UniFlex
A Framework for Simplifying Wireless Network Control

P. Gawłowicz, A. Zubow, M. Chwalisz, A. Wolisz
Motivation

- Wireless network evolved to be highly complex system

- Constantly changing radio environment requires frequent tuning of parameters

- Control applications are usually implemented case-by-case
Objectives

- Reduce threshold for experimentation and shorten time of prototyping
- Provide framework for development of novel control and management solutions wireless networks
- Improve performance and efficiency of wireless networks
System Model

Radio Devices

Node

Node

Controller

Node

Wireless Network

Control Channel
Device Programming Interface

Native Device Programming Interface (NDPI)

Device

WiFi NDPI

LTE NDPI

ZigBee NDPI

SDR NDPI

WiFi

LTE

ZigBee

SDR
Requirements

- Coordinated collection of information from and execution of commands on different:
  - protocol layers (*cross-layer*),
  - heterogeneous devices (*cross-technology*)
  - multiple nodes (*cross-node*) within network
- Possibility to implement logically centralized and physically distributed control applications
- Support for multiple levels of control for scalability reasons
- Support for proactive and reactive control schemes
- A high-level API for control of operation of individual wireless devices and groups of devices
- Unification of different NDPI
UniFlex Architecture
Northbound Interface

- Based on **event** exchange mechanism + Remote Procedure Calls

```python
@on_event(NewNodeEvent)
def my_new_node_cb(self, event):
    node = event.node
```

<table>
<thead>
<tr>
<th>NodeProxy</th>
</tr>
</thead>
<tbody>
<tr>
<td>+get_delay()</td>
</tr>
<tr>
<td>+get_time_synchronization_accuracy()</td>
</tr>
<tr>
<td>+get_device_proxy(name)</td>
</tr>
<tr>
<td>+get_protocol_proxy(name)</td>
</tr>
<tr>
<td>+get_control_application_proxy(name)</td>
</tr>
<tr>
<td>+send_event(event)</td>
</tr>
<tr>
<td>+subscribe_for_events(eventType, callback)</td>
</tr>
<tr>
<td>+unsubscribe_from_events(eventType)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DeviceProxy/ProtocolProxy</th>
</tr>
</thead>
<tbody>
<tr>
<td>+callback(callbackFunction)</td>
</tr>
<tr>
<td>+delay(relativeTime)</td>
</tr>
<tr>
<td>+exec_time(absoluteTime)</td>
</tr>
<tr>
<td>+send_event(event)</td>
</tr>
<tr>
<td>+subscribe_for_events(eventType, callback)</td>
</tr>
<tr>
<td>+unsubscribe_from_events(eventType)</td>
</tr>
<tr>
<td>+command_1(args)</td>
</tr>
<tr>
<td>+command_N(args)</td>
</tr>
</tbody>
</table>

Set of commands that are translated to Native Programming Interface of Device/Protocol
Southbound Interface

- translates function calls from control applications into NDPI
- unifies different NDPIs
Distributed Middleware

- The framework is an distributed middle-ware platform that:
  - provides communication channel for control applications
  - exposes NDPI of network devices to control applications
- It takes care about maintaining up-to-date information, including:
  - node discovery and monitoring connection between all nodes
  - notifying control applications about changes
Implementation

- **Event** delivery mechanisms implemented using PUB/SUB sockets from ZMQ
- **RPC** implemented on top of unicast event mechanism
- Integration with Node-RED – a graphical language for pipeline data processing
Calling Examples

@on_event(PacketLostEvent)  ➙ subscribe for PacketLostEvent
def my_pkt_lost_cb(self, event):
    # get device proxy from node proxy
device = event.device
    # execution of blocking call
    pwr = device.get_tx_power()
    # delay execution of call by 3 seconds
    device.delay(3).set_tx_power(pwr+2)
    # schedule execution of non-blocking call
    t = datetime.now() + timedelta(seconds=6)
device.exec_time(t).callback(my_get_power_cb).get_tx_power()
Example Applications

Mobility Management – Handover

Interference management through air-time management
Conclusions

- UniFlex is a framework that simplifies prototyping of novel wireless solutions
- It provides rich API for control and management of network entities
- It allows to implement local, central and hierarchical control planes.
- Its usability was proved in several implemented use-cases.
Thank you!