An ultra reliable low latency Cloud RAN implementation in GNU Radio for automated guided vehicles

Johannes Demel
Dept. of Communications Engineering
University of Bremen
Bremen, Germany
demel@ant.uni-bremen.de

Carsten Bockelmann
Dept. of Communications Engineering
University of Bremen
Bremen, Germany
bockelmann@ant.uni-bremen.de

Armin Dekorsy
Dept. of Communications Engineering
University of Bremen
Bremen, Germany
dekorsy@ant.uni-bremen.de

***Abstract*—We present our Over-the-Air Ultra Reliable Low Latency Communication system implementation in GNU Radio. This includes our Cloud Radio Access Network concept and the used Out-Of-Tree (OOT) modules for multicarrier synchronization, GFDM modulation, polar coding, and symbol mapping. We demonstrate how we use GNU Radio with Universal Software Radio Peripherals (USRPs) on our Automated Guided Vehicles (AGVs) and as part of our base station to realize a full transceiver. This includes illustrations and hands on descriptions how we integrate all components. Finally, we discuss our measurement results in this paper. To this end, the conducted Signal-to-Noise-Ratio (SNR) measurements over our testbed area as well as latency measurements validate the feasibility of our implementation for Ultra Reliable Low Latency Communication (URLLC) requirements.**

***Index Terms*—GNU Radio, polar codes, GFDM, wireless, SDR, open source**

I. Introduction

We present our Over-the-Air Ultra Reliable Low Latency Communication system implementation in GNU Radio [1]. The demonstrator is comprised of several open source GNU Radio OOT modules that we present in this work [2]–[8]. The considered technologies are polar codes for Forward Error Correction (FEC) [7], [9], fast, optimized, and standardized soft demappers with Bit-Interleaved Coded Mapping (BICM) [2], [10], [11], Generalized Frequency Division Multiplexing (GFDM) multicarrier modulation [3], [12], [13], and high throughput synchronization based on a refined version of Schmidl&Cox (Schmidl&Cox) [4], [14]. The goal of this demonstrator testbed is an evaluation of a URLLC system in a Cloud Radio Access Network (Cloud RAN) setup with distributed Access Points (APs) [15], [16]. The requirements include an end to end latency below $1\,\mathrm{ms}$ and resilience against burst errors [17].

The benefits of our Cloud RAN software implementation are flexibility, re-usability, efficiency and performance [18]–[20]. Further, benefits include more rapid development cycles and thus features may be deployed faster. Hardware development, even if not Application Specific Integrated Circuit (ASIC) focused, is inherently difficult and cumbersome. The author in [20] makes a case for Field Programmable Gate Array (FPGA) development in case requirements cannot be met otherwise but recommends to stick to software development. Thus, we stick to a software implementation for our testbed demonstrator to enable fast technology validation. Often, software development for Radio Access Network (RAN) is carried out with several programming languages including C and C++ for performance critical code as well as Python for support functionality and testing and thus, we focus on these as well [21].

We start with a discussion of our testbed concept, the software environment, and the used and implemented software components and their usage. Afterwards, we discuss the available hardware. Finally, we present our testbed at the NEOS building, Bremen, along with our measurement results. The capabilities of interest are the achievable latencies and reliability investigations for our Cloud RAN setup. The Cloud RAN setup includes distributed antennas and APs to improve reliability by providing spatial diversity, and thus redundancy. Eventually, we demonstrate that our demonstrator supports the communication testbed AGV requirements by using it.

We contribute an open-source software Over-the-Air (OTA) communication system implementation built upon the GNU Radio with multiple OOT modules [2]–[8]. Moreover, we use this system to conduct SNR and Round Trip Time (RTT) measurements in our testbed area. GNU Radio provides the integration framework for our modules, networking capabilities, and enables seamless access to the required capabilities that USRPs provide. These capabilities include, continuous high rate sample streams, exact receive and transmit timing information, and high precision synchronization across multiple devices. With distributed APs to improve reliability through spatial diversity, the chosen approach shows a significant improvement to counter fading induced communication outages. Thus, distributed APs provide an important option to minimize burst errors in URLLC communication

This work was partly funded by the German ministry of education and research (BMBF) under grant 16KIS1012 (IRLG).

systems. SNR measurements over our testbed area confirm that distributed APs complement each other to boost reliability. Moreover, our latency measurements confirm that low latency communication systems are achievable and fully implementable in software. The GNU Radio OTA software implementation is able to provide lower latency than current Long Term Evolution (LTE) and 5th Generation New Radio (5G NR) systems [22].

II. Testbed concept

The testbed is built with the Software-Defined Radio (SDR) concept in mind. Thus, the testbed includes numerous parts that are implemented in software, as well as the SDR hardware to transmit and receive. Here we present an overview as illustrated in Fig. 1. All aspects of our testbed are design with this concept in mind.

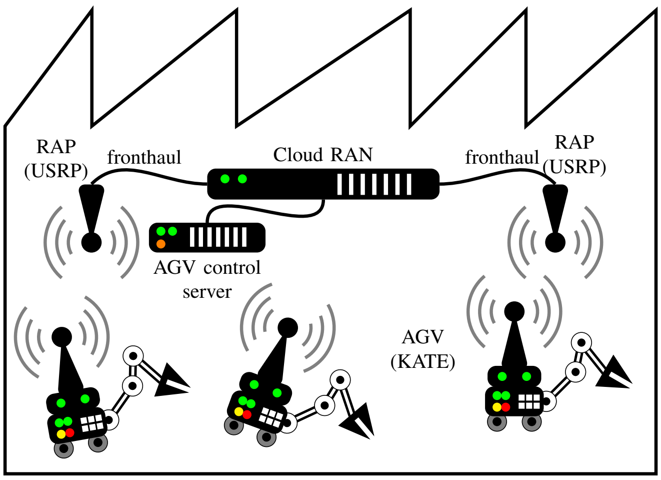

Fig. 1. Our testbed concept with its core components.

We consider a factory hall where multiple AGVs operate, specifically Götting KG Kinetic Automat for Transport Enhancements (Götting KATEs) [23]. All Götting KATEs are controlled by a central AGV control server that requires deterministic communication with small packets and a 100 ms periodicity. The default IEEE 802.11 (Wi-Fi) communication system is replaced with our custom SDR solution to accommodate 20 ms periodicity which is the lowest periodicity supported by the application. Our Cloud RAN system with distributed APs is designed and configured such that it provides higher reliability and lower latency.

III. Testbed configuration

We briefly discuss our URLLC testbed configuration. Since the presented software implementation is very flexible, it may be adopted for widely different scenarios. However, we intend to use it in an URLLC context and hence, we present our testbed configuration that is suited for the URLLC scenario.

Our implementation runs on a host with Ubuntu 20.04 with a Linux 5.x generic kernel. The configuration follows the Ettus knowledge base guidelines [24] with increased network buffer size, Central Processing Unit (CPU) performance governor adjustments, and enabled real-time scheduling for USRP Hardware Driver (UHD) and GNU Radio flowgraphs. In order to plug our GNU Radio flowgraphs via GNU Radio *Socket PDU* blocks into the host network stack, we employ Linux Foo over UDP (FOU) tunneling [25], [26]. The Foo over UDP (FOU) interface is configured with *network-tools* to accept 84 B packets which comprise a 20 B Internet Protocol (IP) header and potentially an 8 B User Datagram Protocol (UDP) header [26]–[28]. Moreover, this configuration allows for unfragmented Internet Control Message Protocol (ICMP) `ping` transmission [29]. The custom Medium-Access-Control (MAC) adds a 15 B header and thus, the PHYsical layer (PHY) conveys 99 B, or 792 bit, packets.

The PHY produces 1800 bit polar coded frames that are mapped to Quadrature Phase Shift Keying (QPSK) symbols and subsequently GFDM modulated with $K_\mathrm{t} = 15$ timeslots, $K_\mathrm{s} = 64$ subcarriers, and $K_\mathrm{on} = 60$ active subcarriers [12]. Together with a $N_\mathrm{CP} = 16$ Cyclic Prefix (CP) and a $N_\mathrm{CS} = 8$ Cyclic Suffix (CS) and an additional $K_\mathrm{t} = 2$ GFDM preamble, a complete frame consists of 1136 S. The configured sample rate is $30.72\,\mathrm{MS\,s^{-1}}$ with two transmit and two receive antennas. Thus, we use frames with 36.98 µs on air duration, or burst duration and occupy a bandwidth of 29.28 MHz. The transmit and receive antennas correspond to two distributed USRPs with a 1×1 configuration each. Finally, we organize the uplink and downlink with Time-Division-Duplex (TDD) and multiple access with Time-Division-Multiplex (TDM). Here, every User Equipment (UE) and our RAN may transmit every T_cycle, i.e. every user is assigned a slot in every cycle.

IV. Software implementation

In this section, we discuss GNU Radio and the UHD along with important used software libraries that comprise our software environment [1], [30]. Our demonstrator flowgraph in Fig. 2 is comprised of several GNU Radio OOT modules [2]–[8]. GNU Radio, UHD, and our OOT modules are available under the terms of the GNU General Public License v3.0 or later (GPLv3+).

This demonstrator is implemented in GNU Radio because GNU Radio offers a modular, multi-threaded framework for SDR applications while we can focus on the implementation of our algorithms. Furthermore, it offers an extensive set of blocks that enhance its capabilities, e.g. UHD and thus USRP integration. Our implemented OOT modules rely heavily on Vector-Optimized Library of Kernels (VOLK) and Fastest Fourier Transform in The West (FFTW), as does GNU Radio [31], [32]. In the following, we introduce our OOT modules in more detail.

A. *gr-tacmac*

The OOT module *gr-tacmac* serves as an umbrella for our demonstrator [8]. The GNU Radio flowgraph in Fig. 2 uses standard GNU Radio UHD blocks to connect our software implementation to USRPs while the *UDP interface* block uses GNU Radio *Socket PDU* blocks as well as our custom MAC

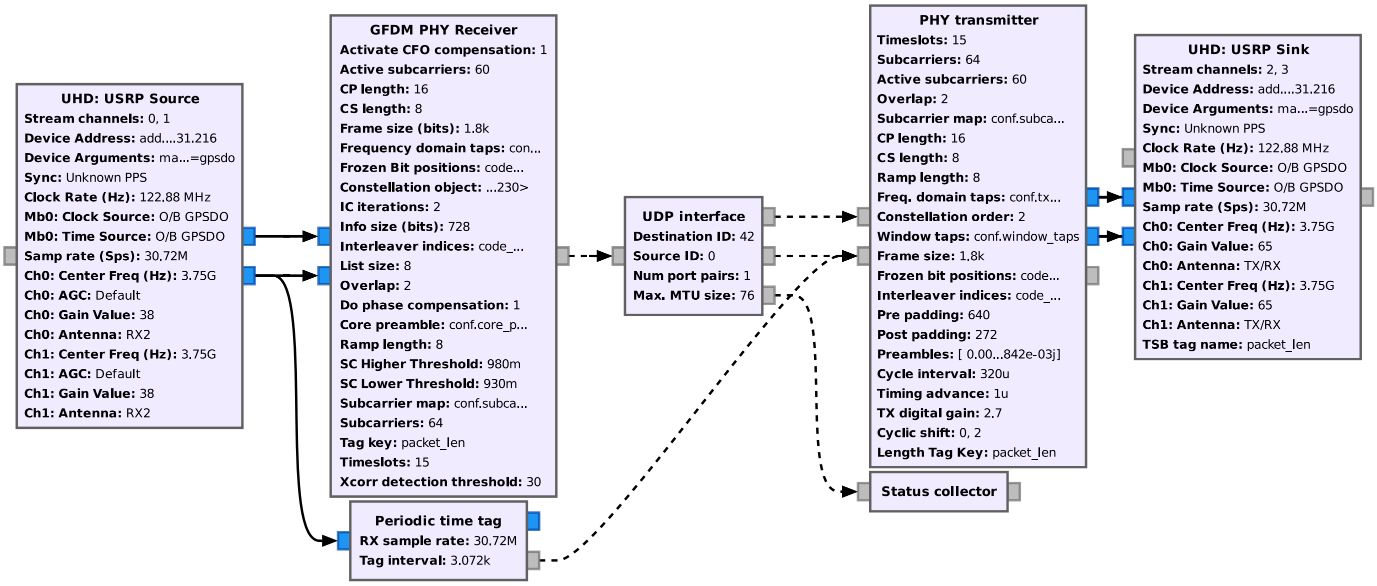

Fig. 2. Top-level GNU Radio flowgraph with UHD sink and source for $N_{\text{ant}} = 2$.

header block implementation to integrate into the Linux host network system. In order to efficiently integrate our system into the Linux host network stack, we employ *network-tools* to configure hosts [26]. Moreover, *gr-tacmac* provides the hierarchical flowgraphs that constitute the final flowgraph in Fig. 2. The hierarchical transmitter and receiver blocks in Fig. 2 encapsulate the transmitter and receiver blocks that rely on our subsequently discussed modules. The *Periodic time tag* block provides timing information to propagate a common time basis between the USRPs via UHD and the transmitter. Thus, it enables timed transmissions with tight latencies. Finally, the *Status collector* block collects metadata, e.g. SNR estimates, from the flowgraph which may be forwarded to an *elasticsearch* database. We use an *elasticsearch* database to permanently store metadata and *Kibana* for visualization.

B. *XFDMSync*

The *XFDMSync* module implements an improved Schmidl&Cox preamble approach [4], [14], [33], [34]. A preamble with two identical parts, consisting of a Zadoff-Chu sequence, is used to detect the beginning of a frame, perform time and frequency synchronization and finally allow for channel estimation. The channel estimation includes per subcarrier taps as well as SNR estimation.

C. *gr-gfdm*

The GNU Radio *gr-gfdm* module provides GFDM modulation capabilities [3], [12], [13]. Beyond the GNU Radio flowgraph integration, the algorithmic GFDM implementation is also available via *pybind11* to enable integration into Python projects such as simulations [35], [36]. The module relies on *XFDMSync* to provide received frames and channel estimates to equalize frames during demodulation. The demodulated and resource demapped frames consist of complex symbols that are expected by the *gr-symbolmapping* module and blocks therein.

D. *gr-symbolmapping*

The *gr-symbolmapping* module provides symbol mapping, soft demapping, as well as hard decision for received complex Quadrature Amplitude Modulation (QAM) symbols [2]. The symbol mapper and demapper blocks provide optimized implementations for a wide range of standardized constellations such as QPSK, 16QAM, and 256QAM as defined in LTE, 5G NR, or Wi-Fi. Arbitrary constellations are possible as well but naturally without the same level of optimization. Furthermore, the module provides interleavers and de-interleavers to enable BICM [10]. This interleaver implementation was upstreamed into GNU Radio [1].

E. *polar-codes*

Our *polar-codes* implementation offers heavily optimized polar code encoders and decoders for highest throughput and lowest latency [7]. The polar code implementation is based on several works on polar codes [9], [37], [38]. In order to use this implementation in GNU Radio, we use the *FECAPI* in the separate *gr-polarwrap* module to integrate our implementation into GNU Radio flowgraphs [6].

F. *gr-latency*

Our *gr-latency* OOT module provides further features to measure latency in GNU Radio flowgraphs [5]. The module is implemented based on the ideas presented in prior publications [39], [40]. The idea is to tag specific samples with precise time tags in a flowgraph while they traverse multiple blocks and gather timing information in a subsequent block to measure flowgraph latencies.

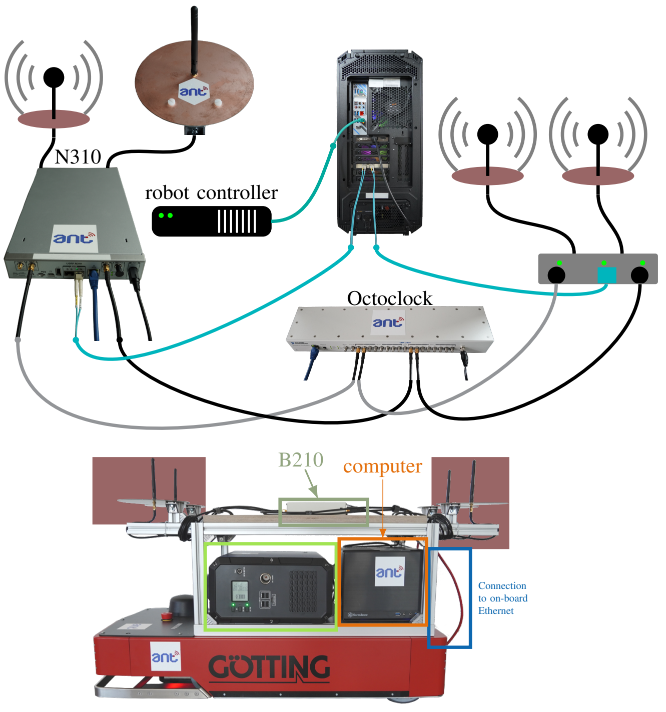

Fig. 3. Testbed hardware overview.

V. Hardware Platform

The demonstrator setup comprises AGVs, computers, USRPs and radio equipment as illustrated in Fig. 3. The Cloud RAN consists of a system with a AMD Ryzen Threadripper 3970X (TRX3970X) and 64 GB Random Access Memory (RAM) that computes samples from two distributed Ettus USRP N310s (N310s) with their corresponding *Pasternack PE51084* antennas with custom ground planes [41]. The distributed N310 are synchronized via an Ettus Octoclock-G (Octoclock-G) [42]. The AGVs, specifically Götting KATEs, carry a Ettus USRP B210 (B210) connected via Universal Serial Bus (USB) to a compact computer with an AMD Ryzen 9 5900X (Ryzen 5900X) CPU and 32 GB RAM [23], [43]. The presented system is measured with `/usr/bin/time` to measure memory usage. Across machines, our measurement tool reports a peak of 222 MB RAM usage even under heavy load in a client configuration.

VI. Testbed Measurements

The presented communication system is deployed in our testbed where we conduct experiments to obtain performance data. The testbed area in Fig. 4 covers approximately 359 m^2 with a stairwell in the middle, indicated in gray. While the testbed is located in an office building, we assume that the testbed represents an industrial-like environment, because the ceiling is covered by heating elements and the top, right, and bottom walls consist of heavily insulated windows. We use per AP SNR measurements to obtain a reliability indicator for each channel between an AP and an AGV. The SNR estimation algorithm uses the Schmidl&Cox preamble and

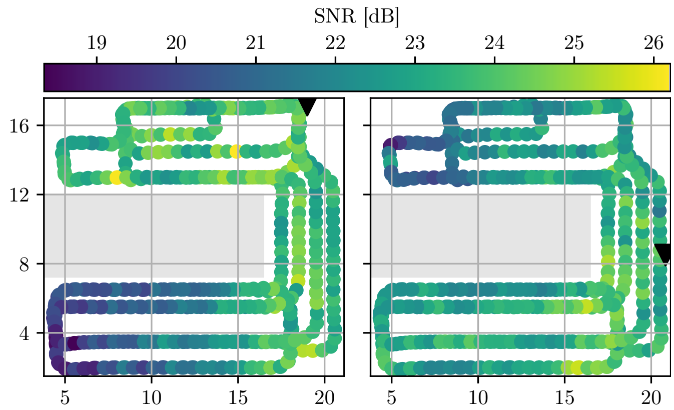

Fig. 4. Two distributed APs with a 1×1 antenna setup each. A black triangle [▼] indicates the AP position in the corresponding heatmap. Area axis ticks are in m.

is integrated into our *XFDMSync* module [44]. The SNR measurement campaign over the testbed area in Fig. 4 validates that our approach with distributed APs may improve reliability. The black triangle [▼] indicates the position of the AP corresponding to a SNR measurement heatmap. The upper area is particularly well served by AP 0 on the left in Fig. 4 while AP 1 improves the SNR in the lower area on the other side of the stairwell. This finding indicates that our Cloud RAN system improves the overall reliability in our testbed.

The RTT measurement results in Fig. 5 show minimum, average, and maximum values. Every data point represents multiple measurements. However, data aggregation is performed accordingly, i.e. the minimum line shows the minimum RTT in this aggregated data point. We observe that the RTT is always below 3.5 ms and on average around 2.8 ms. Also, the minimal measured RTT is just below 2.4 ms. While these values are slightly above the targeted 1 ms end-to-end latency, we are confident it is possible to close this gap in a feature work. It should be noted that the reported RTT values are application level measurements that include the network stack and thus, our application may rely on the same conditions. Finally, we compare the measured latency of our implementation with RTT measurement results of currently available LTE and 5G NR systems that do not yet provide URLLC capabilities [22]. These are the lowest available latency results to the best of our knowledge. In conclusion, our implementation provides lower latencies than currently available LTE and 5G NR systems. Thus, URLLC systems are of special interest for industrial applications as soon as they become available.

VII. Contribution

We presented a full Cloud RAN communication system demonstrator with distributed APs implemented in software with GNU Radio. Hence, we present our open-source software communication implementation modules [2]–[8]. Moreover, the paper includes a presentation of the concept, the software environment as well as important dependencies.

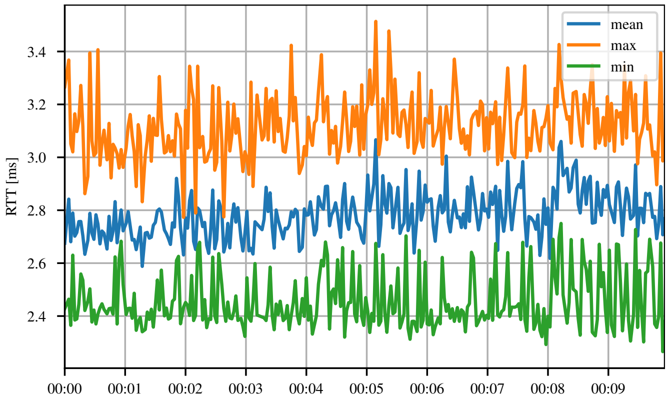

Fig. 5. RTT measurements during a test run.

Afterwards, we presented the available hardware and their integration. With distributed APs to improve reliability through spatial diversity, the chosen approach shows a significant improvement to counter fading induced communication outages. The conducted experiments validate the system performance and confirm that the GNU Radio OTA software implementation is able to provide lower latency than current LTE and 5G NR systems [22]. Here, we can conclude that the presented full software implementation is able to deliver low latency communication for future URLLC systems with periodic deterministic communication behavior.

References

[1] Long, Jeff, Morman, Josh, Abele, Jason, Aigner, Philipp, Alok, Aradhana, Anastasopoulos, Achilleas *et al.*, "GNU Radio," Sep. 2022. [Online]. Available: https://zenodo.org/record/7086439

[2] J. Demel, "GNU Radio Symbolmapping (gr-symbolmapping)," Dec. 2022. [Online]. Available: https://zenodo.org/record/7462854

[3] J. Demel, A. Rode, A. Kielstein, and S. Koslowski, "GNU Radio GFDM Modulator & Demodulator (gr-gfdm)," Dec. 2022. [Online]. Available: https://zenodo.org/record/7458328

[4] J. Demel and L. Göhrs, "XFDMSync," Dec. 2022. [Online]. Available: https://zenodo.org/record/7463105

[5] J. Demel, "gr-latency," Dec. 2022. [Online]. Available: https://zenodo.org/record/7464550

[6] ——, "gr-polarwrap," Dec. 2022. [Online]. Available: https://zenodo.org/record/6325779

[7] J. Demel and F. Lotze, "polar-codes," Dec. 2022. [Online]. Available: https://zenodo.org/record/6325748

[8] J. Demel, "gr-tacmac," Jan. 2023. [Online]. Available: https://zenodo.org/record/7525407

[9] E. Arıkan, "Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009. [Online]. Available: https://ieeexplore.ieee.org/document/5075875

[10] G. Caire, G. Taricco, and E. Biglieri, "Bit-Interleaved Coded Modulation," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 927–946, May 1998. [Online]. Available: https://ieeexplore.ieee.org/document/669123

[11] J. Mao, M. A. Abdullahi, P. Xiao, and A. Cao, "A low complexity 256QAM soft demapper for 5G mobile system," in *European Conference on Networks and Communications (EuCNC)*. Athens, Greece: IEEE, Jun. 2016, pp. 16–21. [Online]. Available: https://ieeexplore.ieee.org/document/7560996

[12] J. Demel, C. Bockelmann, and A. Dekorsy, "Evaluation of a Software Defined GFDM Implementation for Industry 4.0 Applications," in *International Conference on Industrial Technology (ICIT)*. Toronto, Canada: IEEE, Mar. 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7915548

[13] J. Demel, C. Bockelmann, A. Dekorsy, A. Rode, S. Koslowski, and F. K. Jondral, "An optimized GFDM software implementation for future Cloud-RAN and field tests," in *GNU Radio Conference (GRCon)*, vol. 7. San Diego, CA, USA: GNU Radio Foundation, Sep. 2017, p. 9. [Online]. Available: https://pubs.gnuradio.org/index.php/grcon/article/view/23/17

[14] A. B. Awoyesila, C. Kasparis, and B. G. Evans, "Improved preamble-aided timing estimation for OFDM systems," *IEEE Communications Letters*, vol. 12, no. 11, pp. 825 – 827, Nov. 2008.

[15] P. Rost, I. Berberana, A. Maeder, H. Paul, V. Suryaprakash, M. Valenti *et al.*, "Benefits and challenges of virtualization in 5G radio access networks," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 75–82, Dec. 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7355588

[16] 3GPP, "Service requirements for cyber-physical control applications in vertical domains; Stage 1," 3rd Generation Partnership Project, Valbonne, France, Technical Specification 22.104 V16.1.0, Mar. 2019. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3528

[17] J. Demel, C. Bockelmann, and A. Dekorsy, "Burst error analysis of scheduling algorithms for 5G NR URLLC periodic deterministic communication," in *Vehicular Technology Conference (VTC Spring)*, vol. 91. Antwerp, Belgium: IEEE, May 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9129493

[18] V. Marojevic, I. Gomez, P. L. Gilabert, G. Montoro, and A. Gelonch, "Resource management implications and strategies for SDR clouds," *Analog Integrated Circuits and Signal Processing*, vol. 73, no. 2, pp. 473–482, Nov. 2012. [Online]. Available: http://link.springer.com/10.1007/s10470-012-9963-z

[19] I. Gómez-Miguelez, "Radio and computing resource management in SDR clouds," Doctoral thesis, Universitat Politècnica de Catalunya, Catalunya, Spain, Dec. 2013. [Online]. Available: http://hdl.handle.net/10803/134882

[20] E. Grayver, *Implementing Software Defined Radio*. New York, NY, USA: Springer, 2013. [Online]. Available: http://link.springer.com/10.1007/978-1-4419-9332-8

[21] C. Walton and C. O'Dell, "Bridging Analog Land Mobile Radio to LTE Mission Critical Push-to-Talk Communications," *National Institute of Standards and Technology*, vol. NISTIR 8338, no. 8338, p. 33, Dec. 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8338.pdf

[22] G. Soós, D. Ficzere, P. Varga, and Z. Szalay, "Practical 5G KPI Measurement Results on a Non-Standalone Architecture," in *Network Operations and Management Symposium (NOMS)*. Budapest, Hungary: IEEE, Apr. 2020, pp. 1–5, iSSN: 2374-9709. [Online]. Available: https://ieeexplore.ieee.org/document/9110457

[23] Götting KG, "Kinetic Automat for Transport Enhancement, KATE," Dec. 2022. [Online]. Available: https://www.goetting-agv.com/kate

[24] Ettus Research, a National Instruments Brand, "Ettus Research USRP B210," Dec. 2022. [Online]. Available: https://www.ettus.com/all-products/ub210-kit/

[25] J. Corbet, "Foo over UDP," Oct. 2014. [Online]. Available: https://lwn.net/Articles/614348/

[26] N. Schwarzenberg, "network-tools," Jan. 2023. [Online]. Available: https://zenodo.org/record/7525279

[27] J. Postel, "Internet Protocol," Internet Engineering Task Force, Wilmington, DE, USA, Request for Comments 791, Sep. 1981. [Online]. Available: https://datatracker.ietf.org/doc/rfc791

[28] ——, "User Datagram Protocol," Internet Engineering Task Force, Wilmington, DE, USA, Request for Comments 768, Aug. 1980. [Online]. Available: https://datatracker.ietf.org/doc/rfc768

[29] ——, "Internet Control Message Protocol," Internet Engineering Task Force, Wilmington, DE, USA, Request for Comments 792, Sep. 1981. [Online]. Available: https://datatracker.ietf.org/doc/rfc792

[30] Ettus Research, a National Instruments Brand, "USRP Hardware Driver," Austin, TX, USA, Dec. 2021, original-date: 2013-03-27T19:52:19Z. [Online]. Available: https://github.com/EttusResearch/uhd

[31] J. Demel, M. Dickens, D. Anderson, B. Ashton, P. Balister, D. Behar *et al.*, "Vector-Optimized Library of Kernels (VOLK)," Feb. 2022. [Online]. Available: https://zenodo.org/record/6052858

[32] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, Feb. 2005. [Online]. Available: https://ieeexplore.ieee.org/document/1386650

[33] T. Schmidl and D. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613–1621, Dec. 1997. [Online]. Available: https://ieeexplore.ieee.org/document/650240

[34] I. S. Gaspar, L. L. Mendes, N. Michailow, and G. Fettweis, "A synchronization technique for generalized frequency division multiplexing," *EURASIP Journal on Advances in Signal Processing*, vol. 67, no. 1, pp. 1687–6180, May 2014. [Online]. Available: https://asp-eurasipjournals.springeropen.com/articles/10.1186/1687-6180-2014-67

[35] Python Software Foundation, "Python," Mar. 2021. [Online]. Available: https://www.python.org

[36] W. Jakob, J. Rhinelander, H. Schreiner, D. Moldovan, I. Smirnov, R. W. Grosse-Kunstleve *et al.*, "pybind11," Feb. 2022. [Online]. Available: https://zenodo.org/record/5951946

[37] P. Giard, "High-Speed Decoders for Polar Codes," PhD Thesis, McGill University, Montreal, Canada, Sep. 2016. [Online]. Available: https://escholarship.mcgill.ca/concern/theses/3r074x791

[38] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-Based Successive Cancellation List Decoding of Polar Codes," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5165–5179, Oct. 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7114328

[39] B. Bloessl, M. Müller, and M. Hollick, "Benchmarking and Profiling the GNU Radio Scheduler," in *GNU Radio Conference (GRCon)*, vol. 9. Huntsville, AL, USA: GNU Radio Foundation, Sep. 2019, p. 8. [Online]. Available: https://pubs.gnuradio.org/index.php/grcon/article/view/64

[40] B. Bloessl, "gr-sched repository," Nov. 2021. [Online]. Available: https://github.com/bastibl/gr-sched

[41] Ettus Research, a National Instruments Brand, "Ettus Research USRP N310," Dec. 2022. [Online]. Available: https://www.ettus.com/all-products/usrp-n310/

[42] ——, "Ettus Research OctoClock-G CDA-2990," Dec. 2022. [Online]. Available: https://www.ettus.com/all-products/octoclock-g/

[43] ——, "Ettus Research," Dec. 2022. [Online]. Available: https://www.ettus.com/

[44] M. Zivkovic and R. Mathar, "Preamble-Based SNR Estimation in Frequency Selective Channels for Wireless OFDM Systems," in *Vehicular Technology Conference (VTC Spring)*, vol. 69. Barcelona, Spain: IEEE, Apr. 2009, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/5073813