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
Motivation

Applications

Video Streaming (throughput, efficiency)
Gaming (latency)
Emergency (reliability, latency)
Messaging (reliability)

Physical Layers

Trans. Layer
Netw. Layer
Data Layer

Direct Influence

Custom:
e.g. OFDMA
Amateur/HAM radio
Motivation
Enabling Practical Cross-Layer Research
Contents

- Motivation
- Problem Description
- Framework Implementation
- Evaluation
- Conclusion and Outlook
Problem Description
Solving Problems the “Internet” Way

**Application Layer**
- Overlay Network (e.g. P2P Mesh Network)

**Transport Layer**

**Network Layer**

**Data Link Layer**

**Physical Layer**

- we build overlays
- e.g. Video Streaming
- TCP/IP, UDP/IP fixed in Linux kernel
- WiFi, Bluetooth, LTE fixed in hardware
Solving Problems the “Wireless Networking” Way

MAC layer for WiFi access point, station or AdHoc mode with Ethernet bridge
Real time signal processing of WiFi PHY

Application Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer
Android + Overlay

Application Layer

Overlay Network (e.g. P2P Mesh Network)

Transport Layer

Network Layer

Data Link Layer

Physical Layer

WARP + 802.11

Application Layer

Application Layer

Transport Layer

Network Layer

Data Link Layer

Physical Layer
(Android + Overlay) + (WARP + 802.11)
A Cross-Layer Solution

- Control over all layers by our overlay network (Java Code)
- Overlay Network (e.g. P2P Mesh Network)
- Data Link and Physical Layer Control
- Data Link Layer
- Physical Layer

Time critical: ACK, signal processing, …
1st Option: Full Control by Application

Application controls all layers

+ Highest flexibility and optimization potential

- Application needs details about all usable physical layers
- Hard to share control between multiple applications
2nd Option: Flow Control Layer

Application controls network layer and above and sets requirements for lower layers

- Abstraction form lower layer details
- Flow Control service that sets physical layer parameters according to requirements
- Intermediate Layer coordinates access to physical resources
- Transparent for existing applications
Framework Implementation
Interfacing WARP with Android

- **Wireless Interfaces**
  - GPIO Pins
  - no USB device port
  - 1 Gbps Ethernet

**Should use**

- **Wireless Interfaces**

**Use to interface**

- **SDIO through SD card slot**
- **USB host/OTG port**
- **USB-to-Ethernet**

**GPIO = General Purpose Input/Output**
**SDIO = Secure Digital Input/Output**
**OTG = On-the-Go**
A Cross-Layer Solution

Application

Overlay Network (e.g. P2P Mesh Network)

Data Link and Physical Layer Control

UDP/IP/Ethernet

Data Link Layer

Physical Layer

USB-to-Ethernet
Mobility Preservation

WARP software-defined radio

Antenna

Smartphone

Antenna interface

USB-to-Ethernet Adapter
Flow Control Service

- **Application**
  - UDP/IP port 40
  - UDP/IP port 41
  - UDP/IP port 42

- **Linux Stack**

- **TUN Device**
  - Filter Flows and attach WARP Header

- **Flow Table**
  - Filter
  - PHY Setup

- **WARP "VPN" Service**

Transparent solution
Applications simply create sockets

We use the Android VPN service API to read from TUN devices (no rooting required!)
Flow Control Service

Incoming network layer packet:

- Network layer header (e.g. IP header)
- Transport layer header (e.g. UDP header)
- Application layer payload

Flow Filtering with Flow Table:

<table>
<thead>
<tr>
<th>Mask</th>
<th>Value</th>
<th>Requirement → MAC/PHY Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP Port</td>
<td>UDP, 40</td>
<td>High robustness → PHY: BPSK, 1/2 FEC</td>
</tr>
<tr>
<td>UDP Port</td>
<td>UDP, 42</td>
<td>High throughput → PHY: 64-QAM, 4/5 FEC</td>
</tr>
<tr>
<td>default</td>
<td></td>
<td>Best effort → PHY: WiFi standard</td>
</tr>
</tbody>
</table>

Attaching WARP header:

- WARP header
- Network layer packet
Flow Control Service

Transparent solution
Applications simply create sockets

We use the Android VPN service API to read from TUN devices (no rooting required!)

Flow Table entries can be changed by applications

Frames containing WARP header are packed into UDP datagrams and tunneled to the WARP over Ethernet

WARP interprets header and sets PHY and MAC settings accordingly
Flow Control Service

WARP decides which frames to drop or receive depending on:
- FCS check
- MAC address
- Flow Filter result

WARP appends additional information as WARP header
Flow Control Service

Application

UDP/IP port 40 → UDP/IP port 41 → UDP/IP port 42 → Linux Stack

TUN Device

Filter Flows and attach WARP Header

Flow Table Filter

PHY Setup

UDP/IP/MAC Tunnel to WARP over Ethernet

Linux Stack

USB to Ethernet Adapter

Interpret Settings → Custom PHY

WARP Software-defined Radio

WARP VPN Service interprets WARP header and logs results

Interpret WARP Header

PHY Log

CSI

RSSI

Rate

WARP Software-defined Radio

Flow Tables, ...
Flow Control Service

Application

UDP/IP port 40
UDP/IP port 41
UDP/IP port 42

Linux Stack

TUN Device

Filter Flows and attach WARP Header

Flow Table Filter PHY Setup

WARP “VPN” Service

UDP/IP/MAC Tunnel to WARP over Ethernet

USB to Ethernet Adapter

Interpret Settings Custom PHY

WARP Software-defined Radio

USB to Ethernet Adapter

Interpret Settings Custom PHY

WARP Software-defined Radio

APP and PHY in Harmony | M. Schulz et al. | TU Darmstadt | MAKI
Evaluation
Example Application: Scalable Video Coding

Base layer:
APP: Low video quality
PHY: Very robust transmission

First layer:
APP: Increased video quality
PHY: Normal robustness

Second layer:
APP: Highest video quality
PHY: High throughput
Motion-JPEG based Lightweight Scalable Video Coding (SVC)
Evaluating Video Quality

Peak Signal-to-Noise Ratio (PSNR)

Structural Similarity Index (SSIM)
Experimental Setup

Android + WARP Node

Base layer frame reception rates at 10 dBm transmit power
Image Quality Comparison

1\textsuperscript{st} and 2\textsuperscript{nd} layer at 36 Mbps, base layer changing

![Graph showing PSNR and SSIM vs. Gross transmit rate for different nodes](image)

- PSNR in dB
- SSIM

Node 2, Node 3, Node 4, Node 5

---

APP and PHY in Harmony | M. Schulz et al. | TU Darmstadt | MAKI
Image Quality Comparison

base at 6 Mbps, 1\textsuperscript{st} at 24 Mbps, 2\textsuperscript{nd} at 48 Mbps

More information on optimizing power consumption while maximizing image quality is available in our paper.
Conclusion and Outlook
Conclusion

- Goal: Enabling Practical Cross-Layer Research
- Design and implementation of flow-based framework that allows applications to control physical layer properties
- Implementation of lightweight scalable video codec as example application
- Evaluation in testbed using Android smartphones and WARP software-defined radios
- Tuning parameters such as transmit power and rate allow to optimize image qualities while reducing power consumption
Outlook

- Further evaluation with existing SVC codecs
- Introduction of back channel for adaptive parameter adjustment
- Evaluation with more parameters as well as different modulation schemes and physical layers