An Open RF-Digital Interface for Software-Defined Radios

This article presents an open, complete RF-digital interface appropriate for software-defined radios (SDRs). The interface includes data and metadata (control and context) packets. The control and context packets describe the entire RF front end. The proposed description has a hierarchical structure and is a hardware abstraction. The interface supports advanced architectures such as SDR clouds. The metadata packets can be represented using formal, computer-processable semantics.

Todor Cooklev
Indiana University—Purdue University Fort Wayne

Akinori Nishihara
Tokyo Institute of Technology

Wireless signals are centered at a certain RF. The signal-processing operations—synchronization, modulation, and so on—are implemented on baseband signals centered at zero frequency. Therefore, receivers must down-convert from RF to baseband. Transmitters perform the opposite process of up-conversion. The RF front end, located between the antenna and the digital subsystem, performs the tasks of down-conversion and up-conversion and also does filtering for frequency band selection and amplification.

In this article, we advance an RF-digital interface that is open, which allows the RF front end and the digital hardware to be seamlessly replaced independently of each other. This property is not achieved by previous attempts at defining this interface.

The developed interface is a stream of data and metadata (context and control) packets. The proposed interface completely describes the RF front end and can serve as a hardware abstraction language for it. The metadata packets can be described using a simple ontology, such as RDF, and exchanged among different cognitive radio platforms.

RF front-end details

As Figure 1 shows, the RF front end is not entirely analog; it consists of an analog front end and a digital front end (DFE).[1] The DFE does interpolation or decimation to increase or decrease the sampling frequency. Furthermore, the down-conversion and up-conversion can be implemented partially or even entirely with digital signal processing (DSP). (See the "Glossary" sidebar for a list of terms.)

In a software-defined radio (SDR), the radio access technologies (RATs) at the baseband level are implemented entirely in software. A RAT can be called a *radio protocol,* although more accurately it is an aggregation of protocols. For radio protocols implemented entirely in software, the digital hardware platform must be programmable. Such platforms can be built with general-purpose processors, digital signal processors, or field-programmable gate arrays (FPGAs).

The International Telecommunication Union (ITU) defines SDR as a radio "that allows the RF operating parameters including, but not limited to, frequency range, modulation type, or output power to be set

0272-1732/13/$31.00 © 2013 IEEE Published by the IEEE Computer Society

Glossary

Baseband: digitally modulated signal that is low-pass in nature (that is, not up-converted onto an RF carrier)

Interpolation: increasing the sampling frequency

Decimation: decreasing the sampling frequency

IF: intermediate frequency

Down-conversion: the process of moving a signal's spectrum content from RF to zero frequency (or to IF)

Up-conversion: the process of moving a signal's spectrum content from zero to RF (or to IF)

Waveform software: software that implements a radio-access technology

Ontology: a general mechanism to describe objects in a certain domain and the relationships between them

Metadata: data that helps interpret signals (data about data)

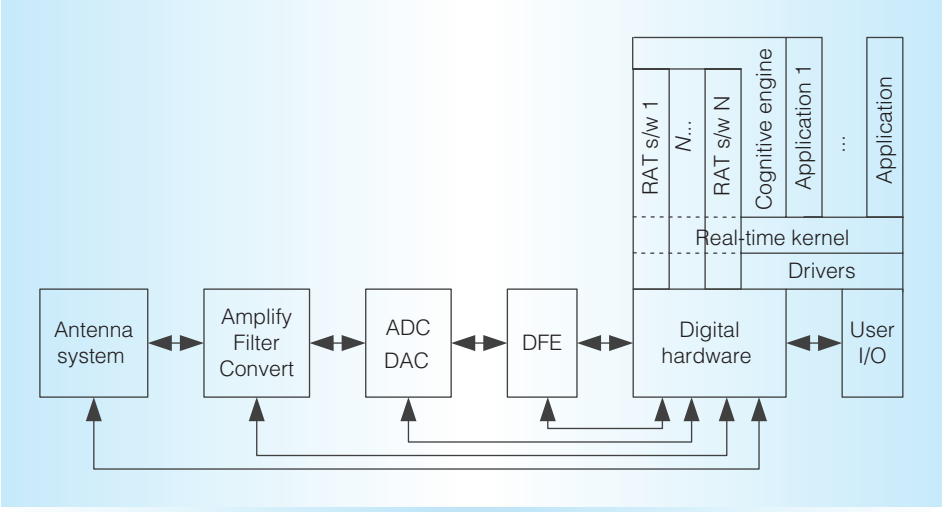

Figure 1. Block diagram of a software-defined radio (SDR) system. The architecture of an SDR includes hardware, system software, and service software layers.

or altered by software, excluding changes to operating parameters which occur during the normal preinstalled and predetermined operation of a radio according to a system specification or standard."[2] Therefore, SDR can be defined simply as a radio that can implement radio protocols to be defined in the future. Therefore, SDR architecture is the science and art of interconnecting hardware and software components to create such "future-proof" radios. To become future-proof, radios should have upgradable software and hardware. Architectures that allow the adding, upgrading, and swapping of hardware and software components can be called *open.* If one makes a parallel comparison with computer architecture, an SDR's architecture includes hardware, system software, and service software layers (see Figure 1).

The radio in Figure 1 can operate at any one of N different RATs, such as Wi-Fi, GSM, and LTE-Advanced. The center frequency and power at the RF level can be considered completely independent of the radio protocol. A special element is necessary to determine which radio protocol is active at any one time, and at what parameters (such as center frequency and power) this protocol will operate. This element in Figure 1 is the cognitive engine (CE). (The architecture in Figure 1 is similar to that in previous work by the SDR Forum,[3] where the term "switcher" is used instead of a CE.) Figure 1 can also describe radios that aren't software-defined, in which the radio protocols are implemented with dedicated hardware that isn't programmable. In such hardware radios, usually every radio protocol is connected using a

closed RF-digital interface to a separate RF front end.

In both SDR and non-SDR, there are other software applications. The difference between these software applications and radio protocols implemented in software (or "radio software," for short) is that the radio software has much greater execution-speed requirements. As a result, the radio software can't be made completely independent of the digital hardware, which is indicated by the dotted lines in Figure 1. This is the main barrier to implementing radio protocols entirely in software. Another, more subtle barrier to achieving SDR is the lack of an open RF-digital interface. If the RF-digital interface is closed, the radio protocols become hardware dependent.

In pursuing an open architecture, several horizontal and vertical interfaces in Figure 1 can be defined. The Software Communications Architecture (SCA) defines a set of interfaces that isolate the radio applications from the hardware.[4] Another related work specifies a programming interface of C++, VHDL (VHSIC Hardware Description Language), and IDL (Interface Definition Language) that is a set of APIs between the waveform application and the rest of the radio.[5] The goal is to make the waveform software portable onto different platforms.

While these interfaces are also important, the RF-digital interface is critical in achieving hardware modularity. Early attempts at defining the RF-digital interface for SDR considered a hardware bus-type interface.[3] One obvious limitation of this solution is that a hardware bus can operate only at specific clock speeds and data rates. Furthermore, the work completely ignored the metadata, including parameters such as frequency and power.[3] These parameters were defined for the first time by VITA 49.[6] However, the VITA 49 standard is incomplete because it doesn't specify the transmitter side or how to control the receiver. In this article, we describe a complete packet-based interface that lets the RF subsystem and the digital subsystem be replaced independently of each other. This interface completely describes the RF front end—that is, it can be used as a hardware abstraction language for the RF front end.

The RF-digital interface

The RF-digital interface in Figure 1 includes both high-speed data and low-speed metadata. Unlike conventional radios, in SDR the metadata can't be assumed to be constant and known. Figure 2 shows a way to implement the RF-digital interface where the metadata is defined explicitly. In general, the main blocks of a transmitter include a digital-to-analog converter (DAC), followed by an up-converter, power amplifier (PA), and transmit RF filter. An up-converter or down-converter generally consists of a local oscillator, mixer, and a filter. The local oscillator's frequency is software-defined. After up-conversion, the RF signal is amplified using the PA, for which gain and output power are the main programmable parameters. To adjust the transmit power level, there might be a programmable attenuator (not shown). The receiver consists of a channel-select RF filter, amplifier, down-converter, and analog-to-digital converter (ADC). The important parameters of programmable RF filters are RF, bandwidth, out-of-band attenuation, and stop-band edge. A programmable impedance-matching circuit (not shown) can also be used. A duplexer separates the transmit and receive paths in time or frequency. The antenna and the duplexing technique must also be software-defined in general, although this isn't easy to achieve in practice. In any practical SDR device, not all parameters will be software-defined. Which parameters are programmable and the specific way they are programmed is hardware-dependent and should be hidden from the RF-digital interface.

To achieve these properties, we propose a packet connection between the RF and digital subsystems (Figure 2). Over this packet connection, there are six packet types that implement the interface: data, context, control, extension data, extension context, and extension control packets. The data, context, and control packet types are mandatory. Each of the extension packet types is optional. The packet header conveys the packet type. If a decoder does not support a particular extension packet type, it ignores its content and could provide feedback indicating a potential problem.

In this article, we focus on the case in which the RF front end generates context packets and the digital hardware side generates control

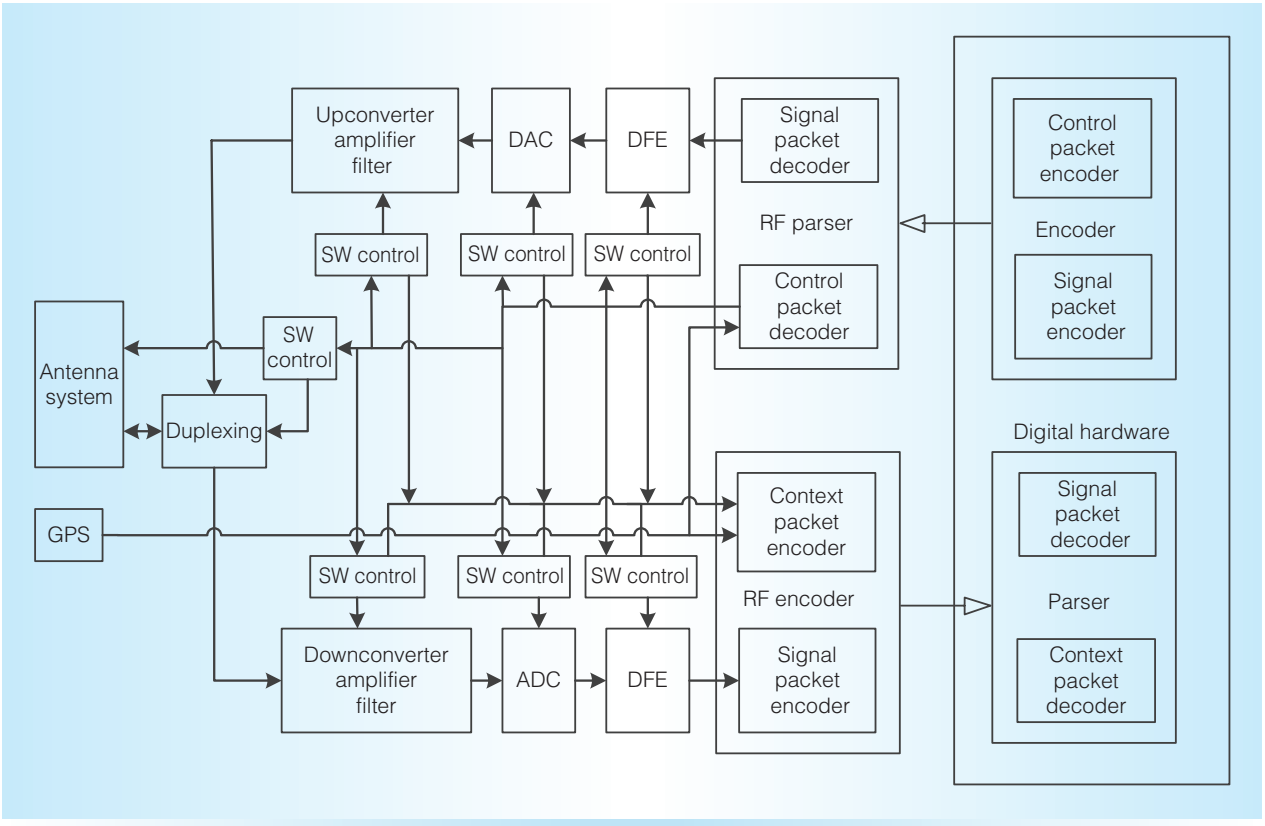

Figure 2. The packet-based RF-digital interface. The metadata is defined explicitly.

packets, but more generally, all metadata packets are bidirectional. Context packets include parameters pertaining to both the transmitter and receiver such as RF, IF, bandwidths, ADC and DAC sample rates, ADC, and DAC number of bits, gain, voltage full-scale range, filter order, out-of-band attenuation, stop-band edge, transmit output power, reference point, timestamp, timestamp delay, and location. Collectively, these parameters describe the RF front end completely. Every time one or more of these parameters changes, the RF encoder sends a context packet containing the current value of the parameters that have changed.

The control packets contain metadata specifying the reference point, timestamp, and output power pertaining to the transmitter, and the DAC and ADC sample rate, DAC and ADC number of bits, reference level, voltage full-scale range, RF, IF, local oscillator (LO) power, bandwidth, gain, filter order, out-of-band attenuation, and stop-band edge pertaining to both the transmitter and receiver. The control packets include all RF front-end parameters that are software-defined. The control packet decoder is tightly coupled to the analog front end and translates the metadata to each programmable device's hardware settings.

In addition to the data, context, and control packets, our proposed solution includes their extension counterparts. The interface is extensible, and arbitrary additional parameters can be defined in extension control packets. For example, extension data packets could carry spectrum information. The extension context packets can be used to convey bit error rates, power supply voltages, and receiver noise figures. Note that there might be power-management circuitry that can turn off parts of the radio when not in use. Although Figure 2 doesn't show power management, if this feature is desired, power-management information can be specified in extension control packets.

At the RF-digital interface, encoders and parsers exchange multiplexed streams of packets (Figure 2). The interface to the RF

transmitter has a packet encoder on the digital hardware side and a packet parser on the RF side. The output of the packet encoder is a stream of multiplexed signal and control packets (and optionally their extensions). The parser on the RF side receives this stream and separates the control packets from the data packets. The payload of the transmit data packets is fed to the DAC. The parameters specified in the control packets are translated into a hardware-specific format for every software-defined element.

Every circuit element of the RF front end in Figure 2 can be controlled individually using a control packet. However, every parameter has minimum and maximum values. For example, the RF front end should not be told to tune to a particular frequency if it can't operate at that frequency. Within one radio, the tuning range (if any) is always known, but like all metadata in previous radio designs, it has never been explicitly defined. Now, the control packet encoder on the digital side must be made aware of every component's tuning range to produce proper control packets. This awareness can be provided by context packets or extension context packets. If a parameter is fixed, this can be communicated by the tuning range information. Typically, the tuning range must be updated when the RF front end is replaced. Because this happens less often than other contexts need to be changed, it's appropriate to specify tuning range in extension context packets transmitted by the context packet encoder in Figure 2.

The RF side also has a packet encoder that's connected to a packet parser on the digital hardware side. The elements in the receiver chain are connected in a hardware-dependent way to the context packet encoder, which in turn produces proper context packets with information such as RF, IF, bandwidth, and sample rate. The context packet encoder can also keep track of GPS data and includes location information in the context packets. The RF packet encoder's output is a stream of multiplexed signal and context packets (and optionally, their extensions). From this stream, the digital-side parser separates the packet types and processes them appropriately.

It becomes clear that the metadata (control and context) packets are multiplexed with the data packets and can be considered as overhead. However, the metadata packets need not be sent as often as the data packets. Control and context packets need to be sent only when the relevant metadata changes—that is, the metadata is understood to be persistent between updates. This makes the overhead due to the control and context packets proportional to how often parameters such as the center radio frequency and bandwidth change. Because sampling periods are typically on the order of nanoseconds, while metadata parameters such as center frequency change on the order of milliseconds, the time to transmit the metadata will be negligible compared to the time to transmit the data packets.

The proposed interface promotes modularity by being completely independent of the PHY and MAC used by the wireless system, and by being independent of the specific technique to carry these packets. For example, the connection between the RF front end and the digital system can be implemented using Gigabit Ethernet. Although the connection could reside at layer 2 and below, it could also be implemented as a full network stack with either TCP or User Datagram Protocol, for example.

Reference points, stream IDs, and timestamps

We specify where the metadata applies using the concept of reference points. For instance, the center frequency or power-level characteristics are unclear if the reference point is unknown. Therefore, metadata packets include reference points.

The ability to multiplex different packet types onto the same connection is supported by the concept of Stream Identifiers or IDs. Data and metadata packets that share a stream identifier have a one-to-one relationship, referred to as *data-metadata pairing.* A data packet stream and a paired metadata packet stream form an information stream. For example, a data packet will be transmitted with the RF parameters specified in the most recent paired control packet. There could be multiple information streams, for example, if there are multiple spatial streams.

All six packet types can include timestamps to enable synchronization. For example, control

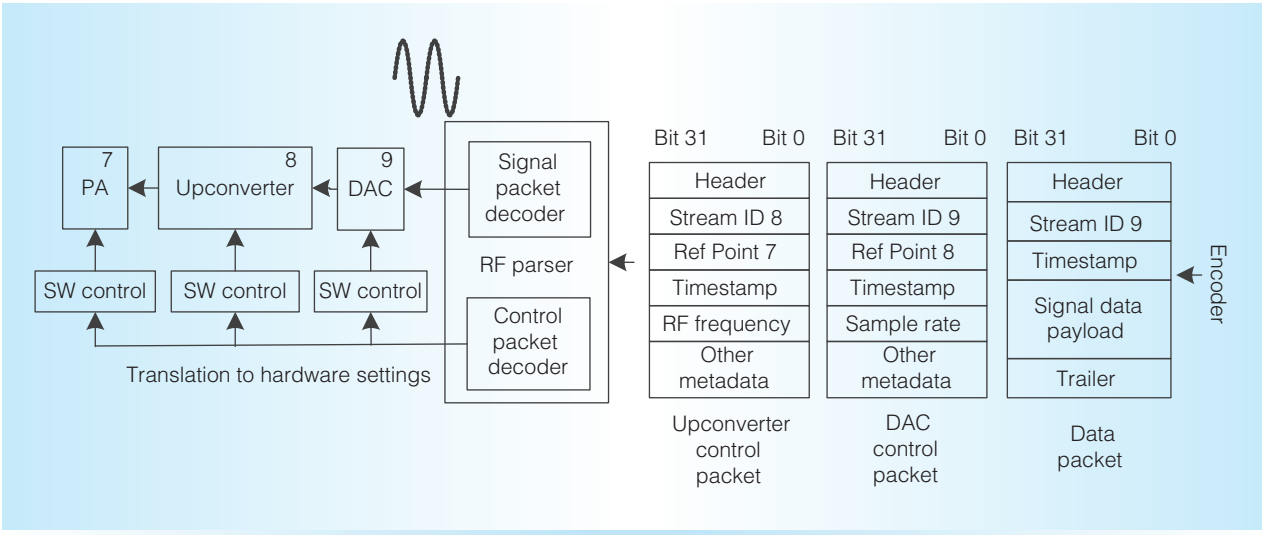

Figure 3. Radio transmitter operating with the proposed control and data packets. Three reference points—a digital IF signal, the DAC output, and an RF signal—describe the RF front end's topology at a coarse level.

packets are time-stamped so that the software-defined elements in the RF front end are configured at the specified time instant.

The timestamp in data packets sent from the RF encoder indicates the local time at the time of timestamping. However, it's often desirable to use the antenna as a reference point and convey accurately the instant a signal is impinging on the antenna. A signal that arrives at the antenna will emerge from the RF encoder with a certain delay due to the delays introduced by the various electronic circuits in the RF front end. This is why there is a timestamp adjustment field to account for all delays prior to attaching the timestamp. If we want the reference point to be the antenna, then negative timestamp adjustments can be included, indicating that events at the reference point occurred earlier than the timestamp specifies. The timestamp adjustment is a single value that's specific for the RF front end and is useful because it does not have to be equal to a multiple of the sampling period. For data packets sent to the RF front end, the timestamp refers to the time to start emission of the data, keeping in mind that the actual transmission will start with a delay that is equal to the timestamp adjustment.

Note that for synchronization timestamps are required, but not sufficient. Together with timestamps, GPS can achieve synchronization. However, there are other techniques for clock and time-of-day synchronization, even without GPS.[7] In this way, even radios that are at geographically different locations can operate synchronously.

Figure 3 illustrates the concepts discussed so far. The figure has three reference points—the digital IF signal fed to the DAC, the DAC output, and the RF signal at the output of the up-converter, which are assigned IDs of 9, 8, and 7, correspondingly. These reference points describe the RF front end's topology. Two control packets are shown—one for the DAC and one for the up-converter, which is described by RF, IF, and bandwidth. An IF frequency of zero can indicate up-conversion from baseband. We can describe the DAC using sample rate, number of bits, and reference level.

Figure 3 also shows the format of the packets. The first word is a header that contains information about the packet type and packet size. The header is followed by a stream identifier, a reference point, and an optional timestamp. For data packets, the time stamp is followed by the payload. For context and control packets, the timestamp is followed by metadata parameters.

Examples

We can combine the metadata descriptions of several circuit elements; ultimately,

all transmit or receive circuit elements of the SDR architecture can be described with a single packet. Therefore, this technique is in fact a hierarchical description language for the RF front end. Then, we can aggregate the control and context to and from each component and generate aggregate control or context packets. This aggregate control or context packet describes the entire RF front end with the reference point being the antenna. For example, we can combine the up-converter and the DAC control packets in Figure 3 in a single control packet.

Furthermore, the ongoing evolution of wireless technology is leading to systems in which there is coordination and cooperation among radios to reduce the probability of error. For example, if data is relayed, two or more independent copies are received at the destination, improving the robustness. Figure 4 illustrates the operation of a relay with aggregate context and control packets that are paired with the respective data packets. Both amplify-and-forward and decode-and-forward relaying techniques can be used.

A related development is distributed antenna systems or SDR clouds,[8] in which the RF front end is collocated with the antenna, whereas the digital signal processing is performed remotely at a data center, conceptually similar to cloud computing. Figure 5 illustrates an SDR cloud that has two information

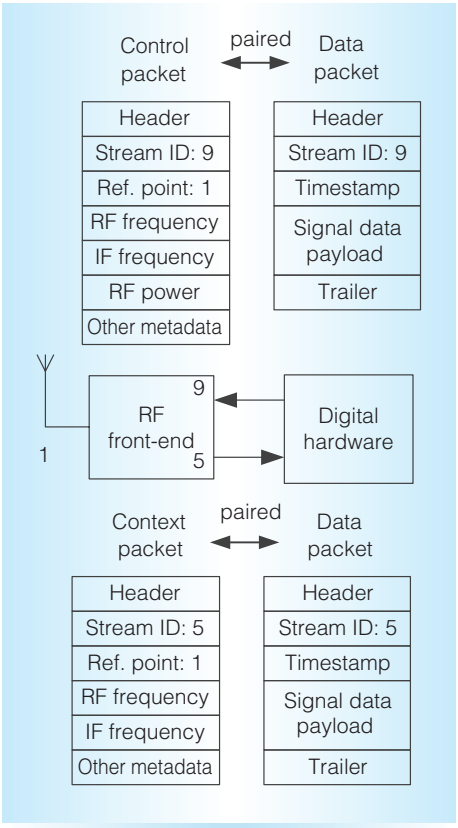

Figure 4. Wireless relay. According to the proposed architecture, an RF front end can implement a repeater by itself, without the digital hardware.

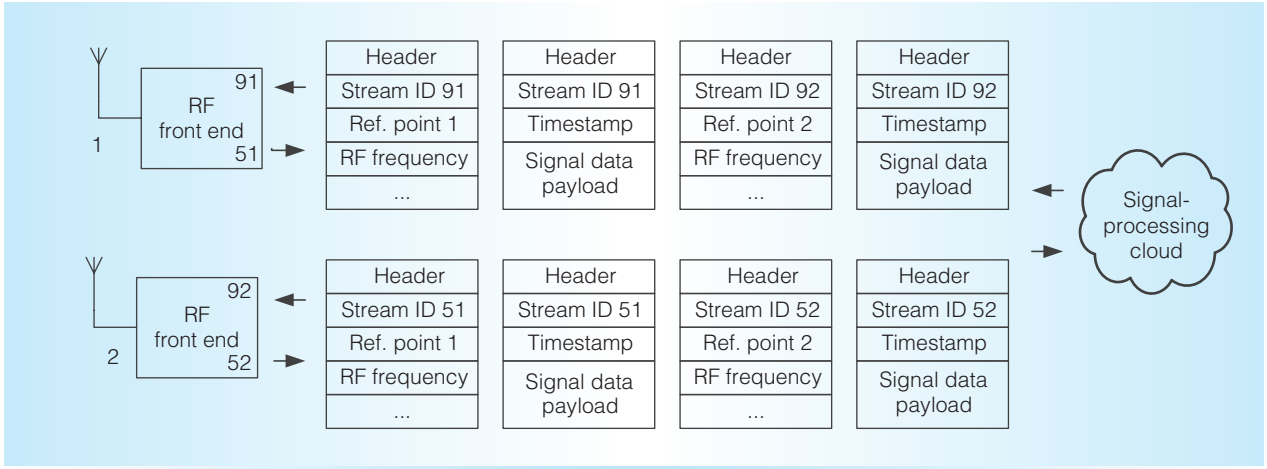

Figure 5. A multiple-antenna SDR cloud. The proposed RF-digital interface becomes the interface between the cloud and the RF front end.

streams, one for every spatial stream. This SDR cloud supports advanced wireless techniques—such as joint transmission—that require coordination among the spatial streams. In joint transmission, multiple devices jointly transmit data to another device to improve the received signal's quality, and the defined metadata packets can accomplish the required coordination. Note that in SDR clouds, the distinction between radio and RF front end is becoming blurred, and RF-digital interface becomes the interface between the RF front end and the cloud.

Metadata packet descriptions using RDF ontology

The ITU defines *cognitive radio* as a radio that can "obtain knowledge of its operational and geographical environment, established policies and its internal state; to dynamically and autonomously adjust its operational parameters and protocols according to its obtained knowledge in order to achieve predefined objectives; and to learn from the results obtained."[2]

Therefore, a cognitive radio must have domain knowledge of radio communication. On the basis of this knowledge, the CE can optimize the various parameters and protocols. There is a difference between the CE and other applications such as web browsing; the CE must have hardware-specific knowledge. Some metadata parameters such as center RF and power level are physically determined only by the RF front end. On the other hand, it is desirable to isolate the CE from the underlying hardware and make it fully portable (Figure 1). This requires some hardware abstraction. Therefore, for cognitive radios supporting multiple radio protocols, an open RF-digital interface is more appropriate than a closed interface for every radio system.

Ontology is a general mechanism to describe objects in a certain domain and the relationships among these objects. A cognitive radio ontology has been developed[9]; it provides a Web Ontology Language (OWL) representation of the Transmitter API.[5] This ontology tries to define all possible radio protocols and describes the basic terms of wireless communications, such as "bit," "symbol," and "channel model."[9]

Our focus is on the RF-digital interface, which contains the minimum set of parameters that must be described. The parameters defined in the metadata packets must be part of any radio ontology, because these parameters describe the entire RF front end.

A radio might be asked, "What are the minimum and maximum RF at which you can transmit?" or "What are the minimum and maximum bandwidths at which you can operate?" For radios to understand such questions, they must not only speak a common language[10] but they must also have explicitly defined metadata. To enable such interactions, it is convenient to use a simple ontology scheme such as the Resource Description Framework (RDF).[10] It describes things using triplets—for example, subject, predicate, and object. The subject is the resource being described. The predicate is a property of the subject, and the object field contains the value of this property. The mapping between such ontology descriptions and the context and control packets is one to one.

This description method allows queries to be made and answered. The query is a series of triplets in which some of the slots contain variables. For example, a query about the minimum and maximum RF would be:

```
PREFIX sdr: <http://opensdr.org/sdr/>
SELECT ?xmin ?xmax
WHERE {
        ?x sdr:MinRFfrequency ?xmin .
        ?x sdr:MaxRFfrequency ?xmax .
}
```

If this device operates between 5,000 and 6,000 MHz, the response would be:

```
<sdr:Device rdf:about="#DeviceID"/>
<sdr:MinRFfrequency
rdf:resource="#5000 MHz"/>
<sdr:MaxRFfrequency
rdf:resource="#6000 MHz"/>
```

The #-prefix states that the device is defined in the local namespace. With this description method, a radio can also be asked to adjust its operating parameters. For example, to ask the radio to tune to 5,200 MHz, we can use:

```
<sdr:Device rdf:about="#DeviceID"/>
<sdr:Purpose rdf:about="ChangeState"/>
```

```
<sdr:RFfrequency rdf:resource="#5200 MHz"/>
```

If the devices are synchronized, then we can add `<sdr:Timestamp rdf:resource="#value"/>` to specify an exact time instant when these parameters should be changed.

All metadata packets—the control, the context, and their extensions—can be represented using such formal, computer-processable semantics. Note that the sequence of descriptions doesn't matter. RDF applications do not have to understand the complete description. They look for parts they understand and ignore the rest. This lets the ontology be extended while maintaining complete backwards compatibility. Therefore, this is a method to describe completely the current operational status and capabilities of RF components.

An open RF-digital interface, where the RF front end and the digital hardware are replaced independently of each other, is possible. Metadata is defined explicitly, which facilitates cognitive radios that require metadata anyway. This interface makes advanced architectures such as SDR clouds practical.

The metadata can be described using ontologies. One task for future work is to develop a comprehensive wireless ontology. MICRO

Acknowledgments

This work was facilitated by a visiting fellowship sponsored by the National Institute of Communication Technologies (NICT) of Japan. The authors also thank the reviewers for their in-depth reviews and comments.

References

1. T. Hentschel, M. Henker, and G.P. Fettweis, "The Digital Front-End of Software Radio Terminals," *IEEE Personal Communications,* vol. 6, no. 4, 1999, pp. 40-46.
2. *Definitions of Software Defined Radio (SDR) and Cognitive Radio System (CRS),* report ITU-R SM.2152, International Telecommunication Union, 2009.
3. *Software Defined Radio Commercial Handset Guidelines,* SDRF-04-S-006-V1.0.0, SDR Forum, 2004; http://groups.winnforum.org/Specifications.
4. *Software Communications Architecture 4.0 Specification,* Joint Tactical Networking Center, http://jpeojtrs.mil/sca.
5. *Transceiver Facility Specification,* Wireless Innovation Forum, SDRF-08-S-0008-V1.0.0, 2009.
6. *VITA Radio Transport (VRT) Standard,* ANSI/VITA 49.0-2009, VITA Standards Organization, 2009.
7. T. Cooklev, A. Pakdaman, and J. Eidson, "IEEE 1588 Over IEEE 802.11 for Synchronization of Wireless Local Area Nodes," *IEEE Trans. Instrumentation and Measurement,* vol. 56, no. 5, 2007, pp. 1532-1539.
8. I. Gomez, V. Marojevic, and A. Gelonch, "Resource Management for Software-Defined Radio Clouds," *IEEE Micro,* vol. 32, no. 1, 2012, pp. 44-53.
9. *Description of the Cognitive Radio Ontology,* WINN-10-S-007, Wireless Innovation Forum, 2010.
10. M. Kokar and L. Lechowicz, "Language Issues for Cognitive Radio," *Proc. IEEE,* vol. 97, no. 4, 2009, pp. 689-707.

Todor Cooklev is the ITT Associate Professor of Wireless Communication and Applied Research and founding director of the Wireless Technology Center at Indiana University–Purdue University Fort Wayne. His research interests include signal processing, radio architectures, and software-defined radio. Cooklev has a PhD in electrical engineering from the Tokyo Institute of Technology. He is a senior member of IEEE and the IEEE Standards Association.

Akinori Nishihara is a professor at the Tokyo Institute of Technology, where he has held several academic appointments. His research interests include digital signal processing, circuits and systems, and educational technology. Nishihara has a PhD from the Tokyo Institute of Technology. He is a fellow of IEEE and the IEICE.

Direct questions and comments about this article to Todor Cooklev, IPFW, 2101 East Coliseum Blvd., ET 229, Ft Wayne, IN 46805; cooklevt@ipfw.edu.