OpenRadio: A Programmable Wireless Dataplane

Manu Bansal, Jeffrey Mehlman, Sachin Katti, Philip Levis
Stanford University
{manub, jmehlman, skatti}@stanford.edu, pal@cs.stanford.edu

Abstract

We present OpenRadio, a novel design for a programmable wireless dataplane that provides modular and declarative programming interfaces across the entire wireless stack. Our key conceptual contribution is a principled refactoring of wireless protocols into processing and decision planes. The processing plane includes directed graphs of algorithmic actions (eg. 54Mbps OFDM WiFi or special encoding for video). The decision plane contains the logic which dictates which directed graph is used for a particular packet (eg. picking between data and video graphs). The decoupling provides a declarative interface to program the platform while hiding all underlying complexity of execution. An operator only expresses decision plane rules and corresponding processing plane action graphs to assemble a protocol. The scoped interface allows us to build a dataplane that arguably provides the right tradeoff between performance and flexibility. Our current system is capable of realizing modern wireless protocols (WiFi, LTE) on off-the-shelf DSP chips while providing flexibility to modify the PHY and MAC layers to implement protocol optimizations.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Design, Management, Performance

Keywords

Programmable, Infrastructure, Dataplane, Wireless

1. INTRODUCTION

Wireless protocols evolve continuously. For example, 3GPP (the standards body responsible for specifying the LTE cellular standard) expects to release major updates to the basic LTE standard once every 18 months, and minor updates continuously. Even within a deployed operational network, operators and vendors need to continuously optimize the network to handle problems such as inter-cell interference, new traffic classes such as video and so on. For example, operators need to implement management mechanisms that dynamically adjust spectrum and power allocation at neighboring basestations at a fine-grained subcarrier level on timescales of hundreds of milliseconds to ensure that mobile handsets at the edge of both cell-sites are not adversely affected. Similarly, to cope with the growing popularity of video over spectrum starved cellular networks, operators would prefer to use a PHY layer optimization such as unequal error protection (UEP) - where certain frames (I-frames) are better protected than others (P- or B-frames) [15][25] - that can deliver much higher video quality while using lesser spectral resources. Similar protocol changes and optimizations abound (some more examples are discussed in Sec. 2) that are beneficial to implement for a variety of reasons.

The wireless infrastructure is expected to support such evolvability. In the past, such changes typically required replacing the basestations, since the protocol definition was closely coupled with the hardware in the form of specific ASICs designed for each protocol. Such an approach worked fine when protocol upgrades were relatively slow (several years) and basestations were deployed sparsely and covered large areas. However, in current networks, protocol changes are continuous (once every few months) and networks are dense (e.g. one basestation per block in urban areas) to provide high capacity and uniform coverage. Physically and frequently upgrading the basestations for such dense networks is prohibitively expensive.

In this paper, we argue that wireless infrastructure needs a programmable dataplane to support such evolvability. Specifically, the basestations should be remotely programmable to enable vendors and operators to upgrade and optimize the network completely in software. To build such functionality, we present OpenRadio, a novel design for a *programmable wireless network dataplane*. Built around a commodity multi-core hardware platform, the core component of OpenRadio is a software abstraction layer that exposes a modular and declarative interface to program the PHY (baseband) and MAC layers. An operator describes a protocol by defining *rules* to match subsets of traffic streams and specifying *actions* to process them. Rules are logical predicates on parameters of packets such as header fields, received signal strength, channel frequency and other fields that may be programmed. Actions describe behavior such as encoding/decoding of data and scheduling of traffic on the channel.

OpenRadio's design makes two contributions. First, it decouples wireless protocol definition from the hardware, even while ensuring that commodity multi-core platforms can be used for implementing the protocols. Such decoupling is feasible because we can identify low-level atomic building blocks in wireless protocols at the PHY

HotSDN'12, August 13, 2012, Helsinki, Finland.

and MAC layers that are shared across different protocols (different versions of LTE 4G, 3G, WiFi and so on). Further, OpenRadio shows that commodity multi-core platforms are capable of executing such atomic blocks efficiently. Consequently, defining and customizing protocol definition programmatically is feasible, as long as modular abstractions exist to re-program individual protocol elements.

OpenRadio's second contribution is the design of this software abstraction layer that exposes a modular and declarative interface to program wireless protocols. The key idea is a principled decomposition of wireless protocols into separate *processing* and *decision* plane components with a simple, well-understood API between them. This decomposition is the main enabler of an efficient implementation of a flexible and modular wireless programming interface. The processing plane includes *actions*: operations that only involve data-manipulation on packet data but no branching logic. The decision plane includes *rules*: all decision logic that picks sequences of operations to transform data stream subsets. Such decomposition lends itself to efficient real-time implementation on commodity digital signal processors (DSPs) that are widely available from a number of chip vendors [26, 12, 9, 14].

OpenRadio can be used to specify both the underlying protocols as well as optimizations. For example, a rule for WiFi would require all received packets matching the 6Mbps data rate in the PHY header to be processed with the action of BPSK demodulation. Protocol optimizations can be expressed by adding rules and actions to an existing definition. For video delivery, an operator using OpenRadio might optimize for this class of traffic by inserting a new rule to match all video packets (based on a header field). The rule would require them to be processed through a special action of UEP (en)decoding. The UEP (en)decoder action would also be programmed by the operator as an atomic block. Meanwhile, the existing standard WiFi-data handling remains untouched.

More generally, our longer-term goal is to design a programmable wireless dataplane that includes the basestations, backhaul network switches/routers and gateways to realize a software defined cellular infrastructure. Our goal in this paper, however, is more modest. It is to discuss the design of a programmable basestation that forms one (albeit important) component of this infrastructure. At this stage, we have a prototype implementation of OpenRadio that supports the WiFi PHY on off-the-shelf multicore DSP processors from TI [26]. The platform is capable of fully processing the WiFi PHY protocol in software running at the highest speeds (upto 108Mbps) and meeting all protocol deadlines. We pick WiFi for proof-of-concept since its processing requirements are the most stringent due to wide-bandwidth, multi-carrier modulation and random-access deadlines on the order of microseconds. We plan to add support for LTE in the near future, as well as leverage the dataplane to start researching the design of a control plane for cellular SDN infrastructure.

In the following section, we discuss more specific examples of the kinds of protocol changes that operators would want to implement in their networks remotely, and that OpenRadio's design supports.

2. USE CASES

Programmable basestations for cellular infrastructure are being increasingly favored over fixed-function hardware as exemplified by offerings of "software-upgradable" platforms for HSPA/WCDMA/LTE from some of the top vendors [23, 3]. Still, many desirable features either remain hard to program or are simply not possible to implement. A well-designed programmable dataplane can enable these features. Let us consider some specific use-cases to gain insight into the kind of programmability required.

2.1 Cell-size based optimization

OFDMA leads to power-amplifier (PA) inefficiencies due to a high peak-to-average-power ratio (PAPR) as compared to single-carrier modulation. This lead to the choice of SC-FDMA for LTE (Rel 8/9) uplink to save user equipment (UE) power. However, advances in PA technology have now led to the adoption of OFDMA (Rel 10) in favor of higher spectral efficiency. This is especially true for small-cells where the UE needs to transmit at a lower power than a macro-cell. The switch from SC-FDMA to OFDMA effectively involves toggling a DFT block on the PHY signal chain. Definitions of individual processing blocks remain unchanged.

Another cell-size based optimization could involve picking the right FFT size to balance multipath effects against PAPR inefficiencies. A higher FFT size leads to better multipath resiliency but a higher PAPR value which is less energy efficient. A smaller cell can use a smaller FFT size for the same data delivery performance since it suffers less multipath fading and delay spread. Similar FFT size adaptation based on link length can also benefit WiFi. The FFT processing block only needs to be configurable in FFT size without any changes in the algorithm.

2.2 Co-existence of heterogeneous cells

Cellular networks are rapidly becoming heterogeneous in cell size with deployment of small-cells. Femto-cells enhance coverage in a small area such as a home but generally provide access to a closed group of subscribers. However, deployment in the same frequency as the containing macro-cell can cause strong interference to external users in the same region. This has led to techniques like Almost Blank Subframes (ABS) [6] that co-ordinate the use of time-frequency resource elements between the two cells to avoid interference. Such co-ordination can be implemented with programmability in the MAC scheduler that decides whether or not to use a resource element at any given time.

2.3 Application-specific wireless service

A promising feature of LTE is multiple service profiles [4] with varying throughput and latency characteristics. The intent of the QoS profiles is to provide customized network service to diverse applications. However, the standard only specifies profile parameters while leaving the mechanisms open to implementation. A programmable dataplane provides the vehicle necessary to implement various profiles and allowing their extension to new application classes. Different processing chains can be tied to different traffic classes. Prioritization, scheduling of traffic and error recovery mechanisms can also be programmed according to application needs. Those features can also evolve with new applications such as public safety networks, urban sensing and media broadcast in addition to classic voice, video and data delivery. A programmable dataplane can similarly facilitate implementation of QoS features for WiFi, WiMAX, White-space networking etc.

2.4 Evolving standards

Cellular wireless standards have been upgrading every couple of years [1, 2]. Users are ready to consume network upgrades at an even faster pace with an average UE lifespan of 18 months [10]. Yet, standards take a decade or more to reach their peak market share [11]. Arguably, the biggest impediment to evolution of the infrastructure is its inability to be programmed in-place and incrementally.

Wireless networks are already operating with very high spectral efficiency (within $\approx 20\%$ of capacity limits). Consequently, signal processing motifs are stabilizing with little room for improvement. Moreover, cellular networks are designed such that most intelligence is place in the infrastructure while keeping UEs simple. This allows a bigger playing fields on the base-station side that is still interoperable with existing users. Most gains are to be had by optimizing networks according to the operational scenario and load patterns, which tend to be very diverse [11]. At the same time, the set of users is always a rich mix of capabilities as new standards take long to penetrate. The ideal base-station must be able to cater to all of them simultaneously. OpenRadio provides a platform where new protocols and features can be "plugged-in" seamlessly. Its programmable dataplane provides flexibility in selection, configuration and arrangement of data processing motifs to compose protocols optimized to the operational scenario.

3. DESIGN GOALS

Our goal is to design a programmable wireless dataplane with a modular interface that can provide both the performance and flexibility necessary for implementing a wide class of modern wireless protocols and their optimizations.

As a *performance* benchmark, the platform must be capable of processing a bandwidth of at least 20MHz for OFDM-complexity protocols, including but not limited to IEEE 802.11 WiFi, IEEE 802.16 WiMAX and 3GPP LTE-Advanced. The platform must provide enough computation throughput for these applications as well as the processing latency requirements associated with specific protocols. In the case of WiFi, the most challenging deadline is an ACK-turnaround time of $25\mu s$ for every successfully decoded data packet.

Flexibility is a more qualitative characteristic than performance. Our design goal is to provision the system to allow a smooth tradeoff between performance and deviation from the standard configuration. Small variations in behavior should be enabled with little or no performance loss. For example, a version of WiFi optimized for video delivery using UEP only requires a change in the encoding of I- and B-/P-frame bits into a packet of the PHY layer. All other operations are preserved. Consequently, the platform must be able to deliver near-WiFi computation performance for video-WiFi. At the same time, the platform should not preclude more significant changes from being realized at all.

An equally important design goal is making the OpenRadio platform easy to program. The system must provide *modularity* such that the involvement of a programmer is limited to only the affected components of the protocol. Adapting a WiFi implementation to video-WiFi should not require the operator to modify or even understand the OFDM (de)modulator or the CRC implementation. Without such modularity, the utility of a flexible dataplane will be far from practical.

4. PROGRAMMING ABSTRACTIONS

The OpenRadio dataplane design provides a network substrate that can be programmed using a modular, declarative interface, provides high performance capable of supporting the highest rate modern wireless protocols, and can be built using commodity DSP silicon. In this section, we describe its declarative interface.

4.1 Modular declarative interface

OpenRadio's dataplane enables the network operator to declaratively program the network stack for specific traffic subsets. He only specifies what packet processing behavior he expects from the network for different traffic classes. However, the operator does not need to specify how the packet processing behavior is integrated into an operational network and realized in practice. This low-level complexity is abstracted out by the interface.

OpenRadio's declarative interface is built on a *principled refactoring of decision and processing paths in wireless protocols*. Specifically, we observe that protocols can be finely partitioned into two separate parts: processing blocks which specify how a protocol transforms analog waveforms into bits, and decision logic which specifies when different processing transformations are used. For example, the 54Mbps OFDM decoding sequence of 802.11g is a processing specification, while interpreting the signal field of the 802.11 PHY layer control header to determine that the payload will be at 54Mbps is a decision specification.

The processing plane encapsulates the series of signal processing algorithms used in a PHY processing chain where each module corresponds to a single algorithm. For example, the processing plane will have modules for FFT, 64-QAM mapping and slicing, convolutional encoding, Viterbi decoding, etc. These modules correspond to *blocks* in a signal processing chain. The decision plane encapsulates all the decision logic functionality. It is responsible for selecting processing plane blocks to be used for encoding or decoding a packet. For example, the decision plane logic for 802.11g will appropriately configure and stitch the processing plane plane blocks to decode a 54Mbps packet without needing to touch any block internals. Similarly, a protocol modification that rearranges the same signal processing blocks would only need to modify the decision plane, leaving the processing plane blocks untouched. As another example, LTE changes the way packets are processed on the downlink and uplink. It uses standard multicarrier OFDM for the downlink, while it uses a modified single carrier modulation (SC-FDM) for uplink to conserve mobile client batteries [5]. The details of the signal processing algorithms are unimportant, but the main take-away is that both OFDM and SC-FDM are simply different ways of configuring and stitching basic signal processing blocks such as FFTs, channel decoders and slicers. The decision plane can simply specify such re-configurations and re-arrangements to realize different protocol optimizations.

The two key choices - granularity of processing plane blocks and decoupling decision plane logic from the processing plane - are directly motivated by the goal of modular programmability. Decoupling allows logical changes and rearrangement of signal processing blocks to be limited to the decision plane without requiring any understanding of the processing plane. Reconfiguring the behavior of a processing plane block limits changes to the specific block without affecting other blocks or the decision plane. Adding a new block (eg. a new channel encoder or decoder optimized for video) is similarly a contained effort. The level of abstraction of processing plane blocks (eg. FFT vs. low level instructions such as complex multiply and accumulate) is high enough to allow most programmability to be expressed declaratively and allows convenient reuse of code. The imperative part is abstracted out in processing plane blocks. Such separation of processing and decision planes allows an intuitive programming model made up of rules and actions.

4.2 Rule-action programming model

The decision plane of a protocol has a natural representation as a set of *rules* encoding the decision logic. These rules govern the processing plane by naming *actions* that must be performed depending on the outcome of rule evaluation. For example, a rule-action pair will read `if hdr.rate == 54Mbps, perform action_decode_54Mbps`.

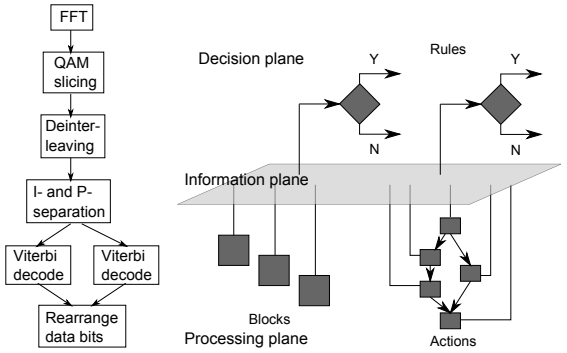

Figure 1: (a) Directed acyclic graph of an action customized for video delivery. (b) OpenRadio programming model depicting the separation of decision and processing planes and their relation with the information plane.

Actions are directed acyclic graphs (DAGs) of signal-processing operations. Each operation is typically simple: it takes values from an input stream, applies a simple operation such as an FFT, and outputs the processed stream. Operations do not include conditionals or other complex, dynamic logic since that is factored out into the decision plane. An example action DAG for a video decoding chain is shown in fig. 1(a).

The regular interface of a processing plane block is composed of the data inputs (eg. complex samples) and data output (eg. soft bits, complex samples). In addition, a block has configuration inputs (eg. FFT size) and output statistics (eg. energy value for a packet-detect block, rate field for a header-parsing block). Collection of all configurations and statistics from the processing plane forms an *information plane* that acts as the bi-directional interface between decision and processing planes (fig. 1(b)). Rules are written on elements of the information plane. Similarly, the decision plane exercises control over the processing plane through configuration primitives made available on the information plane.

The programming model of rules and actions is a specific concrete representation of the concept of decision and processing plane decoupling. It lets us express protocols declaratively by preserving the decoupling semantics. Actions only specify the functional behavior and data dependencies without specifying any execution schedules, hardware placement details or resource allocation. Similarly, rules only specify the logical content without dictating how they will be implemented. Thus, the model provides a generic API without being tied to any particular system design or hardware platform architecture. It also allows us to leverage extensive prior work where the stream computation in actions is naturally expressed using dataflow languages [16]. The current implementation of OpenRadio uses annotated C code, but its decoupled programming model makes it simple to integrate new domain-specific dataflow programming languages.

4.3 State machine model and deadlines

Rules and actions are assembled into a state machine to express a fully-functional protocol, as in fig. 2. Each state specifies exactly one action to be performed and contains rules for transitioning to other states. For example, the header decode state will perform the action of decoding the header field and include rules to transition to data-decoding states, one for each data rate. Such a state machine model matches naturally with the specification of protocols, some of which use formal state machine languages in the standard itself.

In addition to rules and actions, a first class component of wireless protocols is deadlines. Deadlines may apply to specific actions such as a maximum of $1\mu s$ delay in assessing channel idle/busy state for effective channel backoff implementation in WiFi (i.e. slot-time stipulation), or to a series of actions such as a maximum delay of $25\mu s$ in decoding a packet from the end-of-arrival (due to WiFi ACK-turnaround stipulation.) In the latter case, they exist between certain states, and consist of hard timing requirements on all of the computation (decision and processing-plane) that occurs in the interval between those two states. Valid deadlines may exist between any two states in a protocol but there must be a logical route through the rules from the initial state to the end state.

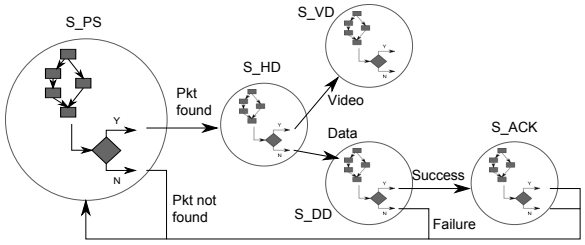

Figure 2: A state machine with states for packet search (S_PS), header decode (S_HD), video decode (S_VD), data decode (S_DD) and sending an acknowledgement (S_ACK).

Similar to the choice of rule-action model, the state machine model is a specific choice for representing protocols that is intuitive and preserves the declarative nature of the programming interface including specification of deadlines. Further, it allows us to leverage prior work on formal state machine languages like [7] with strong semantics to facilitate static analysis that is critical to providing guaranteed execution times and meeting deadlines as we discuss further in sec. 5.

4.4 Challenges

The key challenge in OpenRadio is efficient realization of the decoupled declarative programming model. In particular, the system must be able to meet protocol deadlines reliably without requiring the programmer to optimize implementation specifics. We can summarize the challenges as follows:

- Efficient execution of computationally demanding processing plane actions (≈100 Gflops) while retaining hardware abstraction, programming modularity and flexibility.
- Maintaining realtime property over a sequence of states, i.e., under serial composition, in order to meet deadlines reliably.

The rich programming interface made up of rules-actions and state-machine models is the key feature that distinguishes OpenRadio from a bare-bones software-radio platform or a programmable substrate like an FPGA platform that may offer similar performance. Let us discuss the principles behind an efficient implementation of this programming interface.

5. DESIGN AND IMPLEMENTATION

Decoupling of decision and processing planes allows us to provide a modular and declarative programming interface through the rule, action and state machine models. Can we design a system that implements such an interface with the level of computational performance needed for modern protocols with 20-80MHz of bandwidth while meeting tight latency deadlines on the order of tens of microseconds? We answer the question in the affirmative in this

Processing plane block	WiFi	LTE	3G	DVB-T
FIR/IIR	✓	✓	✓	✓
Correlation	✓	✓	✓	✓
Spreading			✓	
FFT	✓	✓		✓
Channel Estimation	✓	✓	✓	✓
QAM Mapping	✓	✓	✓	✓
Interleaving	✓	✓	✓	✓
Convolutional Coding	✓	✓	✓	✓
Turbo Coding		✓	✓	
Randomization	✓	✓	✓	✓
CRC	✓	✓	✓	

Table 1: Many computationally intensive processing plane blocks are common to all modern PHYs, differing only in configuration.

section describing the design and early results demonstrating its feasibility. Our system is based on two keys principles.

5.1 Design principles

Learning from existing programmable solutions, OpenRadio follows *the principle of scoping flexibility*. We aim to provide enough flexibility to implement most protocols and optimizations of interest but no more. For example, the FFT block only needs to provide configurable FFT-length, not a choice of different Fourier Transform algorithms. Similarly, the convolutional decoder needs to be configurable in the coding rate (say, 1/2, 1/3, 1/4, 1/5) or constraint length over small parameter ranges but the actual algorithm (Viterbi or BCJR) is immaterial. There is, however, still scope for blocks that need almost arbitrary flexibility such as a bit interleaver.

Restricting the scope of flexibility where advantageous allows for optimizing the hardware to perform some of the relatively-fixed operations extremely efficiently while being able to provide the requisite configurability. The merit of this approach lies in the observation that many of the heavy computational processing plane blocks of modern PHY protocols are common to most standards and fall in this category (Table 1). Further, commodity DSP platforms with hybrid architectures optimizing common *compute kernels* are becoming widely available, either through loosely-coupled on-chip accelerators as in TI KeyStone [26] or through tightly-coupled special-purpose instructions as in CEVA-XC Family [9], [20][14][12]. A well-designed accelerator can provide speed-ups of up to an order of magnitude (10-25x) over pure-software [13]. As a price reference point, a TI KeyStone chip housing tens of accelerators is marked at less than $200 and is capable of processing 80MHz WiFi-like intensity of computation, i.e. it easily supports WiFi, WiMAX, LTE and W-CDMA.

While the above hardware substrate provides the raw horsepower needed for a programmable dataplane, OpenRadio's dataplane must provide a modular declarative interface while ensuring that packets are processed within the deadlines of the protocol. To this end, OpenRadio follows *the key principle of decision and processing plane separation throughout its system design*. In contrast with providing modularity and declarative interfaces, the motivation here is to achieve near-deterministic execution times for processing plane actions. On a hybrid architecture, blocks implemented through hardware accelerators would finish in predictable amounts of time by default. This may not be true of other software-based blocks that will run on CPUs or DSPs. The algorithmic reason for such unpredictability is presence of branches in arbitrary code. For example, a soft-slicer block that could do any of BPSK, QPSK, 16-QAM or 64-QAM, will run for different amounts of time in each mode. However, a branch is precisely a decision plane rule. The factoring of algorithm and code into decision and processing planes pushes out all such branches to the decision plane, leaving only branch-free code in processing plane blocks.

Combining the decoupling principle with assisting implementation choices, we not only achieve deterministic-runtime blocks but also preserve the property under composition as required for actions. This is the key property that allows OpenRadio to systematically meet execution deadlines. An added advantage of this principle is a direct mapping of declarative programs to execution primitives which greatly simplifies the system design.

5.2 Preliminary implementation

The design of OpenRadio system is focused towards utilizing multi-core DSP architectures. They typically contain DSP cores optimized for signal processing computation and hardware accelerators providing speed-up of specific, commonly used algorithms. Such platforms provide the most desired tradeoff between performance and flexibility. Our specific hardware choice is the TI KeyStone architecture [26]. The software challenge is to harness the raw compute horsepower while retaining modular and declarative abstractions.

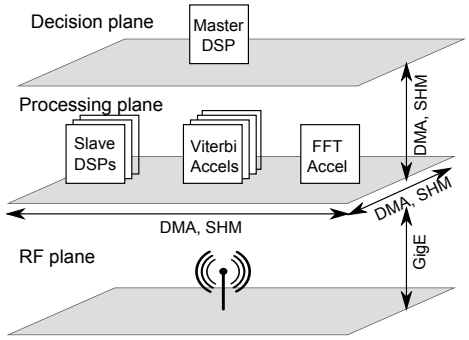

Figure 3: OpenRadio basestation internals with different kinds of compute cores and an RF frontend.

The core objective of the runtime system is *fast and deterministic execution of actions*. While efficient execution is achieved through accelerated hardware, determinism in runtime is obtained through decision/processing separation down to the software level, mimicking the programming model itself. The system architecture is shown in fig. 3. The same separation is applied to hardware resources as well, designating the decision plane core as the master core controlling the processing plane slave cores and accelerators.

The decision plane runtime emulates the protocol state-machine by evaluating rules as transitions are made between states. The action associated with each state is computed on the processing plane by assigning constituent blocks to slave cores (or accelerators) according to a pre-computed execution and resource allocation schedule. Such static scheduling is made possible because of our specific implementation choices such as software-addressed L2 memory and event-polling instead of interrupts. They eliminate major sources of indeterminism from the underlying hardware architecture as well. Further, using a message-passing model for inter-processor communication allows accurate accounting of IPC latencies. Full description of implementation and scheduling framework are beyond the scope of this paper. However, it is worth stating that the scheduling of actions in our deterministic model bears strong resemblance with instruction loop-pipelining in compilers. This reduction of model allows us to leverage compiler scheduling algorithms as in [24].

Our early prototype is capable of processing full 54Mbps 802.11g PHY layer stack on a single KeyStone chip with more than one full DSP core worth of computer power still available for implementing protocol modifications. The utilization figure includes the overhead introduced by the runtime software that strives to maintain modularity. This overhead measures to roughly 20% of the entire compute load.

5.3 Design limitations

Our software system design applies equally well to other multi-core processors, hybrid or symmetric. Still, it is based on certain judicious assumptions.

First, OpenRadio assumes that the majority of the computation time is spent in the processing plane of the protocol, and decision overhead is negligible. However, heavy decision planes would incur prohibitive inter-core communication overheads implying that decision/processing plane separation is not the best design choice. Similarly, a heavy sequential algorithm on the processing plane would restrict the benefits of a multi-core architecture if it forbids pipelined parallelism.

Second, OpenRadio assumes decision/processing refactoring is meaningful in signal processing blocks. However, some blocks that need to take decisions, like packet-search block, are best expressed with some embedded decision code. If the gap in average and worst-case run times becomes too high, maintaining the deterministic semantic on that block could become very inefficient.

6. RELATED WORK AND DISCUSSION

OpenRadio has a lot in common with SDN efforts such as OpenFlow [18], both in goals and in challenges. The key design consideration is to pick the right abstractions to balance flexibility against performance. However, there are some major differences too. Interoperation with existing devices on the Internet is a major consideration for layer-3 SDN interfaces. For OpenRadio, inter-operation is much less of an issue as it operates on layers 1 and 2 which are contained within autonomous domains. Further, a programmable wireless dataplane has a massive computational load that must be processed under strict deadlines, a challenge unique to wireless.

There are comparable solutions for programmable wireless dataplanes that excel in some dimension at the cost of others. GNURadio [8] offers unlimited flexibility through purely software-based processing and convenient programming through high-level interfaces at the cost of performance and real-time guarantees. SORA [27] offers full flexibility and sufficient performance while arguably sacrificing ease of programming by nearly by-passing the desktop operating system. WARP [17] and AirBlue [22] use an FPGA approach to retain flexibility equal to software while gaining performance but end up driving the price-point high. Others like NI PXIe-1082 SDR [21] have taken an all-out approach with unrestrained FPGA and CPU horsepower resulting in extremely expensive systems (≈$50k per SDR platform) only good for small-scale laboratory research and prototyping. Yet other cellular domain platforms [3][26, 12, 9, 14] tend to provide the right price-performance and flexibility point but remain hard to program. This has led to the development of higher-level interfaces [22, 19] to work with these platforms.

OpenRadio strikes a unique combination of the programmability-performance-price metrics by treating all three as primary objectives from the get-go. The design is based on principled refactoring of wireless protocols into processing and decision planes. It is general enough to be implemented on a variety of platforms including DSPs, FPGAs and multi-core desktop machines. Code has enough structure that it can be statically analyzed to yield guaranteed real-time performance and efficient hardware use while still exposing modular, declarative interfaces.

Our intent is to provide programmability across the entire wireless stack. In this paper, most of our design choices are guided by PHY-layer considerations. The decision and processing plane model applies equally well to the MAC layer - the processing plane operation is to map multiple traffic queues onto the channel frequency-time grid while the decision plane operation is to pick the right mapping according to a programmable algorithm. The weighting of computation between the two planes may be different than that in PHY. This exploration is part of our ongoing research.

7. REFERENCES

[1] http://en.wikipedia.org/wiki/3GPP.

[2] http://en.wikipedia.org/wiki/Comparison_of_mobile_phone_standards.

[3] RBS6000 Base Station. *http://www.ericsson.com/ourportfolio/products/base-stations*.

[4] 3GPP TS 23.203 V8.3.1.

[5] 3GPP TS 36.201 - v1.0.0. LTE Physical Layer - General Description.

[6] Alcatel-Lucent. R1-093340: Blank Subframes for LTE. *3GPP TSG RAN WG1,meeting 58, Shenzhen, China, August 2009*.

[7] M. Anlauff. XASM - An Extensible, Component-Based Abstract State Machines Language. In *International Workshop on Abstract State Machines, Lecture Notes on Computer Science (LNCS)*, pages 69–90. Springer-Verlag, 2000.

[8] E. Blossom. GNU radio: tools for exploring the radio frequency spectrum. *Linux J.*, 2004:4–, June 2004.

[9] CEVA, Inc. CEVA-XC High-Performance, Low-Power DSP Cores for Advanced Wireless Communications.

[10] Earth911.com. Facts about cell phones. *http://earth911.com/recycling/electronics/cell-phone/facts-about-cell-phones/*.

[11] Ericsson Corp. Traffic and Market Report, June 2012. *http://www.ericsson.com/res/docs/2012/traffic_and_market_report_june_2012.pdf*.

[12] Freescale, Inc. StarCore Digital Signal Processors.

[13] R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B. C. Lee, S. Richardson, C. Kozyrakis, and M. Horowitz. Understanding sources of inefficiency in general-purpose chips. ISCA '10. ACM, 2010.

[14] Intel Corporation, Inc. Intel Solutions for the Next Generation Multi-Radio Basestation. *Application Note*.

[15] ISO/IEC Moving Picture Experts Group (MPEG). The MPEG4 Standard. *ISO/IEC 14496 - Coding of audio-visual objects*, 1998.

[16] W. M. Johnston, J. R. P. Hanna, and R. J. Millar. Advances in dataflow programming languages. *ACM Comput. Surv.*, 36(1):1–34, Mar. 2004.

[17] A. Khattab. WARP: A flexible platform for clean-slate wireless medium access protocol design. *SIGMOBILE Mob. Comp. Comm.*, January 2008.

[18] N. McKeown. OpenFlow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38:69–74, March 2008.

[19] Microsoft Research. Brick Specification. *The SORA Project*, 2011.

[20] Mindspeed Technologies, Inc. MindSpeed Baseband Processors.

[21] National Instruments, Inc. NI PXIe-1082 SDR Chassis. *http://sine.ni.com/nips/cds/view/p/lang/en/nid/207346*.

[22] A. Ng, K. E. Fleming, M. Vutukuru, S. Gross, Arvind, and H. Balakrishnan. Airblue: A System for Cross-Layer Wireless Protocol Development. 2010.

[23] Nokia Siemens Networks. Flexi Multiradio BTS. *http://www.nokiasiemensnetworks.com/portfolio/products/mobile-broadband/single-ran-advanced/flexi-multiradio-base-station*.

[24] J. Sánchez and A. González. The Effectiveness of Loop Unrolling for Modulo Scheduling in Clustered VLIW Architectures. ICPP '00. IEEE Computer Society, 2000.

[25] S. Sen, S. Gilani, S. Srinath, S. Schmitt, and S. Banerjee. Design and Implementation of an "Approximate" Communication System for Wireless Media Applications. In *ACM SIGCOMM*, 2009.

[26] Texas Instruments. TMS320TCI6616 Communications Infrastructure KeyStone SoC. *Data Manual*.

[27] J. Zhang. Experimenting software radio with the SORA platform. In *Proceedings of ACM SIGCOMM*, 2010.