b_v. The same decomposition/recompoistion operations are performed with these scaled physical bands. Later, the *bandwidth adjustment* module will compensate this back as discussed in Section 4.1.2.

4.1.2 Bandwidth and Sampling Rate Adjustment

This module has two main tasks. First, as aforementioned, signal decomposition may scale the physical bandwidth by α to facilitate the signal splitting. This should be compensated here by reducing the signal bandwidth by α. It is done by interpolation. To understand how it works, let's assume a signal s has a bandwidth b with a sampling rate f_s. To reduce the bandwidth of s, we can add more samples to s (interpolation). For example, if we interpolate twice more samples to s, with the same sampling rate f_s, the bandwidth of s is reduced by half.

Similar, to reduce the signal bandwidth by α, we should add α times more samples. This is achieved by interpolation and decimation. Without loss generality, we assume $\alpha = k/l$, k and l are integers. This is done with following three steps:

1. Zero padding. For every sample, it pads $(k-1)$ zeros.
2. Low-pass filtering. A low-pass filter is applied the the zero-padded samples to remove the high-frequency signal image.
3. Decimation. We pick up a sample for every l samples to get the final signal.

The second task is to adjust the sampling rate to match the RF front-end. This is accomplished by re-sampling the signal with the real sampling rate of the RF front-end. The re-sampling operation is again achieved by interpolation and decimation. Let f_s be the sampling rate after the bandwidth adjustment, and f_r is the real sampling rate of the RF front-end. Let f_{LCM} be the least common multiple of both f_s and f_r. To re-sampling, we first increase the sampling rate of s to f_{LCM} by interpolation. We pad $m = (f_{LCM}/f_s - 1)$ zero samples, and low-pass filter the signal to remove imaging. Then, we decimate it by $n = f_{LCM}/f_r$ to get the final signal with desired sampling rate f_r.

We note that both bandwidth and sampling rate adjustment are actually involving same DSP operations of interpolation and decimation. So they can be combined together to save computation.

At the receiver side, the inverse operations are performed. The received signal is sampled at f_r, and it should be adjusted to match the sampling rate of the virtual baseband before sent to the *recomposition* module.

4.1.3 Frequency Band Shifting

The signal generated by N-point iFFT after signal decomposition is centered at zero. As discussed earlier, when we map N sub-carriers to the physical bands, we artificially shift the physical bands by $-f_{span}$. Here, we should compensate this back to make the signal pass the real physical bands. Shifting a signal s is to multiply a factor of $e^{j2\pi f_h i}$ to each digital sample $\{x_i\}$ of that signal. At the sender side, we shift the signal by f_{span} before sending it to the *mixer*; Similarly, at the receiver side, we shift the signal from the *splitter* by $-f_{span}$ before sending it to *bandwidth and sampling rate adjustment.*

4.1.4 Understanding Reshaping

While bandwidth adjustment, re-sampling and frequency band shifting are common used techniques for digital signal processing and may be well-understood [12], it is not obvious on the impact of signal decomposition and recomposition. In this section, we present a simple mathematical model on this. For the sake of simplicity, we only consider the case where a virtual baseband is split to two separated physical bands, say B_1 and B_2. We note the similar technique can be also applied to analyze the case with more separated bands, but we omit it due to the space limitation. Without losing generality, we assume one band is centered at f_h and the other one is centered at $-f_h$[3]. We assume $b_v = b_{p,1} + b_{p,2}$ in our analysis. Thus, the bandwidth adjustment and frequency shifting operations can be ignored.

We denote $x(t)$ as the baseband signal, and $X(f)$ is the FFT of $x(t)$. Let $X_1(f)$ contain all sub-carriers that are mapped to B_1 and $X_2(f)$ contain other sub-carriers mapped to B_2. Then, according to the linearity of FFT, we have

$$x(t) = x_1(t) + x_2(t), \tag{2}$$

where $x_1(t)/x_2(t)$ is the iFFT of $X_1(f)/X_2(f)$. To shift the frequency of a signal by f, we multiply its time-domain samples by $e^{j2\pi ft}$. Thus, the transmitted signal $x^T(t)$ is

$$x^T(t) = x_1(t)e^{j2\pi f_h t} + x_2(t)e^{-j2\pi f_h t}. \tag{3}$$

The received signal $y^R(t)$ can be modeled as

$$y^R(t) = h * x^T(t - t_0)e^{j2\pi f_\delta t}, \tag{4}$$

[3]This condition can always be satisfied by proper shifting the central frequency of these two bands.

where h models the multi-path fading channel, t_0 models the timing synchronization error, and f_δ models the frequency synchronization error.

Plugging in Equation 3 into Equation 4, we have

$$\begin{aligned} y^R(t) &= y_1^R(t) + y_2^R(t) \\ &= h * x_1(t-t_0)e^{j2\pi f_h(t-t_0)}e^{j2\pi f_\delta t} + \\ &\quad +h * x_2(t-t_0)e^{-j2\pi f_h(t-t_0)}e^{j2\pi f_\delta t}. \end{aligned} \quad (5)$$

The recomposition operation will shift frequency back for $y_1^R(t)$ and $y_2^R(t)$ to recover the baseband signal. Then, we get

$$\begin{aligned} y(t) &= y_1^R(t)e^{-j2\pi f_h t} + y_2^R(t)e^{j2\pi f_h t} \\ &= h * x_1(t-t_0)e^{-j2\pi f_h t_0}e^{j2\pi f_\delta t} + \\ &\quad +h * x_2(t-t_0)e^{j2\pi f_h t_0}e^{j2\pi f_\delta t} \\ &= h * x'(t-t_0)e^{j2\pi f_\delta t}, \end{aligned} \quad (6)$$

where $x'(t-t_0) = x_1(t-t_0)e^{-j2\pi f_h t_0} + x_2(t-t_0)e^{j2\pi f_h t_0}$.

From Equation 6, we can see a number of points. First, in the recovered baseband signal $y(t)$, all wireless channel fading and the frequency offset are preserved. This is the desire behavior of SVL. As we discussed in Section 3, we believe only the PHY itself knows the best way to handle channel distortions.

Second, the recovered signal contains a multi-path version of the original transmitted signal, *i.e.* different portions of the signal have different fading coefficients. In other words, the decomposition/recomposition may create additional multi-path fading for non-contiguous spectrum bonding compared to the contiguous case. We note such self-generated multi-path effect is essential since when using separated spectrum bands, we indeed let different portions of a signal pass different paths. This multi-path effect reduces with the decrease of f_h and t_0. When either reduces to zero, this multi-path effect is completely removed[4]. This means an accurate timing synchronization is desired. As we will show later, we achieve this by adding Layer 0.5 synchronization sequences.

Finally, we note that for some PHY that are designed to handle multi-path channel, such self-generated multi-path effect may have less adverse impacts. For example, for OFDM, the constant multi-path coefficients for $x_1(t-t_0)$ and $x_2(t-t_0)$ may just result a constant phase-shift on a sub-carrier, which can be easily handled by equalization without losing performance. As we will show later with our experiments, spectrum spreading PHY (*e.g.* 802.11b 1/2Mbps and Zigbee) can also handle multi-path channel well. But for other PHY, like CCK in 802.11b 5.5/11Mbps, the demodulation performance will be significantly reduced with multi-path fading [18]. Thus, for such PHY, we would suggest to disable non-contiguous spectrum bond by specifying *SVL_VS_NO_SPLIT* policy when registering to SVL.

4.2 Timing Virtualization

When SVL maps a virtual baseband to a physical band with a narrower width, it will take more time to transfer baseband signals than a PHY would imagine. For example, if an 802.11a PHY with $20MHz$ baseband is mapped to a $10MHz$ physical band, it may take SVL actually $8\mu s$ to send a symbol instead of $4\mu s$ as expected by the virtual PHY. These changes in timing may impact the correctness of the wireless protocols (*e.g.* MAC *etc.*) if they rely on absolute time information. For example, Network Allocation Vector (NAV) and ACK timeout would be expired pre-maturely if the transmitting time of PHY signal is extended.

In this paper, we further use *timing virtualization* to address the issue. The core idea of *timing virtualization* is to present a *virtual clock* to each virtual PHY and wireless protocols sat above. The ticking rate of this *virtual clock* is adaptive according to the actual allocated physical spectrum bands. Specifically, let b_s be the aggregated bandwidth of allocated physical bands, and b_v be the width of virtual baseband. We adjust the ticking rate by a factor of b_s/b_v.

Timing virtualization may require MAC and other high layer protocols to change their implementation to get timing information only from the virtual clock. However, we believe such changes should be minimal as most implementation may have common clock APIs that refer to single clock source. Therefore, we may only need to re-implement these limited clock API functions.

In current SVL, we define the following API for timing virtualization. The returned clock is at a precision of $25ns$.

```
int svl_get_clock ( vs_handle vs_id ,
                    longlong& clock );
```

Note that for a complex wireless protocol that can not be easily changed to support *timing virtualization*, it can still work with SVL by specifying *SVL_VS_NO_SHRINK* policy. Then, SVL will always try to allocate enough physical bands to ensure the signal timing does not change.

4.3 RF Front-end Multiplexing

SVL supports multiple PHY multiplexing on the same wide-band RF front-end as long as the width of the RF front-end can cover the span of physical bands allocated to these PHY. SVL uses *mixer* and *splitter* to support multiple PHY multiplexing.

The mixer sits in the transmitting chain, which collects the physical band signals of each PHY (after reshaping), scales the signal amplitude according to each PHY's power mask and then adds (mix) them up before sending to DAC of the RF front-end. On the other hand, the splitter contains a set of band-pass filters, each of which matches a physical band that has been allocated to a PHY. For PHY that has non-contiguous physical bands, filters to all bands are combined to form a single band-selective filter. The splitter applies a matched band-selective filter for each PHY and the filtered signal samples are fed to the corresponding reshaper to the PHY.

It should be noted that if a device has only one half-duplex RF front-end, multiplexing multiple PHY may require care-

[4]When $f_h = 0$, it means a contiguous physical channel.

ful scheduling, since such a RF front-end can only be either receiving or transmitting. Thus, different PHY should be scheduled to transmit and receive simultaneously. It is easy for SVL to deploy such a scheduler to synchronize the behaviors of multiple PHY. SVL uses buffers to temporarily hold baseband samples when the RF front-end is receiving, and defers the transmissions until the receiving is done (*i.e.* detecting no signal power). And SVL can also hide this buffering latency from PHY by subtracting it from the *virtual time*. However, when working in a network of several nodes with different PHY, there may be a potential *"blind node"* problem. When one PHY is transmitting on a spectrum band, it will put the RF front-end into transmitting mode. This will prevent other PHY mapped to the same RF front-end from receiving any incoming signals, even though they are on an orthogonal band. Such a problem is in general difficult and needs further studies.

On the other hand, RF front-end multiplexing would work best if RF transmitting chain and receiving chain can work simultaneously in a full-duplex mode. It can be achieved with a full-duplex RF front-end, or attaching two half-duplex RF front-ends to SVL. Full-duplex RF front-ends are quite common today for FDD cellular systems. We believe for future DSA systems, it should enable both TX and RX chains of a wide-band agile RF front-end simultaneously. It is because wireless PHY may use only portion of RF bandwidth for sending and therefore it could receive at same time on a separated band.

It should be noted that the full-duplex mode discussed here is different from that in [9]. In [9], Choi *et. al.*, tried to enable both sending and receiving in the *same* spectrum band, which is far more challenging. In our case, the sending and receiving bands are orthogonal and analog notch (band-stop) filters can be applied to filter out self-transmitted signals to prevent the receiving chain from being saturated.

5. IMPLEMENTATION

We have implemented a prototype SVL on Sora, a high-performance and fully programmable software radio platform based on general-purpose x86 PC architectures [15].

5.1 Implementation architecture

Figure 11 illustrates the software architecture. For programming convenience, we place the entire networking stack in user-space. Each instance (represented only by PHY here) is implemented as a user-level thread. It uses direct memory mapping to gain zero-copy access to the kernel DMA buffers that store the incoming digital samples (*Rx Buf*), and is allocated with a local buffer for outgoing samples (*Tx Buf*). Each PHY instance is linked with a SVL library, which implements all the necessary signal processing functions like reshaper and splitter. The mixer function, which collects all the generated samples from each PHY's Tx Buf and mixes them into the *Tx DMA buffer*, is implemented as a separated user-level thread.

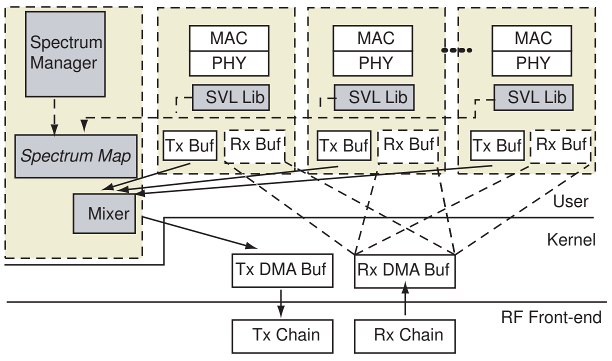


Figure 11: Implementation architecture of SVL on Sora.

The spectrum manager is a separate program. It manages a database called *spectrum map* for all PHY instances, and dynamically configures the reshaper/splitter parameters at their SVL library to yield the desired virtualization effects. The actual spectrum management function is however rudimentary in the current version and serves only as a proof-of-concept: it reads a sequence of frequency allocation changes from a configuration file or from the command line. Sophisticated spectrum management algorithms can be implemented later as DSA research progresses.

5.2 Layer 0.5 Synchronization

Since we use FFT/iFFT to decompose a signal to support non-contiguous spectrum bonding, we need to perform N-point FFT on the aligned time-domain samples at the receiver side to correctly recompose the signal. To achieve this, we need to add some special sequence (sync-sequence) at Layer 0.5 to synchronize the symbol boundaries.

Sync-sequence is generated as follows. Using a long-enough master pseudo-random noise (PN) sequence, we take the first M samples (the same M as in M-point FFT used by the PHY for decomposition) and map them to the corresponding sub-carriers as if they were the frequency components of a PHY symbol. Then we perform N-point iFFT to generate the time-domain samples of the sync-sequence. Note that when sender and receiver sides agree on the same physical band allocation, the sync-sequence between them is also then determined.

At the sender side, this sync-sequence is inserted as header of a block of PHY symbols (*e.g.* a whole PHY frame). The overhead is one PHY symbol per frame, or less than 1% for a 1500-byte frame in 802.11g with 36Mbps data rate. At the receiver side, the recomposition module will continuously look out for this sync-sequence. The symbol boundary is then marked whenever such a sync-sequence is detected.

We note the synchronization is need only when the PHY uses non-contiguous bonding. If the baseband is mapped to a contiguous physical band, there is no need for the decomposition and recomposition operations, and hence no need for keeping FFT boundary information.

6. EVALUATION

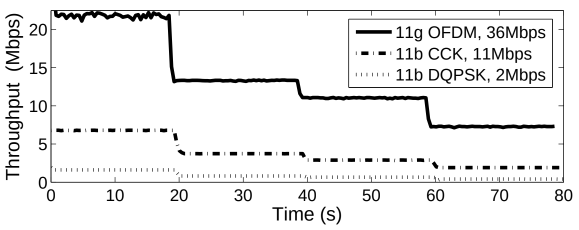
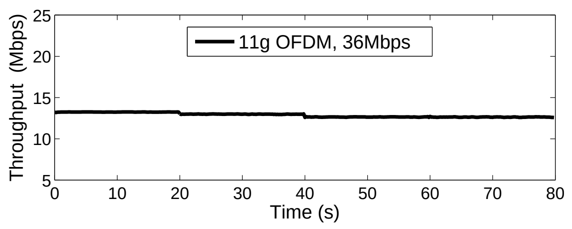
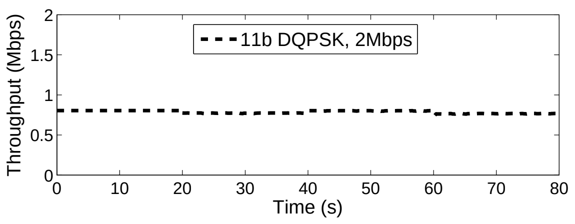


Figure 12: Measured throughput of various PHY with variable contiguous physical spectrum allocation.



(a)



(b)

Figure 13: Non-contiguous spectrum bonding. The sum of bandwidth of two physical band is 10MHz. (a) Measured throughput of 802.11g (OFDM); (b) Measured throughput of 802.11b (DQPSK).

In this section, we evaluate the performance of SVL using our prototype implementation on four Sora stations. Each Sora station is a PC with Intel core i7-980X CPU (3.33GHz, six-core) and 3GB memory. We run several popular wireless PHYs, including 802.11b, 802.11g, and ZigBee. For 802.11b and 802.11g PHY, we use the source code from Sora SDK. We wrote ZigBee PHY ourselves because it is relatively easy. We use two kinds of 2.4GHz RF front-ends in our experiment: one with 40MHz bandwidth and half-duplex (either receive or transmit at a time), and the other with 20MHz bandwidth but full-duplex, *i.e.*, separated TX and RX chains capable of simultaneous transmitting and receiving on different spectrum bands. Although we conduct our experiments in 2.4GHz ISM band, we fully expect similar results in other spectrum bands (*e.g.* TV White-Space).

6.1 Flexible Spectrum Bonding

First, we evaluate SVL's ability to support flexible spectrum bonding. We measure the network throughput with different physical spectrum allocations, and validate if SVL can indeed reshape baseband signals to match the spectrum specification. We conduct the experiment under three different types of PHY, namely 802.11g (OFDM, 36Mbps), 802.11b (CCK, 11Mbps), and 802.11b (DQPSK, 2Mbps), all operating on 20MHz virtual baseband. At every 20 seconds, we change the SVL spectrum map to reduce the allocated width of the physical band from 20MHz to 10MHz, then to 8MHz, and finally to 5MHz. All PHY frames are 1000-byte long, with the exception of 600-byte for 802.11b (DQPSK) case due to a sample length limitation on Sora platform.

Figure 12 shows the measured throughput under each of these three PHYs. We can see SVL can reshape virtual baseband signals to varied physical band very well. The link throughput changes proportionally in each case. For example, when the physical bandwidth reduced from 20MHz to 10MHz, the throughput under each PHY also drops to half. This result validates our expectation that varying spectrum allocation width can be properly supported in SVL.

Next, we measure for the non-contiguous spectrum bonding case. We allocate two disjoint bands with a total bandwidth of 10MHz. At every 20 seconds, we enlarge the gap between these two bands from 0Hz (contiguous case) to 5MHz, to 8MHz, and to 10MHz. Figure 13 shows the measured throughput under two types of PHY, 802.11g (OFDM) and 802.11b (DQPSK). The results imply that neither the existence of a spectrum hole nor its size seems to affect the throughput, validating that the non-contiguous spectrum bonding can also be properly supported.

6.2 DSA Networking

We now further evaluate SVL in a simple DSA network. There are three different types of wireless PHY used in this network, namely 802.11g (OFDM), 802.11b (CCK), and ZigBee, all sharing the same spectrum band. We perform the experiments based on the following scenario (Figure 14(d)). At the beginning, there is only one 802.11g flow in the network and it occupies a 20MHz band as normal. At 20th second, a ZigBee flow starts, which usually takes 5MHz in the same band as in 802.11g. To avoid interference, the DSA manager intervenes and moves the allocation for the 802.11g flow away from the spectrum used by ZigBee. The new allocation now occupies two disjoint spectrum bands, with a hole in the middle (occupied by ZigBee). The sum of these two disjoint bands is still 20MHz. At 40th second, an 802.11b flow starts. The DSA manager intervenes again and assigns half of the 802.11g flow's allocation to the new 802.11b flow. Finally at 60th second, the ZigBee flow ends and at the same time the 802.11g flow's traffic requirement reduces significantly. The DSA manager then re-allocates spectrum accordingly and assigns a 20MHz band to the 802.11b flow. Figure 14(d) shows this spectrum allocation change over time.

We use an out-of-band control mechanism to coordinate all nodes in this DSA network to follow the above schedule. The frame size is 1000-byte for both 802.11b and 802.11g flows, and 30-byte for the ZigBee flow. Figure 14(a)–(c) show the measured throughput in each flow. The results largely match our prediction. The 802.11g flow receives a throughput of 22.7Mbps when it uses 20MHz spectrum

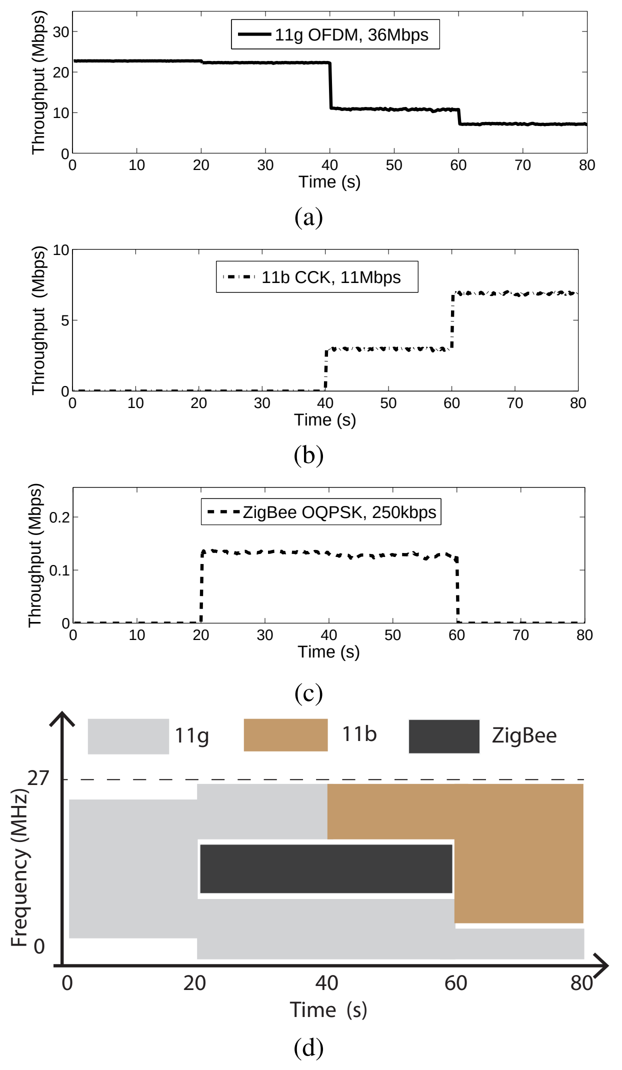


Figure 14: Measured throughput of different flows in an example DSA network. (a) 11g(OFDM, 36Mbps); (b) 11b(CCK, 11Mbps); (c) ZigBee(250kbps); (d) The spectrum allocation.

in contiguous or non-contiguous bands. Then it drops to 10.8Mbps after yielding 10MHz band to the 802.11b flow at time 40. It further reduces to 7Mbps after yielding even more bandwidth at time 60. Similarly, the 802.11b flow doubles its throughput from 3Mbps to 6.9Mbps when its bandwidth increases from 10MHz to 20MHz at time 60. Since ZigBee does not change its spectrum allocation, the throughput remains the same at 130Kbps.

This experiment also demonstrates the benefit of DSA to improve spectrum usage efficiency in wireless networks. In a traditional wireless network, a ZigBee transmission would prevent 802.11g from using the same channel, even though ZigBee only occupies a small portion of the spectrum. Here in a DSA network, 802.11g can still make efficient use of the remaining bandwidth and the wireless spectrum is not wasted.

6.3 RF Front-end Multiplexing

SVL supports RF front-end multiplexing, i.e., to multiplex multiple separate wireless networks, hence multiple PHY instances, on one full-duplex RF front-end. That is, the node can simultaneously communicate on multiple separate wireless networks (such as on different frequency bands), each via a different PHY. In this experiment, we run two instances of 802.11g PHY (OFDM, 18Mbps) on one Sora node. Each PHY is allocated with a 5MHz physical band and is expected to have a throughput around 4Mbps.

In the experiment, we turn on both PHYs and put one in receive mode and the other in transmit mode. We are particularly interested in this case because it is easier otherwise (both in the same receiving or transmitting mode). We vary the "distance" by which the two physical bands are seperated from each other, and measure the throughput of both PHYs. The result is listed in Table 2. It clearly shows that, as long as the two physical bands are orthogonal (e.g., Distance ≥ 0), each PHY can achieve the same throughput independent of the other PHY. This indeed validate that a full-duplex RF front-end can support two simultaneous PHYs, as long as the two signals occur on different spectrum bands. [5]

Table 2: Throughput measured of two PHY multiplexing on a single full-duplex RF front-end.

	Distance			
	0MHz	**2MHz**	**4MHz**	**8MHz**
PHY1	3.97Mbps	3.98Mbps	3.98Mbps	3.99Mbps
PHY2	4.06Mbps	4.06Mbps	4.06Mbps	4.06Mbps

6.4 Reshaping Operation Precision

SVL employs sophisticated digital signal processing (DSP) to reshape the baseband signals, which will introduce additional processing errors. To quantify these errors, we first reshape baseband signals to physical band signals, and then immediately reshape them back into baseband. Then we compute the mean square error (MSE) between the original signals and the "reshape-back" signals. Table 3 summarizes MSE values normalized to the signal energy when we reshape a 20MHz baseband to two 5MHz non-contiguous physical bands and back. Rows marked "11g" show the results when 802.11g PHY (OFDM, 36Mbps) is used and rows marked "11b" are the results when 802.11b PHY (DQPSK, 2Mbps) is used. The first column labels the bit-width of our fix-point data in DSP algorithms. We also vary the choice of FFT points (N). As we have discussed in Section 4, a larger N means a wider separation of these two non-contiguous

[5] In this experiment, we do not see much out-band power leakage in our full-duplex RF front-end because its output power is relatively small compared to commodity radios. This may become a problem with high power RF hardware, but it can be mitigated with an advanced filter or with proper guard-band (*i.e.*, enlarging the distance between two physical bands) [17].

physical bands.

We can see that the processing error increases with FFT points. With 128 point FFT and using 16-bit fix-point DSP, the average error is about -17dB below the signal energy, which is acceptable for many PHYs like 802.11b and 802.11g at a rate of 36Mbps or below. If we want to support higher data rate, or if we want to bond to widely separated bands with large N, we will need to increase our computation precision by using 32-bit fix-point DSP. At this point, we believe 32-bit will be good enough because the error introduced is below -40dB (Table 3), which is much lower than noise from a common wireless channel.

Table 3: Error introduced by SVL DSP operations (dB).

Int-Width	PHY	N-Point			
		128	256	512	1024
16b	11g	-17	-15.2	-13.3	-8.3
	11b	-17.8	-15.9	-13.6	-10.7
32b	11g	-46.4	-43.9	-41	-38.1
	11b	-48	-45.4	-42.6	-39.8

6.5 SVL Overhead with Software Implementation

Since we implemented SVL in software, we are interested in knowing its overhead. We measure the CPU time spent on SVL and normalize it as a fraction of one CPU core (Intel Core i7-980X, 3.33GHz). The result is listed in Table 4. We consider three settings, "CB 10M" for 10Mhz contiguous band, "CB 5M" for 5Mhz contiguous band, and "NCB 10M" for non-contiguous bands with 10Mhz total bandwidth. The physical bandwidth is 40Mhz in all settings. As we can see from the table, SVL requires an additional 0.45–0.62 CPU core for reshaping contiguous bands. It requires even more for non-contiguous spectrum bonding (up to 0.89 CPU core), because it needs additional FFT/iFFT in the non-contiguous case. This suggests that we should budget an additional CPU core for SVL if we are to adopt a pure software implementation, Or, given the layering separation between spectrum management and spectrum access, we can opt for hardware acceleration because it contains only regular DSP operations and have a well-defined interface.

The abnormal data point of 802.11b (DQPSK, 2Mbps) with non-contiguous physical bands (the last item in the first row in Table 4) can be explained as follows. Although it requires additional FFT/iFFT to support non-contiguous spectrum bonding, in this particular setting, it does not need to adjust its bandwidth and sampling rate. It thus needs less computation compared with the contiguous spectrum bonding cases.

7. SVL APPLICATIONS

White-Space networking is an immediate application of SVL. White-Space networking allows opportunistic use of

Table 4: Computation overhead (fraction of a CPU core).

PHY	CB 10M	CB 5M	NCB 10M
11b TX (DQPSK)	0.45	0.47	0.39
11b RX (DQPSK)	0.53	0.62	0.89
11b TX (CCK)	0.48	0.47	N/A
11b RX (CCK)	0.53	0.62	N/A
11g TX (OFDM)	0.55	0.54	0.71
11g RX (OFDM)	0.47	0.59	0.61

unused TV channels for data communications. In United States, the White-Space spectrum is 512-698Mhz and each TV channel occupies 6Mhz. The traditional approach towards utilizing this spectrum is to develop a new PHY/-MAC. Indeed, IEEE is working a new standard called IEEE 802.11af, which is expected to be a long process.

Alternatively, we have designed an immediately deployable White-Space network based on SVL and the very proven 802.11g PHY. We use SVL to virtualize a contiguous 20Mhz baseband over the 512-698Mhz band. As long as there is a suitable RF front-end, such as USRP WBX, we can establish a WiFi network on any number of spare TV channels. While SVL maps one or more spare TV channels (contiguous or non-contiguous) into a 20Mhz virtual baseband, the 802.11g PHY will establish links as if it were in 2.4Ghz band. This solution differs from the WhiteFi project [4] in which only 5Mhz over one TV channel is used, as we can utilize any number of channels, even non-contiguously.

Another application of SVL is to facilitate the integration and co-existing of hetergoneous wireless systems. As mentioned earlier, there is an increasing number of different wireless standards, each of which is designed to fit a specific application need. It is a common case that multiple devices with heterogenous wireless standards are simultaneously used in a vicinity. For example, at home, a user may simultaneously talk over a cordless phone while browse pages using WiFi; while home appliance may communicate or sense with ZigBee and RFID. Currently, it needs to setup multiple Access Points (AP) or Network Controllers (NC) for each wireless standard, adding considerable management overhead and hardware cost. Further, it is also a complex issue to avoid mutual interference from one another.

With SVL and software radio, we can deploy multiple heterogenous wireless standards on the same wide-band RF front-end hardware, such as *Multiple Purpose Access Points (MPAP)* [11]. The benefit of MPAP is three-folds: First, it consolidates multiple wireless devices into a single hardware platform and thus reduces the maintenance cost. Second, based on software radio, it is flexible and extensible to support future wireless standards. Finally, SVL enables *dynamic spectrum access* for all wireless PHY deployed. Thus, they can use spectrum much efficiently to achieve better coexistence. We believe DSA is important technique to improve the spectrum usage efficiency when the wireless band is crowded by many heterogenous wireless standards.

8. RELATED WORK

Dynamic spectrum access (DSA) has received a lot of attentions in the research community in recent years [3]. Several DSA systems have been implemented [4, 13, 16, 17]. For example, White-Fi [4] uses UHF White Space for data communication. It takes a standard Atheros Wi-Fi card and down-converts the Wi-Fi signals to UHF band and vice versa. It further supports multiple contiguous TV channels with variable channel width of 5, 10 and 20MHz [8]. SWIFT [13] enables wide-band wireless devices to co-exist with narrow-band users in unlicensed band. It customizes the OFDM PHY to avoid the sub-areas in the spectrum band that are used by narrow-band users. Jello [16], another OFDM PHY based DSA system, is similar to SWIFT in the sense that they both support non-contiguous spectrum bands. They differ from each other in the way they senses the channel and manages the spectrum allocation. Unlike these DSA work that proposed a new cognitive PHY, our goal is to devise a network architecture so that DSA can be provided as a general function to all wireless PHY. Nevertheless, these work can also be complementary to SVL because SVL requires a separate spectrum management function and it can adopt the spectrum sensing and management approaches from these work.

The term *Virtual Radio* was proposed by Bose, *et. al.*, about a decade ago [5], to refer to a software radio running on general purpose PC architecture. With many recent efforts (like [15]), software radio is increasingly recognized as a powerful enabler for cognitive and DSA systems. Indeed, SVL is implemented on a software radio platform.

9. CONCLUSION AND FUTURE WORK

This paper presents *spectrum virtualization layer (SVL)*, a *Layer 0.5* approach to support *dynamic spectrum access* for general wireless networks.

SVL decouples the traditionally tightly connected wireless PHY and RF front-end and adds a layer of indirection. SVL provides a *virtual baseband* abstraction to PHY, which is fixed, contiguous, with a desirable width defined by the PHY. At the sender side, PHY generates digital waveform as if it connected to a RF front-end directly. Then, SVL performs real-time DSP operations to reshape the digital waveform to match the dynamically allocated physical bands. At the receiver side, SVL performs inverse reshaping operation on the physical band signals to recover the original digital waveform for PHY. Consequently, SVL provides general DSA support for various wireless PHY without modification.

Beside, SVL further provides the flexibility to map wireless PHY to RF front-ends. It can map different PHY to different RF front-ends, and it is also able to multiplex multiple PHY onto single RF front-end. Thereby, it enables multiple PHY to share a common powerful wide-band RF hardware, and thereby simplifies the integration of multi-radio in one mobile device and reduces the cost.

We have implemented a prototype of SVL based on a software radio platform. Our experiments show that our SVL design are flexible and effective to support DSA for general wireless designs with affordable overheads.

There are several future directions we are going in: First, SVL can facilitate the takeoff of TV Whitespace networking by adopting existing well-understood wireless designs into whitespace with little efforts. Second, SVL is flexible to deploy various dynamical spectrum management algorithms. So we can evaluate and compare these algorithms with a single common platform. Finally, we will continue optimizing our reshaper design in software and explore the possibility to accelerate its processing using hardware.

10. REFERENCES

[1] 3GPP TS 25.213:. 3rd Generation Partnership Project (3GPP); Spreading and modulation (FDD). 1999.

[2] 3GPP TS 36.201-820:. Evolved universal terrestrial radio access (E-UTRA); long term evolution (LTE) physical layer; general description.

[3] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty. Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *COMPUTER NETWORKS JOURNAL (Elsevier)*, 2006.

[4] P. Bahl, R. Chandra, T. Moscibroda, R. Murty, and M. Welsh. White space networking with wi-fi like connectivity. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, pages 27–38, New York, NY, USA, 2009. ACM.

[5] V. Bose, M. Ismert, M. Welborn, and J. Guttag. Virtual Radios. *IEEE Journal on Selected Areas in Communications*, 17(4), April 1999.

[6] D. Cabric, A. Tkachenko, and R. W. Bordersen. Experimental study of spectrum sensing based on energy detection and network cooperation. In *ACM 1st International Workshop on Technology and Policyy for Accessing Spectrum*, 2006.

[7] L. Cao, L. Yang, and H. Zheng. The impact of frequency-agility on dynamic spectrum sharing. In *DySPAN '10*, 2010.

[8] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl. A case for adapting channel width in wireless networks. *SIGCOMM Comput. Commun. Rev.*, 38(4):135–146, 2008.

[9] J. I. Choi, K. Srinivasan, M. Jain, P. Levis, and S. Katti. Achieving single channel, full duplex wireless communication. In *ACM Mobicom*, 2010.

[10] FCC press release. FCC adopts rules for unlicensed use of television white spaces. 2008.

[11] Y. He, J. Fang, J. Zhang, H. Shen, K. Tan, and Y. Zhang. MPAP: Virtualization Architecture for Heterogenous Wireless APs. *CCR Online*, 2010.

[12] A. V. Oppenheim, R. Schafer, and J. Buck. *Discrete-Time Signal Processing (2nd Ed)*. Prentice-Hall, 1999.

[13] H. Rahul, N. Kushman, D. Katabi, C. Sodini, and F. Edalat. Learning to share: narrowband-friendly wideband networks. *SIGCOMM Comput. Commun. Rev.*, 38(4):147–158, 2008.

[14] K. Tan, J. Fang, Y. Zhang, S. Chen, L. Shi, and J. Z. Y. Zhang. Fine grained channel access in wireless lan. In *ACM SIGCOMM 2010*, 2010.

[15] K. Tan, J. Zhang, J. Fang, H. Liu, Y. Ye, S. Wang, Y. Zhang, H. Wu, W. Wang, and G. M. Voelker. Sora: High performance software radio using general purpose multi-core processors. In *NSDI 2009*.

[16] L. Yang, W. Hou, L. Cao, B. Y. Zhao, and H. Zheng. Supporting demanding wireless applications with frequency-agile radios. In *In Proc. of NSDI (2010)*, 2010.

[17] L. Yang, B. Y. Zhao, and H. Zheng. The spaces between us: setting and maintaining boundaries in wireless spectrum access. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, MobiCom '10, pages 37–48, New York, NY, USA, 2010. ACM.

[18] P. Yang and D. Noneaker. Performance analysis of cck modulation in a multi-path channel. In *Research Report at UC Berkeley*, 2002.