APP and PHY in Harmony: A Framework Enabling Flexible Physical Layer Processing to Address Application Requirements

Matthias Schulz*, Denny Stohr†, Stefan Wilk†, Benedikt Rudolph*, Wolfgang Effelsberg† and Matthias Hollick*
*Secure Mobile Networking Lab, TU Darmstadt, Germany, Email:{mschulz, brudolph, mhollick}@seemoo.tu-darmstadt.de
†Distributed Multimedia Systems, TU Darmstadt, Germany, Email:{dstohr, swilk, effelsberg}@cs.tu-darmstadt.de

***Abstract*—Mobile data traffic, particularly mobile video, grows at an unprecedented pace. Despite recent advances at the physical layer, today's wireless network infrastructure cannot keep up with this growth. This is partially due to the missing flexibility to adapt the physical layer continuously to best support both application level as well as network requirements. In this paper we show how to harness the flexibility of advanced physical layers in practice. We designed and implemented a research platform that provides a flexible application-centric physical layer for Android smartphones using software-defined radios (SDRs) as radio interfaces. Our solution allows applications to define flows and apply per-flow settings that are mapped into distinct physical layer settings. As a proof-of-concept and for testbed evaluation, we implemented our system together with a mobile video streaming application. The latter uses a Motion-JPEG based lightweight scalable video codec (SVC) to generate incremental data flows. We show that our system maximizes video quality at the receiver's side, while keeping the energy consumption at the transmitter at a minimum. Our solution demonstrates that jointly optimizing network traffic and application quality is feasible in practice using a flexible physical layer processing approach.**

I. Introduction

Today's Internet traffic is being shaped by the rise of digital video transmission. This growth is forecasted to be accelerated in mobile networks: video data is expected to grow 14-fold between 2013 and 2018—outpacing the general data traffic growth—and amounting to close to 70 % of all mobile data traffic in 2018 [6]. This growth cannot be handled by advances in wireless network technology such as Long-Term Evolution (LTE) or LTE-Advanced alone. Especially in crowded regions with a high density of wireless devices, the gap between desired and available bandwidth is huge and it tends to increase. Reasons are for instance that wireless resources, such as frequency spectrum, are limited, while the number of devices, users and applications continues to grow.

While LTE allows to differentiate between different traffic classes such as conversational voice and video, real-time gaming, video (buffered streaming), etc., these settings are hardly accessible by mobile applications. Moreover, state-of-the-art video codecs, especially SVCs, require more than simple per application parameterization of network, data link and physical layer settings to offer sufficient flexibility. Other wireless technologies, such as Wi-Fi, offer less and incompatible quality of service (QoS) control features. As a result, video streaming applications, such as cloud-rendered games or video conferencing systems, are using whatever best-effort service the network offers. We argue that this one-size-fits-all approach is ill suited for mobile (video) applications. We propose to design future network protocols such that they acknowledge the special requirements of these applications and capitalize on the flexibility offered by modern physical layer technologies, thus jointly optimizing network traffic and application quality.

Flexible research platforms to practically study cross-layer optimization in mobile wireless networks are scarce. On the one hand, building on integrated circuit designs for existing wireless technologies such as Wi-Fi, Bluetooth or LTE gives little room to researchers to explore physical layer settings outside the envelopes set by standards. This limits the design space for new wireless systems that allow to adapt the physical layer to the needs of an application. On the other hand, utilizing flexible SDR platforms complicates research with 'real' applications that exist in the smartphone ecosystems. SDR-based solutions often operate on synthetic traffic assumptions, which might only poorly match the constantly changing traffic requirements of applications deployed in the wild. To the best of our knowledge, there is currently no research and development platform combining the flexibility of SDRs and mobile application platforms with all their applications. In this work, we close this gap.

Our target application platform is Android, which is the mobile operating system for smartphones, tablets and wearables with the most widespread deployment [2], [8]. Its open source nature allows for extensions even to core operating system features. The hardware support as well as the software ecosystem of Android allows the recording, streaming and playback of high definition video, amongst other applications. In this work, we connect Android smartphones with SDRs—specifically the Wireless Open-Access Research Platform (WARP) [1]—to overcome the limitations of the wireless interfaces offered by off-the-shelf devices. While our solution requires to tether WARP nodes to Android smartphones, our implementation enables basic mobility of the phone by feeding back the wireless signals to an antenna system attached to the phone. As a result, we obtain a research platform that provides us with access to all Android features in combination with a highly flexible physical layer. Our proof-of-concept implementation further consists of an Android SVC-based streaming system supporting layered video.

This work contains the following contributions:

- We designed and implemented a real-time capable Android to WARP interface to allow mobile devices direct access to physical layer parameters.

978-1-4799-5804-7/15/$31.00

- We designed and implemented a flow-based packet processor to enable applications to change physical layer communication mechanisms on a per flow basis.
- We evaluated how well our system performs in realistic video streaming scenarios based on our lightweight Motion-JPEG-based SVC.

In Section II, we describe our system and detail our design decisions. In Section III, we introduce our lightweight SVC. In Section IV, we discuss implementation details of our system and describe the extensive performance evaluation using our proof-of-concept application. This is followed by a discussion of the results and future work in Section V, related work in Section VI as well as the conclusion in Section VII.

II. System Overview

In this section, we present a detailed overview on our system and the important design decisions for its implementation.

Regarding hardware, our system consists of two main components: Android smartphones and WARP software-defined radios. The smartphones represent mobile end-devices running applications. The WARPs offer flexible physical layer implementations that run in real-time on field programmable gate arrays (FPGAs). In our system, we bring both devices together to allow applications the setup of physical transmission requirements on a flow basis. This level of flexibility is not provided by off-the-shelf wireless chipsets that implement standard-compliant communication mechanisms, leaving little room for research outside the bounds of the standards. A flexible physical layer implementation can be controllable by applications that intend to optimize their data transmissions. However, full control is infeasible when multiple applications require access to the physical layer. Additionally, each application would need separate optimization strategies for different physical layers. To abstract from the details of the physical layer, we introduce an abstraction layer that takes control over setting physical parameters according to the applications' needs (see Figure 1).

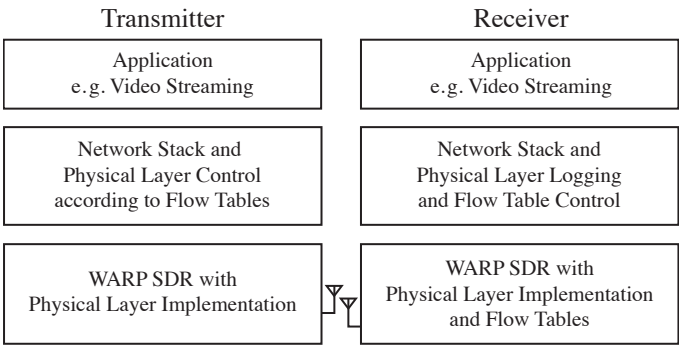


Fig. 1. Abstract System Overview

To handle different connections, we take advantage of the concept of flows. Flows can, for example, be differentiated according to TCP or UDP ports. Applications can then define abstract properties per flow such as robust packet delivery or high throughput. Our intermediary service translates these requirements to physical layer properties such as modulation scheme, transmit power or transmission priority. This way, we can handle multiple applications and abstract from lower layer details. In the following, we describe our system in more detail.

The proposed system consists of four components: (1) an SVC-based video streaming application as a practical example application, (2) the *WARP "VPN" Service* that allows to create tunneling devices (TUN) under Android to tap and inject IP packets without root privileges, (3) an Ethernet connection to the WARP to tunnel packets to the lower layers, and (4) the physical and data link layer implementations in the WARP. In the following, we describe the four components in detail.

Motivating example for our application-centric physical layer is the increasing demand for reliable, network-efficient video streaming. Due to requirements for flexibility and opportunistic delivery coming from the heterogeneity of video formats, playback devices and network throughput variations, we focus on scalable video coding. Compared to other data streams, video streams do not rely on error free transmission and can cope with a certain number of bit errors and packet loss. The latter might lead to degradation of the image quality, but it still meets real-time requirements. SVC additionally offers to encode video streams into multiple quality layers. The base layer is always required to playback the video. Higher layers, that lead to better image quality, resolution or frame rate, are optional. If the physical layer can differentiate, which data streams are more important for the application than others, it can allocate the appropriate amount of resources to fulfill the application's requirements. For example, by using more robust modulation schemes and different transmit powers, the system can save energy at the transmitter, while maximizing quality at the receivers.

In Figure 2, we present our complete system. At the left side is the transmitter that encodes a video stream and uses our system to transmit it to the receiver on the right, who decodes and plays back the video stream. The encoder splits the video stream into three different quality layers. On the right side you see how the videos would play back if only the base, the base and the first layer or the base and both extra layers were received. To transmit the data streams of each layer, the application could generate headers for all layers up to the application layer on its own and could tunnel the resulting frames over Ethernet to the WARP (at the bottom of the figure). Even though this gives an application full control over the network stack and allows frame-wise settings for the physical transmissions, it also complicates the access control to the physical layer, especially if multiple services need access to it at the same time. Also system wide services such as name and address resolution would be application depended.

To avoid these complications, we decided to use the existing Linux kernel to handle transport and network layers. This gives all IP-based applications access to our implementation and it allows to use existing transport layer protocol implementations such as UDP and TCP. To get access to IP packets on Android devices without root privileges, we instantiate a virtual private network (VPN) service to get a file descriptor on a TUN device. This VPN setup has the side effect that we can define IP subnetworks that are handled by our application-centric physical layer. All other packets are handled by the existing network interfaces.

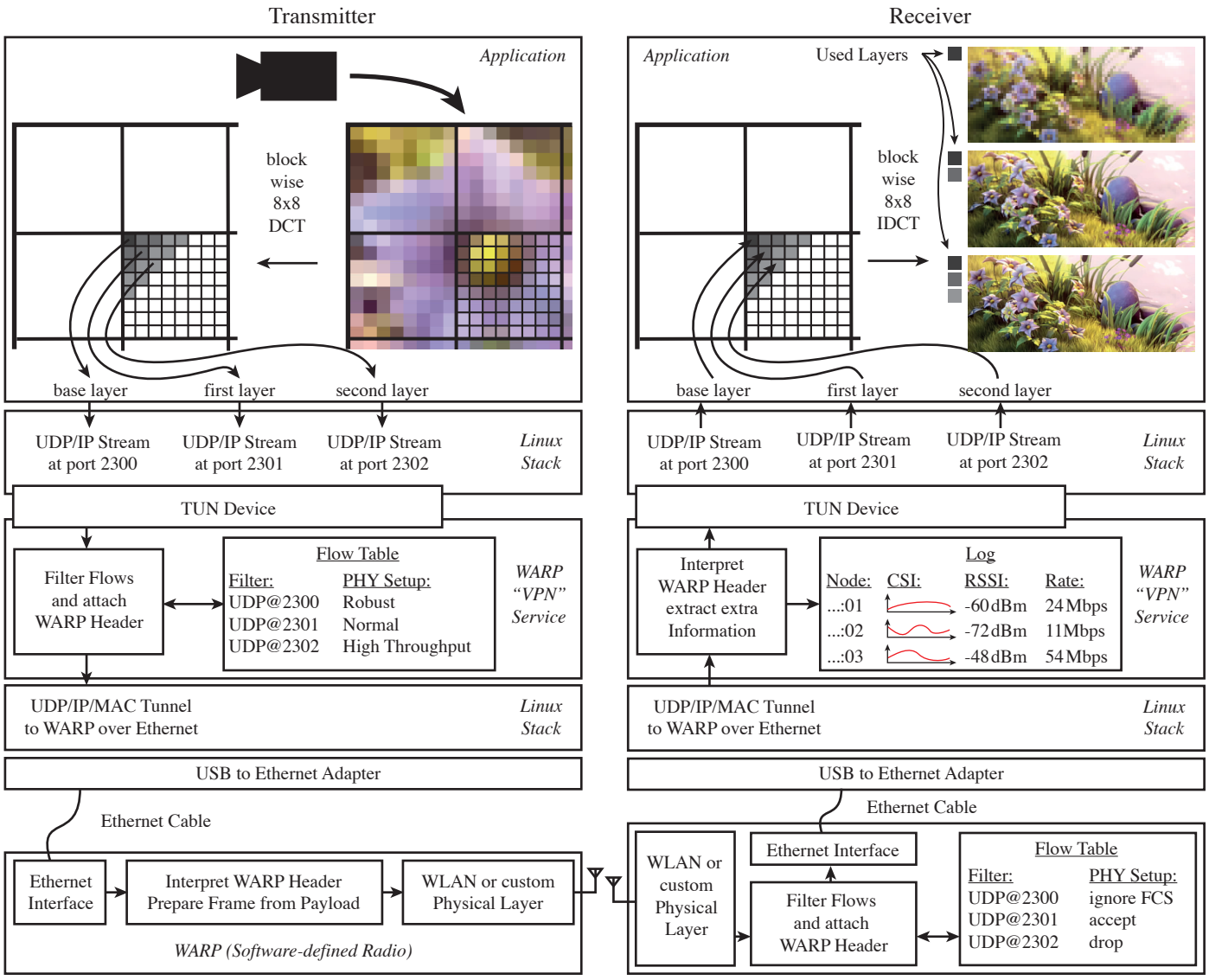


Fig. 2. System overview that shows the components of our transmitter and receiver. At the top, we illustrate our example application—a scalable video codec. It uses regular UDP over IP communication offered by the Linux kernel. Below is the *WARP "VPN" Service* to handle IP packets and setup physical layer requirements according to application-controlled flow tables. At the bottom are the WARPs that we use as wireless interfaces for our smartphones.

On the transmitter side, this *WARP "VPN" Service* takes all IP packets coming from the TUN device and attaches physical and data link layer headers to them. To decide which physical layer features are required by an application, we use flow tables. These tables contain filters that match bits in packet headers and they contain physical layer settings that should be passed to the WARP. In our example the different SVC layers are transmitted using different UDP ports. Each of the three streams is considered a flow and can have unique requirements on the physical layer.

To interface with the WARP, we considered multiple interfaces. To avoid interference during our wireless experiments, we decided against wireless interfaces. Secure Digital Input Output (SDIO) is not available on all smartphones. Directly interfacing the WARP using USB On-The-Go is not possible due to the lack of a USB interface at the WARP. However, we can use USB-to-Ethernet adapters to connect to the WARP's 1 Gbps Ethernet ports. Over the Ethernet link, we tunnel our physical layer frames to the WARP for transmission.

To gain maximum flexibility at the receivers and to be able to implement non-standard wireless communication schemes or extensions to existing ones such as 802.11, we also use WARPs as receive interfaces. On the one hand, they are part of the flow handling and can filter packets that should be tunneled to the Android smartphone. On the other hand, they can attach measurements that are not available on unmodified off-the-shelf wireless chipsets, such as channel state information (CSI) measurements. After reception, selected frames are tunneled to the *WARP "VPN" Service*, that extracts the payload passes IP packets on to the Linux kernel. The additional measurement information gets logged and can be used to enhance future transmissions, for example, by feeding it back to the transmitter who can optimize its transmit filters on the physical layer.

III. Robust Scalable Video Transmission

As a use case for our application-centric physical layer, we choose video streaming. Videos have the advantage, that their information can be encoded and compressed in various qualities and decoders can cope with packet loss and bit errors, which lead to image quality degradation that users may tolerate in real-time applications and while being mobile. To meet the requirements of heterogeneous end-devices as well as varying network speeds, scalable video coding offers to encode video material into different layers. A low quality base layer is required by every user. To improve video playback users can request additional layers. This principle is often illustrated by the SVC cube shown in Figure 3.

Currently, SVC extensions for existing video codecs such as H.264/AVC and H.265 [19] exist in the form of H.264/SVC [18] and scalable high-efficient video coding (SHVC) [9]. However, those codecs have a high complexity, as they combine the idea of SVC with very efficient video compression. To give researchers a smooth entrance into the world of scalable video coding, we present a lightweight scalable video codec that is based on the idea of Motion JPEG, where each video frame is encoded separately using JPEG images. JPEG compression relies on the fact that high frequency components in a picture are less important for the human visual system than low frequency components. Our video codec uses this idea and only packs the lowest frequency components into the base layer, which is required by all receivers. Additional frequency components come with the first and second layer and are used to increase video quality. The result is illustrated in the upper right corner of Figure 2.

To get the frequency components, the video encoder applies the discrete cosine transform (DCT) to each 8×8 block in each video frame on each of the three YCbCr color components. The receiver of the layered video stream converts the frequency components back to the spatial domain using the inverse discrete cosine transform (IDCT). The layers are required in the order base, first and second layer. If parts of a layer are missing, e. g. due to frame loss on the physical layer, missing parts are replaced by parts from proceeding frames.

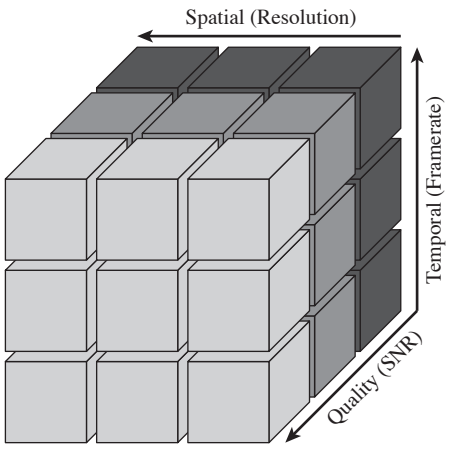


Fig. 3. The H.264/SVC cube model.

IV. Implementation and Evaluation

We implemented the system described above on Nexus 5 Android smartphones running stock firmware (version 4.4.2 (Android KitKat)) with a custom kernel to support the AX88179 based USB-to-Ethernet adapters. As SDRs we used WARP nodes running version 0.95 of the 802.11 reference design with an Ad-Hoc implementation from the developers' repository. We extended this design to handle our UDP-based tunneling protocol that can set physical layer parameters on a per-frame basis. To ease experimentation, we developed a simple control protocol to trigger video transmissions and setup flow tables over Ethernet.

A. Experimental Setup

To evaluate the feasibility of our solution, we set up a testbed in our office environment as illustrated in Figure 4. We used five smartphones that are equipped with WARP SDRs, as illustrated in Figure 5. To reduce interference by other Wi-Fi users, we performed experiments at night and selected channel 14 in the 2.4 GHz band with 20 MHz bandwidth.

During our experiments, we used Node 1 as the transmitter that generated a layered SVC video stream. We transmitted each of the three layers on a separate UDP port resulting in three different flows. Depending on the experiment, we varied the used modulation scheme, the amount of forward error correction (FEC) and the transmit power for each flow. The transmitter fragmented all video frames into 5 (base layer), 23 (first layer) or 41 (second layer) Wi-Fi frames (with 1348 byte payload) that we broadcasted in Wi-Fi Ad-Hoc mode without

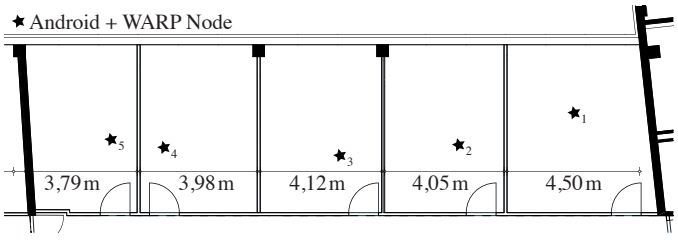


Fig. 4. Experimental setup in our office environment. All phones are hanging on movable hat stands in a height of roughly 170 cm. Positions look random as the devices are placed around the tables in our offices.

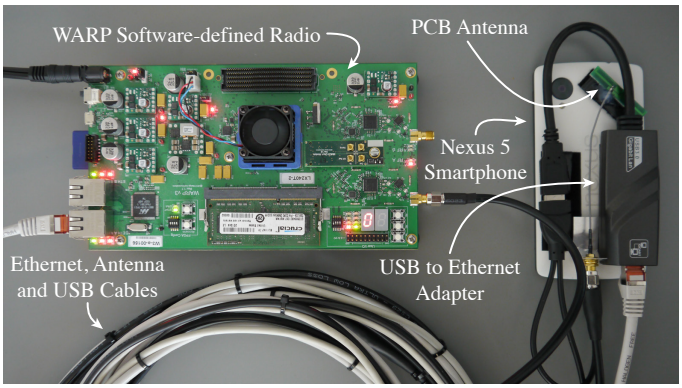


Fig. 5. To each Android node we attach a USB-to-Ethernet adapter with a 5 m long cable to the WARP and a power supply. Using a low-loss CS29 cable, we feed back the WARP's RF output to a PCB antenna (3 dBi gain) that is fixed to the smartphone.

TABLE I. 802.11G DATA RATES ACHIEVABLE ON WARP

Gross Rate	Achievable Rate	Modulation	FEC	Payload Air Time
6 Mbps	5.3 Mbps	BPSK	1/2	1.979 ms
9 Mbps	7.6 Mbps	BPSK	3/4	1.198 ms
12 Mbps	9.8 Mbps	QPSK	1/2	0.899 ms
18 Mbps	13.6 Mbps	QPSK	3/4	0.599 ms
24 Mbps	17.0 Mbps	16-QAM	1/2	0.449 ms
36 Mbps	22.2 Mbps	16-QAM	3/4	0.300 ms
48 Mbps	26.4 Mbps	64-QAM	2/3	0.225 ms
54 Mbps	28.1 Mbps	64-QAM	3/4	0.200 ms

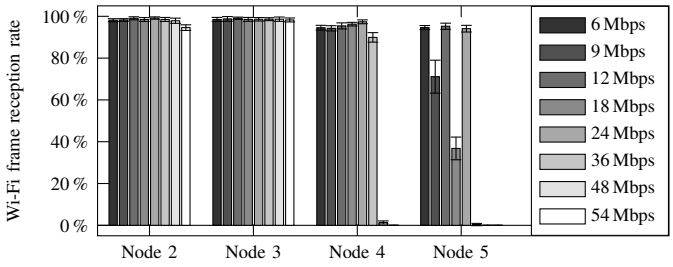
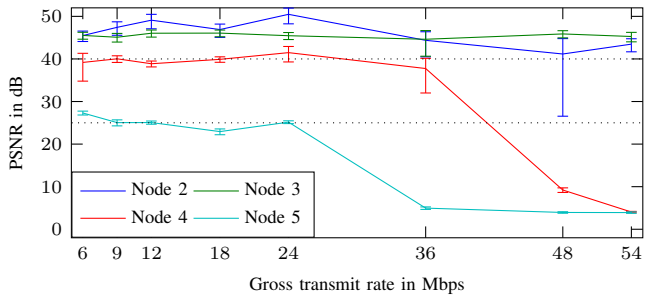
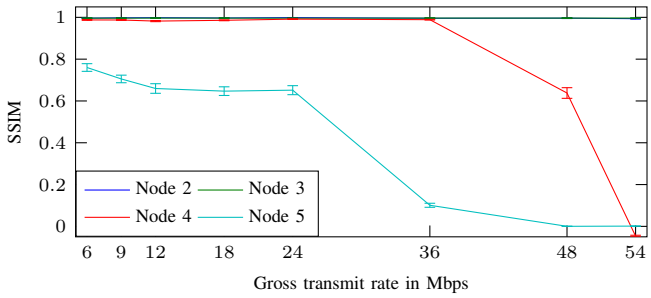


Fig. 6. Wi-Fi frame reception rates of base layer frames transmitted at different gross data rates at 10 dBm transmit power.



(a) PSNR



(b) SSIM

Fig. 7. Video qualities when the transmit power is fixed at 10 dBm and the transmit rate of the base layer changes while keeping the first and second layer fixed at 36 Mbps.

retransmissions. Nodes 2 to 5 were receivers, on which we measured the number of received Wi-Fi frames and the quality of the video playback in terms of the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM).

B. Experiment Definition

Each of our experiments lasted 20 seconds (480 video frames at 24 frames per second). We measured video qualities on a per-frame basis and frame reception rates per second and plotted the average including 99 % confidence intervals. The selection of the performed experiments was based on the fact that each video layer had minimum requirements on throughput. As we do not focus on video compression, the bit rates per frame are comparably high. However, this allows us to test our system under load requirements for high-definition videos. For our frame size of 800×480 pixels and 8×8 pixel block size, we required 17.280 Mbps for all three grayscale video layers (1.152 Mbps for the base, 5.760 Mbps for the first and 10.368 Mbps for the second layer). According to [3] and considering 1348 payload bytes per Wi-Fi frame, the achievable data rates lie below the gross data rates (see Table I). Hence, we needed to choose a data rate of 36 Mbps to have sufficient throughput for our 17.280 Mbps video stream. This 36 Mbps setting acts also as the baseline for further experiments. To analyze the video quality as well as energy efficiency, we defined two sets of experiments: (1) We used a transmit power of 10 dBm and kept the first and second layer's bit rate fixed at 36 Mbps and changed the rates of the base layer to all rates given in Table I. (2) We fixed the rates of the base, first and second layers to 6, 24 and 48 Mbps and changed the transmit power between −12 dBm to 18 dBm in steps of 3 dB.

C. Evaluation of Transmit Rate Variations

In the following, we present our results. In Figure 6, we show the rates of successfully received Wi-Fi frames, i. e., with correct frame check sum (FCS), for each receiving node. With the given transmit power, Nodes 2 and 3 have almost no packet loss, while Node 4 only receives packets up to a rate of 36 Mbps. Node 5 is the most distant node from the transmitter and receives well at rates of 6, 12 and 24 Mbps. Higher rates result in almost no reception, while rates of 9 and 18 Mbps endure higher frame loss than their neighboring higher rates. This effect is most likely due to the fact that at 9 and 18 Mbps each *three* data bits are protected by one FEC bit, while at 6, 12 and 24 Mbps, *each* data bit is protected by one FEC bit. We conclude that gaining throughput by increasing modulation orders is more efficient than reducing FEC.

Regarding video quality (see Figure 7), we observe that Nodes 2, 3 and 4 have PSNR values around 40 dB and SSIM values close to one, which means that they have very good video quality. Compared to Node 5, they receive the first and second layers in addition to the base layer as they have low frame loss rates at 36 Mbps (at which the higher video layers are transmitted). Node 5 does not receive the higher layers, therefore, it only achieves an acceptable video quality around 25 dB (PSNR) or 0.7 (SSIM). As soon as we increase the base layer's transmit rate above the point of low frame loss rates, we encounter very bad video qualities below 10 dB (PSNR) or 0.2 (SSIM), even though the higher layers might still be receivable. We conclude that video quality is highly correlated with frame loss rates and that it is essential to ensure a good reception of the base layer which which alone gives acceptable video quality.

D. Evaluation of Transmit Power Variations

In our second set of experiments we analyzed the influence of transmit power on video reception. We selected a robust

transmit rate of 6 Mbps for the base layer and less robust rates of 24 Mbps and 48 Mbps for the first and second layers. Figure 8 illustrates that the higher the transmit rate, the higher the transmit power requirements to successfully receive the Wi-Fi frames. However, if the transmit power gets too high, transmission at high transmit rates becomes impossible, which is most likely due to the errors introduced by high and therefore non-linear amplification. The effect is observable in Figure 8 (c) at a transmit power of 18 dBm. We also observe that nodes closer to the transmitter are able to receive at lower powers than nodes that are further away. Only at low transmit powers Node 3 has a better reception than Node 2. This is either due to different WARP boards whose receiver sensitivities can vary from board to board, or it is due to the direction the smartphones are facing in the room. As the antenna of Node 2 faces away from the transmitter it might receive noisier signals than Node 3.

Video qualities illustrated in Figure 9 reflect the observations of the frame reception rates. At low transmit powers Nodes 2 and 3 already get medium video quality that increases to very good quality as soon as transmit powers reach 6 dBm so that the second layer is receivable. Video reception at Nodes 4 and 5 start at -3, respectively -6 dBm with acceptable quality.

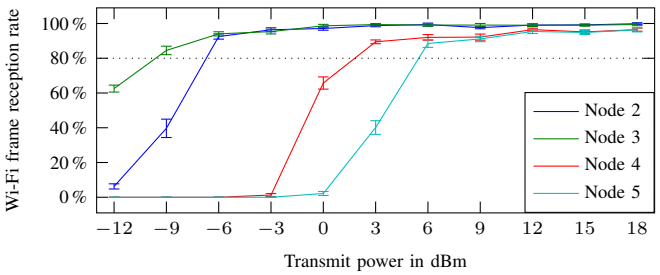
E. Optimizing Energy Consumption

We analyze the energy consumption at the transmitter, while meeting the requirement that the video quality at all nodes stays between 25 dB and 40 dB. To this end, we define a PSNR per consumed energy metric Q (in dB/mWs), where energy is transmit power P (in mW) times the air time $t(R)$ (in s) of a 1348 byte payload frame at transmit rate R (see Table I). For each video frame, we need 5 of these Wi-Fi frames for the base, 23 for the first and 41 for the second layer (ignoring that the last frame is not full). The metric is defined as follows (for powers in linear scale):

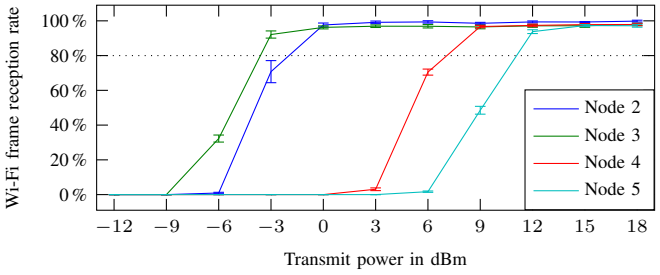
$$\tau(q) = \begin{cases} 0\,\text{dB} & \text{if } q < 25\,\text{dB} \\ 40\,\text{dB} & \text{if } q > 40\,\text{dB} \\ q & \text{else} \end{cases}$$

$$Q = \frac{[\tau(q_{\text{Node 2}}) + \tau(q_{\text{Node 3}}) + \tau(q_{\text{Node 4}}) + \tau(q_{\text{Node 5}})]/4}{5 \cdot t(R_{\text{base}})P_{\text{base}} + 23 \cdot t(R_{1^{\text{st}}})P_{1^{\text{st}}} + 41 \cdot t(R_{2^{\text{nd}}})P_{2^{\text{nd}}}}$$

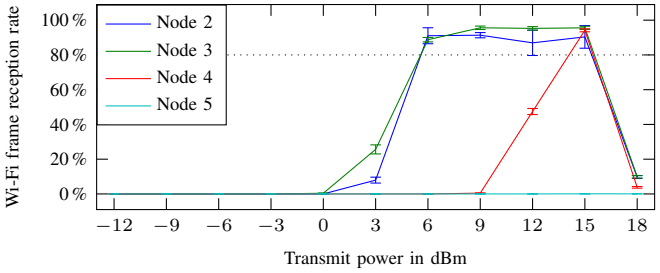
Our baseline is 36 Mbps and 10 dBm for all layers leading to $Q = 29.4\,\text{dB}/0.207\,\text{mWs} = 142\,\text{dB/mWs}$ according to values from Figure 7 (a). Considering the results in Figures 8 and 9, we see that the base layer frame reception rate should not fall below 60 % for a minimum of 25 dB (PSNR). To reach at least 40 dB, the first and second layers should be received with a frame reception rate higher than 90 %. To achieve 25 dB at all nodes, the base layer has to be transmitted with 6 Mbps and 6 dBm. If we do not transmit the other layers, we get $Q = 25.0\,\text{dB}/0.039\text{mWs} = 641\,\text{dB/mWs}$, which is more than four times more efficient than the baseline transmission. To increase video quality, we have to activate the first and second layers. Setting the first layer to -3 dBm and 24 Mbps and the second one to 6 dBm and 48 Mbps allows to maximize video qualities at Nodes 2 and 3, while giving minimum qualities to Nodes 4 and 5. This leads to $Q = 32.5\,\text{dB}/0.087\,\text{mWs} = 374\,\text{dB/mWs}$, which is almost three times more efficient than the baseline, and results in a 3 dB higher average video quality.



(a) Base layer at 6 Mbps

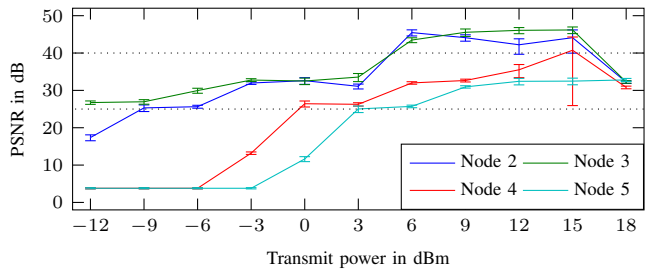


(b) First layer at 24 Mbps

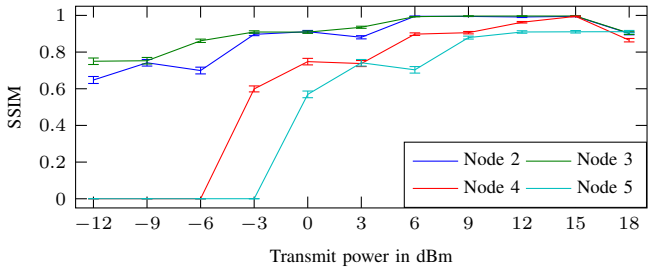


(c) Second layer at 48 Mbps

Fig. 8. Wi-Fi frame reception rates when keeping the base, first and second layer transmit rates fixed at 6, 24 and 48 Mbps, while varying the transmit powers.



(a) PSNR



(b) SSIM

Fig. 9. Video qualities when keeping the base, first and second layer transmit rates fixed at 6, 24 and 48 Mbps, while varying the transmit powers.

V. Discussion and Future Work

Our experiments show that our solution to allow applications to change physical layer parameters by using flow tables in a VPN service on Android smartphones is practical. Using a VPN service to implement an abstraction layer has multiple benefits. First, existing IP-based applications can profit from physical layer optimizations which are transparent to them. Second, the service abstracts from complex physical layer parameters by translating application requirements to physical layer settings that can vary between wireless technologies. Third, the flow-based approach avoids an over-optimization on a per-frame basis. Instead, applications need to set requirements only once per flow and our service is responsible to choose and adapt physical layer settings over time.

In our evaluation, we demonstrate that our solution is suitable to implement and optimize application-centric physical layers that are aware of application requirements. For video streaming, our results show that adjusting transmit power and transmit rates already allows optimizations that help to meet video quality requirements at receivers while saving energy at the transmitter. Even though the variation of the parameters mentioned above could be implemented on off-the-shelf hardware with special drivers or firmware modifications, our solution is capable to be extended to adjust more physical layer parameters or even exchange complete physical layer implementations on the SDRs, depending on application requirements. In future work, we plan to extend our implementation with advanced schemes that are currently being researched in theory and might end up in future fifth generation wireless networks. Additionally, we intend to support standardized SVC solutions and additional applications in the areas of gaming and emergency communication.

VI. Related Work

Our work is related to three main areas, including (1) the access to lower communication layers on customer off-the-shelf (COTS) hardware, SDRs and specialized experimental platforms, (2) flow based networking approaches with focus on the wireless domain, and (3) scalable video coding schemes.

The highest flexibility to evaluate physical layer schemes is offered by SDRs such as the Wireless Open-Access Research Platform (WARP) [1], the Universal Software Radio Peripheral (USRP) or AirBlue [16]. All platforms are build around FPGAs that allow real-time capable implementations of wireless communication schemes. For the WARP, there even exists an IEEE 802.11g compatible reference design usable as starting point for research applications. Nevertheless, SDR platforms are often bulky and limit mobility. Therefore, solutions that unlock lower layer modifications on COTS hardware are desirable. Examples are the bcmon project [10] that activates monitor mode and packet injection on Android smartphones or the Wireless MAC processor [20] respectively MAClets [5] that allow the implementation of state machines or exchangeable building blocks to control time critical tasks in Wi-Fi chipsets. Some Wi-Fi drivers also support the manual setting of transmit rates and transmit powers. As these implementations are chipset specific and often not available on smartphones, we decided to build our framework with SDRs, which allows us to also support future applications that exceed the capabilities of current wireless chipsets.

To bundle and manage network frames that logically belong together, we rely on the concept of flows, an idea introduced in the 1960s [7]. Flows can either be extracted from header information, for example ports and addresses, or they can be explicitly defined by using flow labels as in IPv6 [17]. OpenFlow [15] is a project implementing flows in network hardware to allow controllers direct access to flow tables, for example, in switches to control how data is forwarded. This implements the concept of software-defined networking (SDN). OpenRoads or OpenFlow Wireless [21] is the corresponding project for wireless systems. A major application is to allow handovers between wireless technologies and to manage wireless networks. OpenRadio [4] is a more application-centric solution that aims at developing a platform that supports multiple physical layers that can be adjusted to the needs of an application. However, it misses a complete implementation and extensive evaluation of this approach.

Video streaming is already evaluated in various works. The authors of [11] focus on flow-based control for video streaming services and significantly enhance the quality of experience (QoE) of video on demand (VoD) services like YouTube. Therefore, they rely on deep packet inspection (DPI) and application metrics. There are also works like [12] that focus on optimizing physical layer transmission for video applications in multi-antenna systems. The physical layer knows which bits are more important for a successful video transmission and can adjust according to channel conditions. Last but not least, SVC is another solution to cope with fluctuating network throughput by encoding video into multiple quality layers that have to arrive at receivers with different priorities. This makes SVC streams ideal applications for SDN. In [13], the authors create flow based routing protocols to cope with varying network conditions as well as playback device properties. In practise, SVC extensions exist for the H.264 and H.265 codecs [9], [18]. However, the support for mobile playback devices is still very limited. The authors of [14] offer at least a software-based player for Android devices. In this work, we neither intend to reinvent scalable video codecs, nor do we want to compete against existing solutions. Our intention is to have a lightweight video codec based on a simple implementation that runs smoothly on mobile devices and is easily extensible so that we can combine it with the concept of flows as well as physical layer optimizations.

VII. Conclusion

In this work, we present a solution that allows mobile applications to take advantage of physical layer properties when transmitting data. Our framework relies on the concept of flows to avoid over optimizing the physical layer on a per frame basis. For each flow, an application can define requirements on the physical transmission such as the use of robust modulation schemes and high power transmissions if a flow should be received by as many receivers as possible, or the combination of high throughput and low transmit power to only serve nodes in close proximity. To abstract from the complexity of the physical layer and to support multiple applications per device, we introduce an intermediary layer that handles application requirements and sets corresponding physical layer parameters according to flow tables. By implementing our solution as VPN service on Android, it becomes transparent to existing applications that rely on IP based communication.

As an example application to evaluate our solution's performance, we chose scalable video coding on Android smartphones. We designed and implemented a lightweight scalable video codec that focuses only on offering multiple quality layers that can be transmitted over separate flows. To be able to change physical layer parameters on smartphones and to have flexibility for future research, we use software-defined radios that currently implement the 802.11g standard in an easily extendable way. Attaching the SDR's antenna to the smartphones allows to still support mobility around the SDR.

In our evaluation, we analyze the effect of choosing different modulation schemes and transmit powers on frame reception rates and video quality. Thereby, we show that video transmissions can be optimized for both maximum video qualities and minimal energy consumption, which is especially important for mobile devices.

ACKNOWLEDGMENT

This work has been funded by the German Research Foundation (DFG) in the Collaborative Research Center (SFB) 1053 "MAKI – Multi-Mechanism-Adaptation for the Future Internet" and by LOEWE CASED.

REFERENCES

[1] (2014) Rice university WARP project. [Online]. Available: http://warp.rice.edu

[2] (2014) Statcounter global stats – top 8 mobile operating systems from july 2013 to july 2014. [Online]. Available: http://gs.statcounter.com/#mobile_os-ww-monthly-201307-201407

[3] (2014) Warp project – throughput benchmarks. [Online]. Available: http://warpproject.org/trac/wiki/802.11/Benchmarks/Throughput

[4] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "OpenRadio: A programmable wireless dataplane," in *Proc. of the 1st Workshop on Hot Topics in Software Defined Networks (HotSDN)*, 2012.

[5] G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, F. Gringoli, and I. Tinnirello, "MAClets: Active MAC protocols over hard-coded devices," in *Proc. of the 8th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2012.

[6] Cisco Systems Inc. (2014) Cisco VNI: Forecast and methodology. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

[7] L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton University Press, 1962.

[8] Gartner. (2014) Gartner says worldwide tablet sales grew 68 percent in 2013, with android capturing 62 percent of the market. [Online]. Available: http://www.gartner.com/newsroom/id/2674215

[9] P. Helle, H. Lakshman, M. Siekmann, J. Stegemann, T. Hinz, H. Schwarz, D. Marpe, and T. Wiegand, "A scalable video coding extension of HEVC," in *Proc. of the Data Compression Conference (DCC)*, 2013.

[10] O. Ildis, Y. Ofir, and R. Feinstein. (2013) Wardriving from your pocket — Using wireshark to reverse engineer broadcom wifi chipsets. [Online]. Available: http://bcmon.blogspot.de/

[11] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran-Gia, "SDN-based application-aware networking on the example of youtube video streaming," in *Proc. of the 2nd European Workshop on Software Defined Networks (EWSDN)*, 2013.

[12] A. Jeyaraj and M. El Zarki, "A real-time cross-layer design of the multimedia application layer with a mimo based wireless physical layer," in *Proc. of the 3rd International Symposium on Wireless Pervasive Computing (ISWPC)*, 2008.

[13] S. Laga, T. Van Cleemput, F. Van Raemdonck, F. Vanhoutte, N. Bouten, M. Claeys, and F. De Turck, "Optimizing scalable video delivery through openflow layer-based routing," in *Proc. of the IEEE Network Operations and Management Symposium (NOMS)*, 2014.

[14] Y.-S. Li, C.-C. Chen, T.-A. Lin, C.-H. Hsu, Y. Wang, and X. Liu, "An end-to-end testbed for scalable video streaming to mobile devices over HTTP," in *Proc. of the IEEE International Conference on Multimedia and Expo (ICME)*, 2013.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, 2008.

[16] M. C. Ng, K. E. Fleming, M. Vutukuru, S. Gross, Arvind, and H. Balakrishnan, "Airblue: A system for cross-layer wireless protocol development," in *Proc. of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, 2010.

[17] J. Rajahalme, A. Conta, B. Carpenter, and S. Deering, "IPv6 flow label specification," RFC 3697, 2004.

[18] H. Schwarz and M. Wien, "The Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Signal Processing Magazine*, vol. 25, no. 2, 2008.

[19] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, 2012.

[20] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, and F. Gringoli, "Wireless MAC processors: Programming MAC protocols on commodity hardware," in *Proc. of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2012.

[21] K.-K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. McKeown, "The stanford openroads deployment," in *Proc. of the 4th ACM International Workshop on Experimental Evaluation and Characterization (WINTECH)*, 2009.