Using Deep Learning and Radio Virtualisation for Efficient Spectrum Sharing Among Coexisting Networks

Wei Liu[1(✉)], Joao F. Santos[2], Xianjun Jiao[1], Francisco Paisana[2], Luiz A. DaSilva[2], and Ingrid Moerman[1]

[1] IDLab University Ghent - IMEC, Technologiepark-Zwijnaarde 15, Ghent, Belgium
{wei.liu,xianjun.jiao,ingrid.moerman}@ugent.be
[2] Trinity College Dublin - CONNECT Centre, Dunlop Oriel House, Dublin, Ireland
{facocalj,paisanaf,dasilval}@tcd.ie

Abstract. This work leverages recent advances in machine learning for radio environment monitoring with context awareness, and uses the obtained information for creating radio slices that can optimally coexist with ongoing traffic in a given spectrum band. We instantiate radio slices as virtualised radios built on a software-defined radio platform. Then, we describe a proof-of-concept experiment that validates and demonstrates our proposed solution.

Keywords: Machine learning · Radio access technology classification
Radio virtualisation · Software-defined radio

1 Introduction

Increasingly, multiple Radio Access Technologies (RATs) must coexist in shared spectrum; this trend is only accelerating with the advent of 5G. In addition, the usage of flexible Radio Access Networks (RANs) with high reconfiguration capabilities presents new opportunities for interference coordination in shared spectrum scenarios. Radio Access Netowrks (RANs) may possess different degrees of compatibility, due to their unique waveforms, Medium Access Control (MAC) schemes, Radio Frequency (RF) parameters, or traffic requirements. In the context of network coexistence, it will be increasingly important for RANs to recognize the channels that exhibit transmission conditions that are favourable to their specific needs to optimize performance and, eventually, overall spectrum utilisation efficiency.

The project leading to this publication has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 732174 (ORCA project).

I. Moerman et al. (Eds.): CROWNCOM 2018, LNICST 261, pp. 165–174, 2019.
https://doi.org/10.1007/978-3-030-05490-8_16

Slicing is a concept used across Software-Defined Networks (SDNs), Network Function Virtualisation (NFV), and the 5G standard [1]. It involves sharing the underlying network infrastructure between multiple tenants, by assigning each tenant to a virtual network created with a portion of the network's resources, known as a network slice. This approach leads to an overall increase of network resource utilisation while offering greater flexibility, as each network slice can be used and managed independently [2,3]. Strategies similar to network slicing may be applied to optimize the utilisation of radio resources, which is referred to as radio slicing. More specifically, a radio slice in this work refers to a portion of the spectrum-time plane and the time share to utilize a radio interface assigned to a wireless link.

In general, network slicing requires some level of abstraction and virtualisation of physical devices and resources. Thus, network slicing and network virtualisation are closely tied [4]. The virtualisation of wired network components, e.g., links, routers and network interfaces, is a well-studied subject and is applied routinely, e.g., TCP/UDP ports and networking in virtual machines. In the wireless domain, virtualisation is still an active research topic [5]. In this paper, we combine deep learning, applied for spectrum awareness, and radio virtualisation for efficient spectrum sharing among coexisting networks.

It has been historically challenging to achieve such level of context awareness in scenarios where a wide variety of RATs coexist. However, with recent breakthroughs in the area of machine learning, it may be finally possible to bring these capabilities to existing RANs in a sufficiently flexible and cost-efficient manner. From the radio technology point of view, today's developers generally choose software-defined radio (SDR) solutions for their flexibility and fast development cycle. The adoption of SDR with high reconfiguration capabilities, together with breakthroughs in machine learning, brings the potential to use radio slicing as a means to optimize the coexistence among wireless networks and the utilisation of radio resources.

1.1 Spectrum Monitoring and Machine-Learning

The allocation of radio resources and configuration of RATs for supporting different types of services require the assessment and characterization of the underlying spectrum. Overall, spectrum monitoring can be used for enforcing spectrum policy, protecting incumbents in shared spectrum scenarios, supporting the coexistence between different RATs, and performing efficient radio resource management.

Contextual awareness is also key to radio and network slicing. The same RAT and its network slices may possess distinct protection requirements in terms of interference, and also employ different PHY parameters (e.g. subcarrier spacing), MAC schemes, and control structures.

Machine learning approaches have started to show promising results for spectrum monitoring and awareness, outperforming traditional solutions such as energy detection and expert feature detection [6]. These new approaches can collect features that go beyond simple waveform classification, e.g., frequency,

bandwidth, and frame duration. Furthermore, such machine learning-based solutions achieve greater generality, as a single neural network can be trained to recognize multiple types of RATs, and evolvability, as the given neural network can be retrained to recognize new RATs.

1.2 Radio Virtualisation

Current commercial devices use dedicated chipsets that implement a specific wireless standard. This approach works well if the device only needs to support a single RAT. However, devices often must support multiple RATs. As a result, more and more chipsets are installed in one device, which inevitably increases the device's form factor, power consumption, and complexity.

In addition to the need for supporting multiple standards, it is often required to have multiple radio interfaces of the same technology on one device. For example, some commercial Wi-Fi Access Points (APs) support simultaneous operation on different channels, and according to multiple variants of the IEEE 802.11 standard. Behind the scenes, the AP is switching channels periodically. Before it switches to the next channel, it broadcasts the "unavailable period" to the associated stations in the current channel. In this way, the AP can act as virtual APs on multiple channels. However, this approach is not entirely transparent to the upper layers, as the AP services are interrupted during the times of switching between different channels.

From the radio virtualisation aspect, in this paper we demonstrate that: (i) a physical radio can be instantiated into multiple virtual radios, (ii) the virtualisation process is application-transparent, meaning that the virtual radio offers uninterrupted services from the upper layer perspective in a way that is indistinguishable from multiple physical radios.

2 Proposed Solution

A block diagram of our proposed solution is shown in Fig. 1. On the left side, a classifier uses a deep Convolutional Neural Network (CNN) trained for classifying different RATs. The input of the classifier is time domain IQ samples, captured from a finite number of channels (denoted by N) with equal bandwidth. The use of time domain IQ samples instead of a spectogram provides amplitude and phase information to the classifier: this additional information makes the classifier more robust in the classification and feature extraction of signals [7]. Figure 2 shows screenshots of the graphical display of the classifier: it illustrates the classification of different types of RATs, where we use the bounding boxes for extracting features from the RATs and calculating signal statistics. The output of the classifier indicates: the kind of signal present in a channel, measured by the signal intensity and bandwidth; and the type of traffic the signal carries, measured by the burst length, average channel duty cycle, and minimum Inter-Frame Interval (IFI).

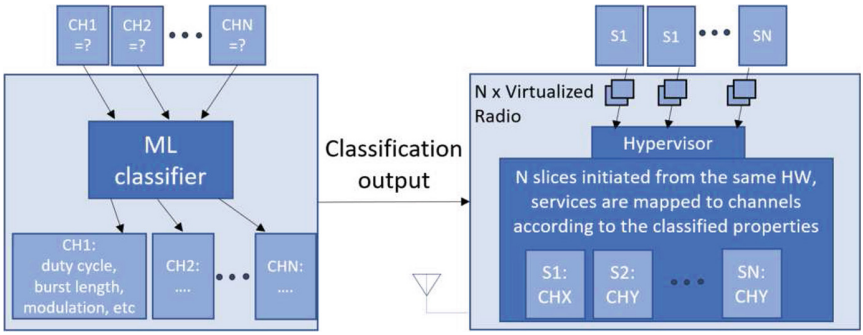

Fig. 1. A block diagram of the proposed solution: N virtual radios are instantiated according to service demands and context information provided by the machine learning (ML) classifier.

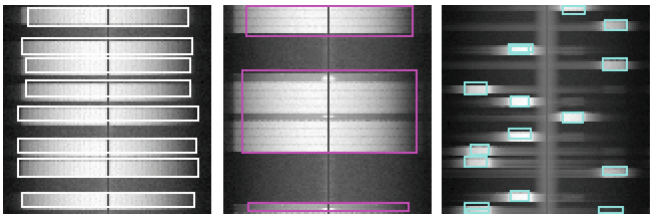

Fig. 2. Example of the RAT detection on three different channels over time. The different colours of the bounding boxes indicate different radio acccess technologies. (Color figure online)

The radio hypervisor handles N services from the host PC using a single SDR device. We identify data streams from different services by their TCP port. Our hypervisor collects the data from the N services, and then streams each of them to the SDR device with an appended "slice identifier". The slice identifier informs the SDR to send the data on a given channel and treat it with a given priority. Then, our hypervisor uses the output of the classifier to decide which channel will be used for a given service. This decision is based on the properties of the background traffic, described by the output of the classifier, and the demand of the traffic we want to serve.

For instance, a channel can appear highly occupied at first glance, but the classifier may identify that the background traffic on the channel is in fact composed of many short bursts. This kind of channel cannot sustain a service with high throughput requirements, but it can be suitable for a service with low throughput and low latency requirements, as there will be frequent transmission opportunities in between the bursts of background traffic. Moreover, the classifier also recommends the optimal coexistence settings for a slice, e.g., the transmission duration on a channel that can fit into the minimum IFI of the

background traffic. In this way, the classifier's output is used to facilitate the resource mapping and optimization process on each virtual radio.

Zooming in towards the SDR side, the virtualisation comes down to using a single radio front-end and transceiver chain on multiple channels. Behind the scenes, the transceiver chain runs N times faster than the speed required for a single channel operation. On the transmission side, the baseband IQ samples for each channel are first up-sampled, and then shifted to the desired frequency offset, and finally combined with the baseband samples from the other channels and streamed to the Digital to Analog Converter (DAC), as detailed in [8]. The reverse process occurs on the receiver side. The concept of radio virtualisation does not depend on the exact architecture of the transceiver or other processing modules: it could happen at CPU level [9] or at the hardware level on the FPGA [10].

Our goal is to instantiate radio interfaces according to the required services, finding the best combination between a set of channels and services while taking into account the context information (such as existing background traffic) and the demand. Though this work focuses on frequency slicing, similar strategies can be applied to other types of resources, such as space or time.

3 Proof of Concept

3.1 Experiment Setup

We use an experiment to validate our solution, and its setup is shown in Fig. 3. We consider two 20 MHz channels: on each channel, we use a USRP N210 for generating the background traffic, referred to as the traffic generator, and another USRP N210 for capturing samples of the given channel, referred to as a channel probe. We stream the samples from the channel probes to the 'spectrum monitor', which is a laptop running a trained machine learning model on its Graphical Processing Unit (GPU) for classifying the radio environment on both channels. The Machine Learning (ML) model classifies the RAT of the background traffic present on each channel and extracts their contextual information in real time. This information will be used by a Base Station to select its optimal settings, including channel of operation.

A simple downlink scenario consisting of one Base Station (BS) and two User Equipments (UEs) is displayed in Fig. 3. The optimal setting from the ML classifier is transferred to a host PC that controls the BS. Each BS/UE consists of a host PC with an SDR device, which is further composed by a Xilinx Zynq 7000 ZC706 evaluation board [11] and an Analog Devices FMCOMMS2 front-end [12], referred to as the Zynq SDR hereafter.

On channel 1, the background traffic comprises relatively short bursts (on average 2 ms) interleaved by comparable IFI, while on channel 2, the individual frame is relatively longer (10 ms), with on average 60 ms IFI. We used an Anritsu MS2781B spectrum analyser for measuring the characteristics of the background traffic, and the results are illustrated in Fig. 4.

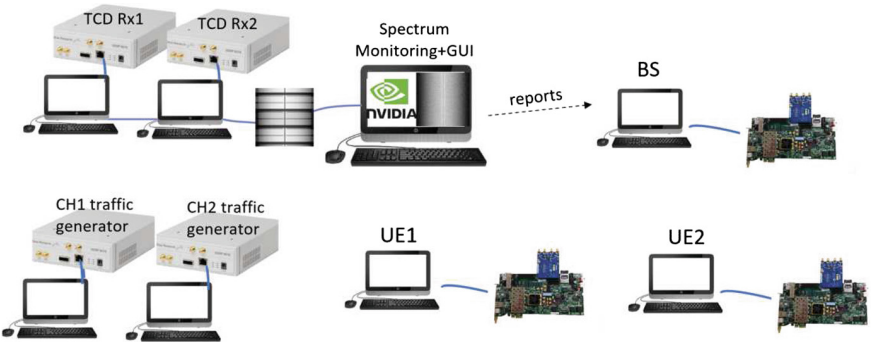

Fig. 3. The experimental setup: 2 N210 USRPs as background traffic generators, 2 USRP N210 as channel probes, 1 laptop as spectrum monitor, 1 BS which receives context information and serves the two UEs. Each BS/UE consists of a host and a Zynq SDR platform (Xilinx zc706+FMCOMMS2).

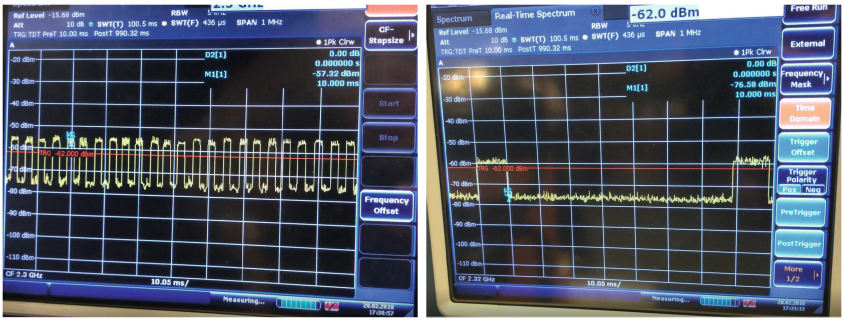

Fig. 4. The background traffic in the time domain: the traffic on channel 1, shown on the left side, has on average a 2 ms period and a duty cycle of roughly 50%; the channel 2 traffic, shown on the right side, has on average a 70 ms period and occupies the channel for about 13% of the time.

At the base station, the first question is which channel should be used to serve traffic for which UE. In this experiment, the traffic towards UE1 (referred to as T1) generates 5 frames at 100 bytes per second, and each frame must be delivered within 5 ms. This type of traffic is representative of traffic generated by a watchdog application in a factory environment, which reports regular "heartbeats" of a system, and can trigger critical safety procedures when needed. A large file transfer produces the traffic towards UE2 (referred to as T2). T2 only requires the link to sustain an average throughput in the order of Mbps. After combining the requirements of T1 and T2, and the context information regarding the background traffic, the BS makes its first decision: Channel 1 should be used to serve T1 for its more frequent transmission opportunities, and Channel 2 should be used to serve T2 for its overall lower occupation level.

We use a simple Listen Before Talk (LBT) module on the BS, which allows the BS to inject traffic only when a channel is idle. This process is shown on the left side of Fig. 5, where T2 shortly follows the background traffic on channel 2 with relatively lower signal strength.

The next question is how long the BS transmission can last on each channel, which is required to maximize the throughput for T2. The classifier collects different signal statistics, including the minimum IFI (1.35 ms), and provides them to the BS to optimize its transmission burst length. The application of this information results in a better usage efficiency for channel 2, as shown on the right side of Fig. 5.

Fig. 5. The traffic injected by the BS on channel 2 under default (left hand side) and optimal (right hand side) transmission burst length settings.

3.2 Measurements and Results

We measure the latency of T1 and the throughput of T2 to indicate the quality of the desired services. Details of these measurements are given below.

Throughput of T2 is measured by capturing frames at the Ethernet interface on the host of UE2. The Wireshark IO statistic graph is shown in Fig. 6. The average throughput for default and optimal slice configurations are 912.7 kbps and 1908.9 kbps, respectively. We observe that the throughput doubles by applying the optimal setting. The standard deviation of the throughput under both conditions is around 150 kbps, which illustrates that the stability of the throughput is unaffected.

Latency can be measured at different levels. The time delay between the moment when a frame is being transmitted on the wireless medium and the moment when the receiver decodes a complete frame is the latency at the physical layer. In general, it is subject to the frame size, the specific wireless technology, and it is also PHY implementation dependent. However, these are not the

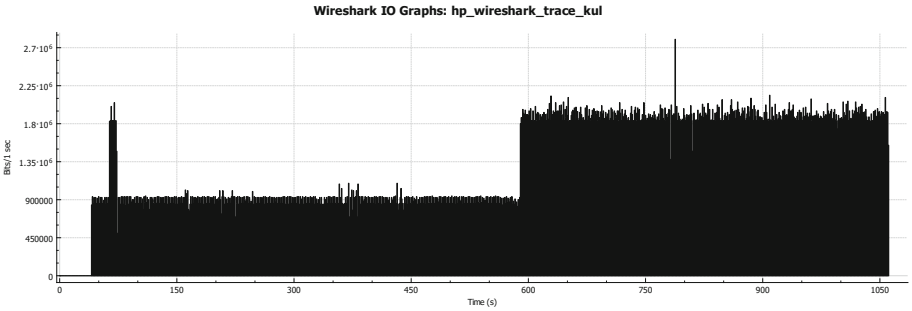

Fig. 6. Throughput of T2 shown by Wireshark IO statistics: an improvement is visible after applying the optimal setting recommended by the classifier.

focus of this work: we focus on the end-to-end delay, which includes the aforementioned physical layer delay, but additionally includes the time consumed by queuing in upper layer buffers, and processing delays at both transmitter and receiver sides in the SDR. For consistency, this measurement is conducted with fixed payload size, modulation and coding scheme.

The end-to-end latency of T1 is measured as follows: (i) Whenever the BS SDR receives a frame from the host, it triggers a TCP interrupt service routine (ISR), which toggles a General Purpose Input or Output (GPIO) pin within the same ISR. (ii) On the UE1 SDR side, whenever a frame pops out of the RX queue[1] before it is sent towards the host, the UE toggles a GPIO pin. (iii) Both the GPIO pins at the BS and UE are connected to a Saleae logic analyser. The logic analyser samples both pins at 6.25 Msps, ensuring the measurement precision at sub-microsecond level. A screenshot of the logic analyser's graphical user interface (GUI) is given in Fig. 7. The "sender" line shows the GPIO activity of the BS, while the "Receiver" line shows the GPIO activity of UE1. The latency is indicated by the distance between the two markers (i.e., A1, A2). For this particular case in the screenshot, it is 87.2 µs. The GUI also allows exporting the trace of the pin activity into a CSV file, which we later process using a simple Matlab script for generating the probability distribution of the latency measurements under various conditions.

Fig. 7. The interface of Saleae logical analyser.

[1] The physical layer receiver at the UE SDR decodes frames and inserts them into a queue, which is referred to as the RX queue.

The measurements are conducted under three conditions: (i) serving only T1 with no background traffic; (ii) serving T1 and T2 with no background traffic; (iii) serving T1 and T2 together with background traffic on both channels. The histograms of latency measurements under all conditions are given in Fig. 8. We observe that the latency for the first two conditions have similar distribution range, and they both have a high peak around 87 µs, which indicates that the latency of T1 is not negatively affected by serving T2. The virtual radio can hence operate in a way that is transparent to the user. The latency distribution for the 3rd case shows a high peak in the first interval, which corresponds to 87 µs—the best case scenario when the frame can be sent immediately without waiting for the background traffic, just as in the first two conditions. We also observe that probability is generally higher in $[0, 2000]$ µs range than the remaining part. This shows that the majority of the frames are sent after waiting for at most one complete frame of the background traffic, which lasts on average 2 ms. The worst case latency is around 3 ms: this can be explained by longer frames in the background traffic or missed detection of the transmission opportunities by the simple LBT module.

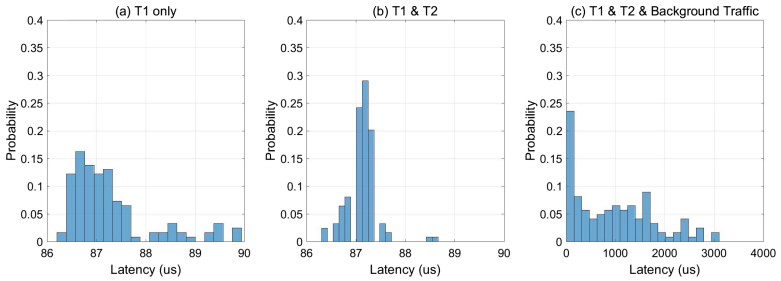

Fig. 8. The probability distribution of latency of T1 when: (a) serving only T1; (b) serving T1 and T2; (c) serving T1 and T2 with background traffic on both channels.

4 Conclusions

This work combines radio virtualisation for Software-Defined Radio platforms with deep learning technologies for optimizing spectrum utilisation, and the coexistence of wireless networks. We use a proof-of-concept experiment to showcase the combined application of: (i) radio virtualisation; and (ii) deep learning-based radio environment monitoring. Regarding radio virtualisation, it is demonstrated that a single radio device can be instantiated upon requests into multiple logical instances, each serving a different traffic flow with diverging requirements. Regarding radio environment monitoring, we showcase how decisions based on context awareness could improve coexistence and the quality of service experienced by each traffic flow.

References

1. 3rd Generation Partnership Project: 3GPP TR 28.801: study on management and orchestration of network slicing for next generation network. 3rd Generation Partnership Project, Technical report, May 2017
2. Rost, P., et al.: Network slicing to enable scalability and flexibility in 5G mobile networks. IEEE Commun. Mag. **55**(5), 72–79 (2017)
3. Khan, S.N., et al.: Virtualization of spectrum resources for 5G networks, In: 2017 European Conference on Networks and Communications (EuCNC), pp. 1–5 (2017)
4. van de Belt, J., et al.: Defining and surveying wireless link virtualization and wireless network virtualization. IEEE Commun. Surv. Tutor. **19**(3), 1603–1627 (2017)
5. Liang, C., et al.: Wireless network virtualization: a survey, some research issues and challenges. IEEE Commun. Surv. Tutor. **17**(1), 358–380 (2015)
6. Wunsch, F., et al.: DySPAN spectrum challenge: situational awareness and opportunistic spectrum access benchmarked. IEEE Trans. Cogn. Commun. Netw. **3**(3), 550–562 (2017)
7. Selim, A., et al.: Spectrum monitoring for radar bands using deep convolutional neural networks. IEEE Globecom (2017)
8. de Figueiredo, F.A.P., et al.: Radio hardware virtualization for software-defined wireless networks. Wirel. Pers. Commun. **100**(1), 113–126 (2018). https://doi.org/10.1007/s11277-018-5619-3
9. Mendes, J., et al.: Cellular access multi-tenancy through small cell virtualization and common RF front-end sharing. In: Workshop on Wireless Network Testbeds, Experimental evaluation and Characterization. ACM, pp. 35–42 (2017)
10. Jiao, X., Moerman, I., Liu, W., de Figueiredo, F.A.P.: Radio hardware virtualization for coping with dynamic heterogeneous wireless environments. In: Marques, P., Radwan, A., Mumtaz, S., Noguet, D., Rodriguez, J., Gundlach, M. (eds.) CrownCom 2017. LNICST, vol. 228, pp. 287–297. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76207-4_24
11. Zynq-7000 All Programmable SoC ZC706 evaluation kit, Xilinx (2015). https://www.xilinx.com/support/documentation/boards_and_kits/zc706/2015_4/ug961-zc706-GSG.pdf
12. AD-FMCOMMS2-EBZ User Guide, Analog Device (2018). https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms2-ebz