PV-RAN for Whole-Stack Slicing

Matthias Sander-Frigau, Tianyi Zhang, Hongwei Zhang, Ahmed Kamal, Arun Somani

Presented by Matthias Sander-Frigau, Ph.D

Department of Electrical Engineering and Computer Science
Iowa State University – USA

CyNet for Wireless Living Labs

NSF CyNet project

- End-to-end Software-Defined Infrastructure for Smart Agriculture and Transportation (ECE, PSI, InTrans)
- Deployment of two field-deployed SDR-based RANs in Ames, IA, USA

Goals

- ➢ Support diverse applications, e.g.: NB-IoT & Predictable, Reliable, Real-time, and high-Throughput (PRRT)
- ➢ Allow domain scientists/ networking researchers to prototype and deploy solutions in-field

Smart transportation for R&E
AR-based transportation with field and cyber virtuality

Smart agriculture for R&E
Real-time camera imaging & real-time control of UGVs

Heterogeneous requirements
Throughput, timeliness, reliability, energy efficiency

Require isolated slices

The solution?

Slice-oriented solution...

What if networking researchers want to:

- Prototype and experiment novel PHY & MAC protocols together with existing solutions?
- Develop advanced wireless networking applications that have radically different requirements within a family of services (i.e.: uRLLC)?

Problem: How to allow RAN virtualization to use distinct PHY & MAC layers per slice on the same radio front-end (a.k.a. Whole-Stack Slicing)?

What we propose

- **PV-RAN (Paravirtualized RAN):** A lightweight, open-source platform that enables Whole-Stack Slicing where different PHY and MAC layers can be adopted at different RAN slices on the same physical radio device (SDR)

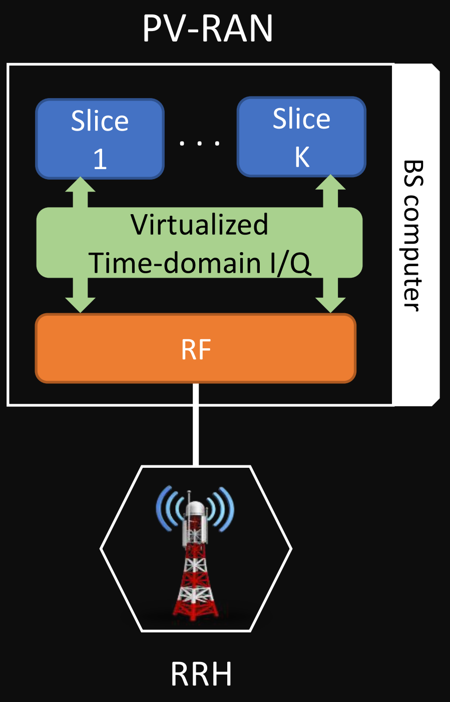
- Solves the following key challenges:

  - ✓ Share the same SDR physical resources across multiple slices
    by virtualizing the physical wireless resources (i.e.: time-domain baseband traffic also known as I/Q samples)

  - ✓ Allow multiple concurrent PHY & MAC layers to simultaneously access the radio device
    by decoupling RF I/O from the PHY layer

PV-RAN

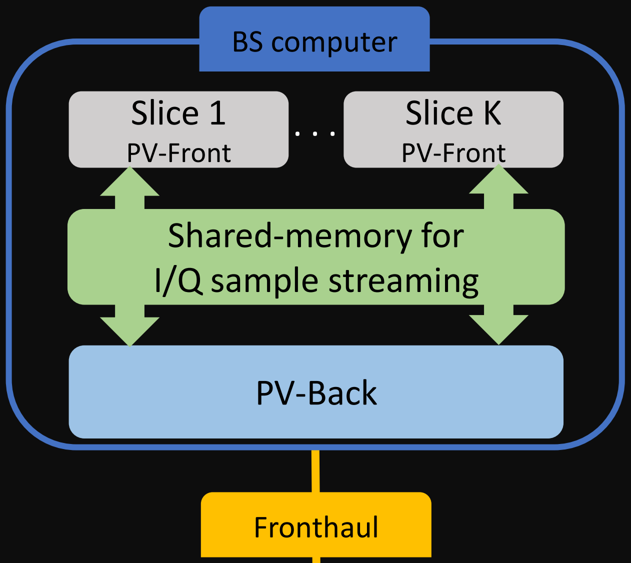
- Centralized RF
- Virtualized baseband traffic
- One radio device can serve several OpenAirInterface slices with different PHY and MAC layers
- Generic service for sharing physical RF resources
- Orthogonal to the 3GPP functional split
- Virtualization layer can be integrated with O-RAN



Physical Wireless Resource Virtualization

Challenge #1: Physical Wireless Resource Virtualization

Base Station Computer (Dell 3630)



BS computer

Slice 1 PV-Front

. . .

Slice K PV-Front

Shared-memory for I/Q sample streaming

PV-Back

Fronthaul

Software-Defined Radio (USRP x310)

- BS computer uses Xen hypervisor
- Slices are paravirtualized instances of OpenAirInterface (PV guests)
- I/Q samples are transferred though shared-memory inter-domain communication channels
- Split-driver model (PV frontend and PV backend device drivers)
- PV-Back for I/Q sample streaming (RF I/O) between PHY layers and the SDR device

SDR concurrent access

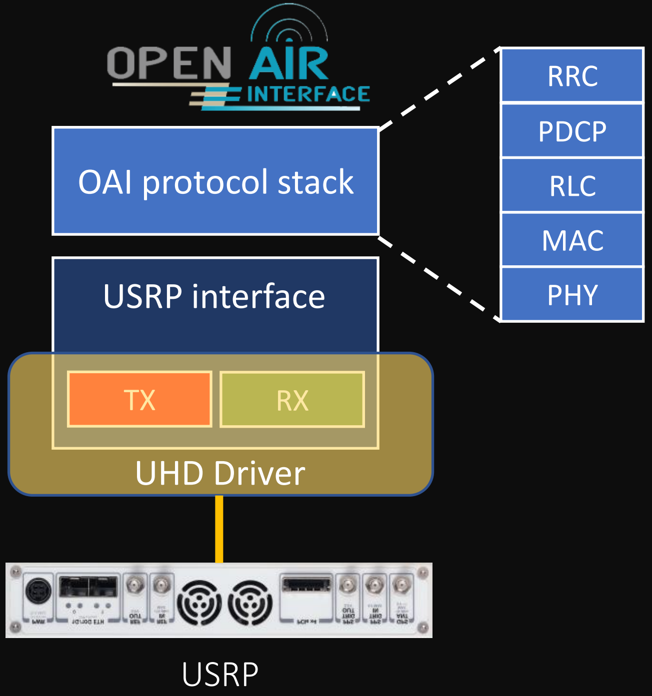
OAI USRP interface (usrp_lib.cpp) communicates with the USRP device through Universal Hardware Driver (UHD)

- USRP interface relies on UHD API (C++ bindings):
  RF front-end configuration (channel bandwidth, frequency, gains, …)
  RF I/O operations for I/Q sample streaming

BUT… Only one process can access UHD!

Solution: Delegate slices RF I/O to PV-Back exclusively

- In-band timestamping synchronization mechanism
- Add Xen virtual channel support to OAI’s USRP interface
- Transparent Virtualization via UHD API Remoting

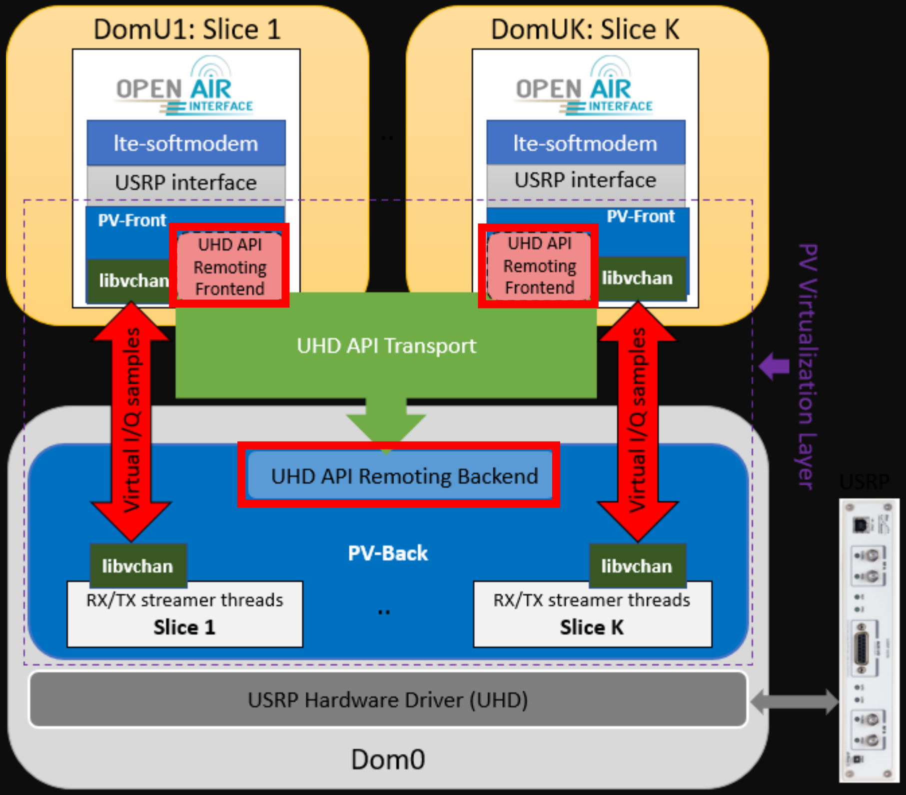


Transparent Virtualization

Transparent virtualization via UHD API Remoting

API Remoting uses run-time library interposition of UHD API function calls. It consists of:

- **UHD API Remoting Frontend**
  Dynamic library pre-loaded before USRP interface shared library (liboai_usrpdevif.so) code's execution. Substitutes UHD library calls with wrapper function stubs.
- **UHD API Transport**
  Virtual control channel that transmits control messages to the UHD API Remoting Backend in Dom0.
- **UHD API Remoting Backend**
  Handles call requests received over the UHD API Transport and executes them on the SDR device.



PV-RAN system configuration

Hardware

BS computer

Dell Tower Precision 3630
CPU: Intel Core i9-9900 @ 3.10 GHz
Memory: 64 GB
NIC: Intel x722-DA4

SDR

USRP x310
Daughterboard: 2x UBX-160
Fronthaul: OS2 fiber, Intel SFP+ transceiver
Antennas: VERT 2450 (for LTE), VERT900 (for TVWS)

For IDC
Require custom kernel compiled with gntalloc, gntdev, evtchn and xenbus

Software

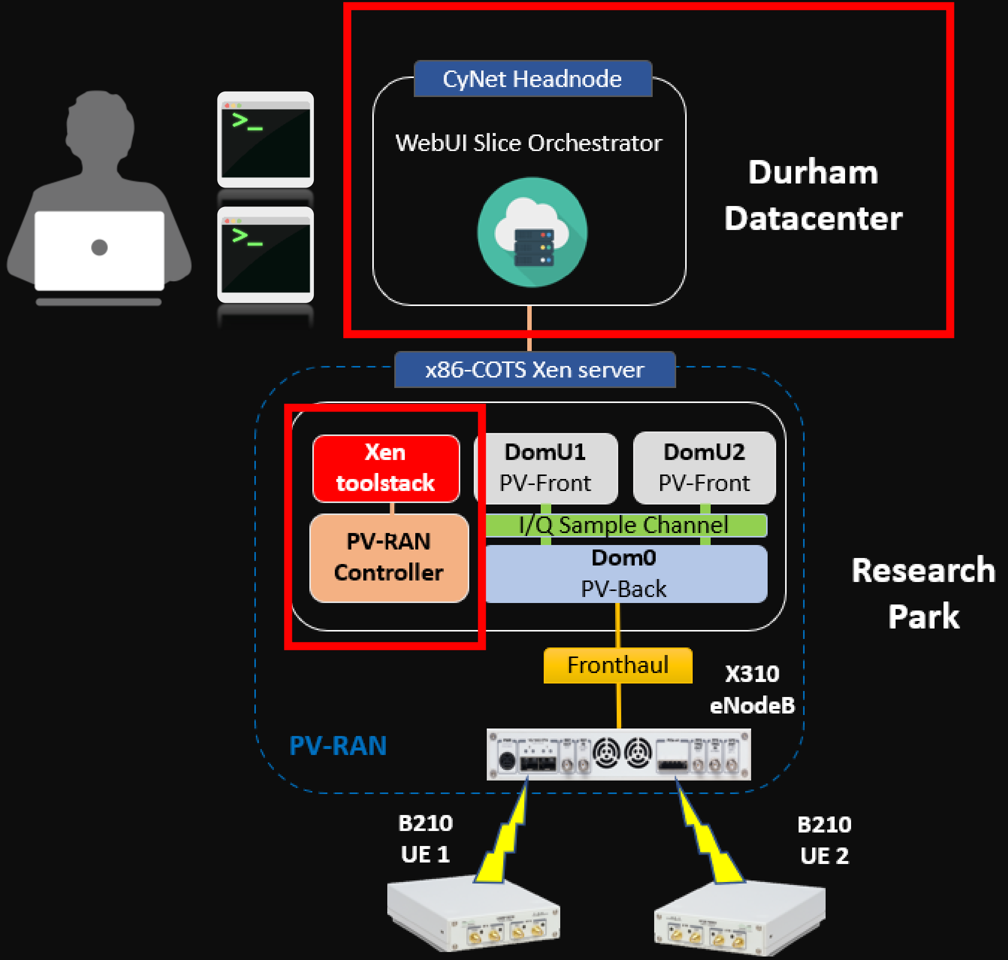
Xen Hypervisor

Version: Xen 4.9.2
Dom0 OS: Ubuntu 18.04.4 LTS
LVM volume group size: 475 GB
UHD version: v.3.15.0.0
Intel NIC driver: i40e

DomU PV guests

DomU OS: Ubuntu 16.04.4 LTS
vCPUs: 4
Memory: 6 GB
LVM logical volume size: > 18 GB
OAI version: v1.2.1
Xen shared libraries: libxenvchan, libxenevtchn, libxengnttab, libxentoolcore, libxentoollog

PV-RAN demo testbed



WebUI Slice Orchestrator

- Remote access to DomU guests
- Statistics visualization

PV-RAN platform

- PV-RAN Controller
- 1x X310 eNodeB (TVWS frequency bands)
- 2x DomU slices (one B210 UE per slice)

Conclusion

PV-RAN: First Paravirtualized RAN for physical wireless virtualization that enables whole-stack slicing where different PHY and MAC can be adopted for diverse communication services

- ➢ Compatible with other components from MOSAIC5G (FlexRIC, OAI-CN, etc.)
- ➢ Open-source software: https://gitlab.com/dnc-isu/cynet

PV-RAN is extensible

- ❖ Can be extended to network slicing strategies involving Time Division Multiplexing (TDM) and hybrid-TDM-FDM
- ❖ Extensible to other SDR platforms (HackRF, BladeRF)
- ❖ Can be ported to other open-source platforms such as KVM (using the shared-memory interface Nahanni)
- ❖ Can also be integrated with end-to-end network slice management systems such as those based on the O-RAN architecture and Open Networking Foundation