iJOIN

INFSO-ICT-317941

INFSO-ICT-317941 iJOIN

IR 6.1

Proof-of-concept requirements, evaluation criteria, and performance measures

Editor:	Marco Consonni - HP
Deliverable nature:	Confidential
Suggested readers:	iJOIN GA
Due date:	July 31st , 2013
Delivery date:	July 31st , 2013
Version:	1.0
Total number of pages:	29
Reviewed by:	GA members
Keywords:	iJOIN
Resources consumed	10.93 PM

Abstract

This internal report is aimed at defining the available environments where iJOIN proof-of-concept experiments will be run. The specification of testbed environments includes requirements and constraints for deploying the components to test. This further describes the implementation guidelines for the three testbeds so as to make the implementation of relevant candidate technologies over the respective platforms easier.

The report will serve as baseline information for the implementation phase of three testbeds.

List of authors

Company	Author
HP	Giovanni Giuliani, Marco Consonni, Marco Di Girolamo
Intel	Umer Salim
TUD	Jens Bartelt
UC3M	Carlos J. Bernardos

History

Modified by	Date	Version	Comments
Marco Consonni , Marco Di Girolamo, Umer Salim	31-7-2013	1.0	Final report IR6.1.

Table of Contents

List of authors ... 2
History ... 3
Table of Contents ... 4
List of Figures ... 5
List of Tables ... 6
Abbreviations ... 7
1 Introduction ... 8
2 Executive Summary ... 9
3 Testbed Physical Description and Setup ... 10
3.1 RANaaS Testbed ... 10
3.1.1 Testbed Concept ... 10
3.1.2 OpenStack IaaS Platform ... 12
3.1.3 HP Cloud Trial Configuration ... 14
3.1.3.1 Compute Services ... 14
3.1.3.2 Network Connectivity ... 14
3.1.3.3 Advanced Networking Features ... 15
3.2 Joint Access and Backhaul Testbed ... 16
3.2.1 Testbed Concept ... 16
3.2.2 Physical Layer and Digital Baseband ... 16
3.2.3 60 GHz Frontend and Antennas ... 18
3.3 SDN Testbed ... 19
3.3.1 Motivation ... 19
3.3.2 Testbed Description ... 19
3.3.3 Demonstrator Capabilities ... 20
4 Implementation Guidelines ... 21
4.1 RANaaS Testbed ... 21
4.1.1 Testing iJOIN RANaaS in the Cloud ... 21
4.1.2 The Testing Environment ... 22
4.2 Joint access and Backhaul Testbed ... 23
4.2.1 Application Example: CT 2.1 In-Network Processing ... 23
4.2.2 Requirements for Development and Evaluation Parameters ... 25
4.3 SDN Testbed ... 26
4.3.1 Application example: CT 4.3 Joint Path Management control ... 26
5 Summary and Conclusion ... 27
Acknowledgements and Disclaimer ... 28
References ... 29

List of Figures

Figure 1 – iJOIN Testbed Coverage 10
Figure 2 - RANaaS Testbed........ 11
Figure 3- IaaS Cloud Platform........ 12
Figure 4 - OpenStack IaaS Cloud Platform 13
Figure 5 - HP Cloud Trial Configuration 14
Figure 6 - Backhaul-in-the-loop platform 16
Figure 7 - Digital Baseband Architecture of Transmitter (top) and Receiver (bottom) of the 60 GHz Demonstrator........ 18
Figure 8 - Analog Frontend of the Transmitter (left) and Receiver (right) of the 60 GHz Demonstrator....... 18
Figure 9 - SDN testbed layout 19
Figure 10. Example of Network Control Demonstration 20
Figure 11- Creating and Using a RAN Services........ 21
Figure 12 - OpenStack Training Appliance........ 22

List of Tables

Table 1: Baseband parameters of the 60 GHz demonstrator 17

Abbreviations

Acronym	Description
API	Application Programming Interface
BER	Bit Error Rate
CLI	Command Line Interface
CT	Candidate Technology
DQPSK	Differential Quaternary Phase Shift Keying
eNB	evolved Node B
EPC	Evolved Packet Core
FPGA	Field Programmable Gate Array
KVM	Kernel-based Virtual Machine
IaaS	Infrastructure as a Service
IF	Intermediate Frequency
iJOIN	Interworking and JOINt Design of an Open Access and Backhaul Network Architecture for Small Cells based on Cloud Networks
iLGW	iJOIN Local Gateway
iNC	iJOIN Network Controller
iSC	iJOIN Small Cell
IT	Information Technology
iTN	IJOIN Transport Node
LO	Local Oscillator
MIMO	Multiple Input Multiple Output
OTA	OpenStack Training Appliance
PC	Personal Computer
QPSK	Quaternary Phase Shift Keying
RAN	Radio Access Network
RANaaS	Radio Network Access as a Service
RPC	Remote Procedure Call
SDN	Software Defined Network
SNR	Signal-to-Noise-Ratio
S/P	Serial-to-Parallel
UE	User Equipment
VM	Virtual Machine
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

1 Introduction

The iJOIN project goal is to overcome the enormous data traffic increase ignited by the pervasive spreading of 4G networks, through the deployment of innovative small cells based on a special evolved architecture. The two technology conceptual pillars underpinning iJOIN architecture are “Radio Access Network as a Service” (RANaaS) and joint radio access/backhaul design and optimization, duly presented and explained in the deliverable D5.1.

As reported in internal report IR4.1, Software Defined Networks (SDN) is another fundamental area of investigation for iJOIN project for meeting requirements of “*higher degree of configurability (e.g. granularity of information rates,) resource sharing and prioritization of operators, traffic shaping, admission control, and load balancing*“ stated for the backhaul networks.

Inside iJOIN, WP6 is in charge of implementing the platforms enabling to run an experimental proof-of-concept of the involved technology.

In the present internal report, the partners involved in the proof-of-concept will provide an assessment of the underlying technologies and of the available experimental facilities. The purpose of such assessment is to support the technology work packages in defining which of the key technology components of iJOIN (in the project glossary, *candidate technologies*) are best suitable to be deployed onto the testbeds, which are the pre-requirements and constraints, what the testing methodology and evaluation criteria should be. This document will therefore describe the existing platforms in the different test beds.

The RANaaS testbed is based upon a general purpose cloud computing platform, at least initially of IaaS (Infrastructure as a Service) type. The report will outline the characteristics of a cloud computing platform most relevant to perform the iJOIN proof-of-concept, and will draw some conclusion about the cloud computing shape best fitting iJOIN’s needs. It will then describe the specific cloud computing facility available at HP Italy Innovation Center premises, detailing the software environment, the supported baseline (operating systems, hypervisors,…), and providing an insight on the actual requirements to meet for deploying iJOIN functionalities into this testbed.

The joint backhaul/RAN testbed is based on a simulation environment available at University of Dresden’s Labs, which will provide a reference implementation of a 60GHz backhaul. Such backhaul instance will be accessible by other iJOIN components through an open interface. This platform will be described in detail, in terms of its overall capabilities, the crop of transmission technologies made available by the demonstrator, and its interface details specifying the way to access its available functions. The technology work packages will hence obtain all the information needed to use the testbed as backhaul endpoint to evaluate the joint optimization algorithms they will develop during the research phase.

The SDN testbed is based on OpenFlow technology that provides functions for reporting the status of the network and modifying the routing tables in the network nodes. These features will be used for implementing algorithms aimed to continuously monitor the network status, calculate the best path between nodes and re-configure the nodes’ routing tables in order to optimize network communications. The testbed is described highlighting how both the iJOIN nodes and the backhaul network links will be emulated and what capabilities will be made available.

2 Executive Summary

This report describes the main activities carried out in WP6 in the first four active months of this work package. This work package deals with the proof of concept for proposed optimized network design in iJOIN which encompasses radio access, backhaul and core network optimization. For demonstration activities of iJOIN, three different test platforms have been selected. The cloud platform provided by HP will be used to demonstrate how proposed technological approaches may benefit from the flexible processing shift to the cloud centres. The 60 GHz platform made available by TUD implements a 60 GHz backhaul following the principle of hardware-in-the-loop. This platform can be harnessed to demonstrate joint access-backhaul algorithms as envisioned in WP2 and WP3. The third demonstrator is the network control demonstrator. This would be exclusively used to demonstrate the novel network layer algorithms and proposals made in iJOIN.

Section 3 of this report gives a detailed introduction and physical description of the all the reference demonstrators. The objective is to well familiarize all the partners with the demonstrators to facilitate the implementation activities in the later stage of the project. Section 3.1 gives a detailed description of the RANaaS test platform. It describes how this cloud platform has been physically set up in HP lab premises. The physical limitations, performance capabilities and computational services are described in detail. Section 3.2 focuses on the 60 GHz backhaul platform. The main concept of this demonstrator, the physical layer and digital baseband design are also provided. Some description about the frontend of this 60GHz backhaul, its properties and the antennas used is also given in the subsequent subsection. The principle of backhaul-in-the-loop is highlighted as well. Section 3.3 provides details about the network control demonstrator. It is described how this platform would be realized based upon the principle of SDN network architecture.

Section 4 gives implementation guidelines for the demonstration platforms of concern. It describes how an engineer can interact and use a particular platform to implement and demonstrate a relevant candidate technology. Section 4.1 is dedicated to the description of the RANaaS test platform. It describes the interaction possibilities with this cloud setup. A training appliance which is a piece of software and can be run on local machines is detailed. This may serve as a first implementation step to see the working of a particular technology on the cloud before actual demonstration in a cloud centre. Section 4.2 describes demonstration principles for the 60 GHz backhaul-in-the-loop platform. In subsequent sub-sections, it describes an implementation example with the WP2/3 candidate technology “In-Network Processing”. Section 4.3 provides the implementation criteria for network control demonstrator.

The concluding remarks of this document can be found in section 5.

3 Testbed Physical Description and Setup

The following picture shows the iJOIN logical architecture and how the testbeds described in this document cover the most relevant areas of investigation:

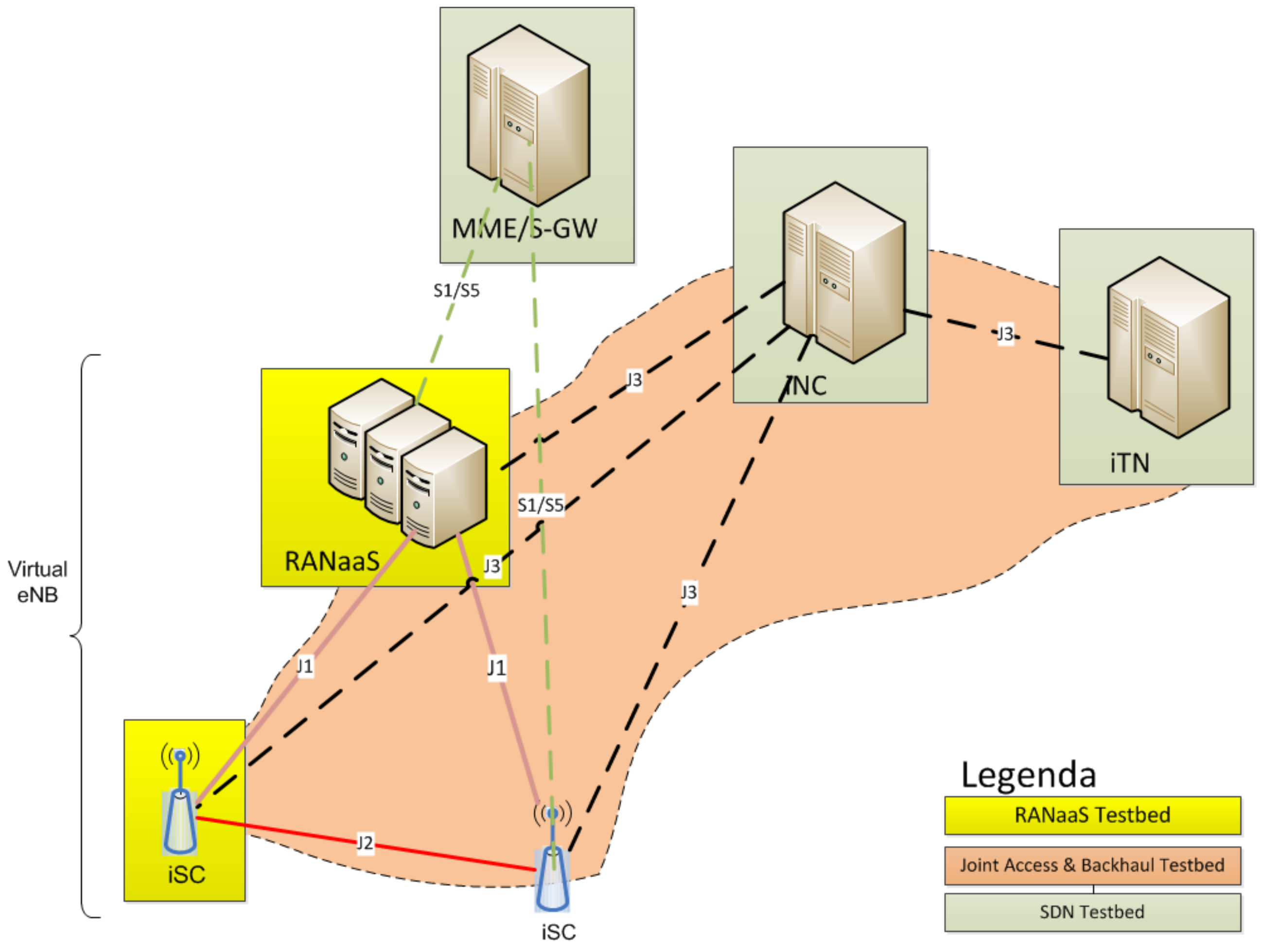


Figure 1 – iJOIN Testbed Coverage

The RANaaS testbed is mainly focussed on emulating the Virtual eNB elements which consists of RANaaS nodes and iSCs. The RANaaS platform will host RAN services implemented as programs running in an Infrastructure as a Service type cloud platform.

The joint Access & Backhaul testbed is intended to be used to investigate and validate the technology for implementing the physical layer of the communication links between the architecture elements. For example, it may provide the physical implementation of J1 and J2 links. These two platforms combined enable the demonstration of principal ideas of iJOIN such as flexible provision of RAN resources through cloud and over the heterogeneous backhaul.

Another important area of innovation in iJOIN is network optimization through network layer algorithms. The objective of the SDN testbed is the implementation of algorithms for orchestrating and configuring the network entities in order to optimize the network communications. As such, the effort is mainly focused on the implementation of the iJOIN network controller while other network entities will be simulated.

3.1 RANaaS Testbed

3.1.1 Testbed Concept

The purpose of RANaaS Testbed is to provide a component for simulating the RANaaS platform, as identified in the iJOIN logical architecture.

The testbed is based on a cloud Infrastructure as a Service (IaaS) platform providing basic IT capabilities like:

- computation services: ability to start Virtual Machines (VMs) running an operating system and, optionally, application software;
- storage services: ability to create storage elements (i.e. volumes) that can be attached to VMs;
- networking services: ability to create network objects like Level 2 (L2) networks, subnets, DHCP services, etc., used for connecting VMs.

In iJOIN, IaaS cloud platform hosts Radio Access Network (RAN) Services that are implemented as a collection of applications (i.e. programs) running on top of VMs.

The following picture shows summarizes the concept:

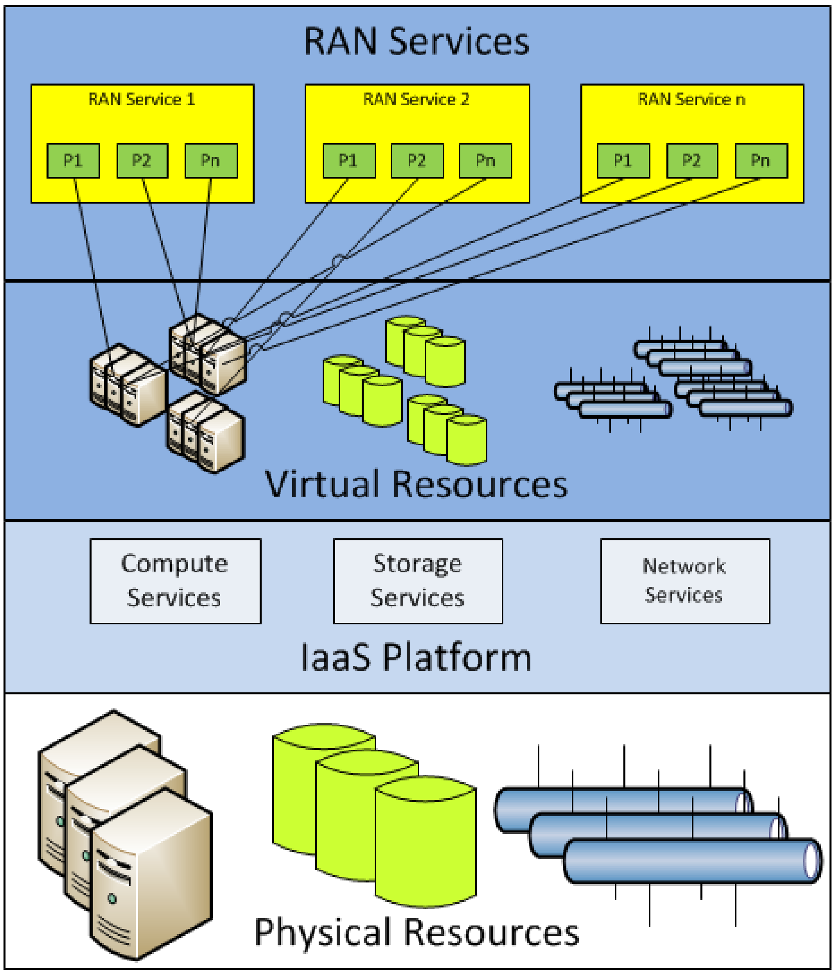


Figure 2 - RANaaS Testbed

In this perspective, a RAN Service is made available by the RANaaS testbed using both IaaS elements (e.g. VMs) and program(s), running on top of VM(s). Programs communicate via Virtual Networks and can store data on Virtual Volumes, if needed.

Virtual objects made available at IaaS level are to be considered as 'enablers' whilst the real RAN Services provided by RANaaS testbed are obtained as the conjunction of IaaS objects plus the software running on top of them.

3.1.2 OpenStack IaaS Platform

In general terms, when services are deployed in a cloud environment, IaaS platform is at the basis of the solution.
The following Figure 3 summarizes the concept:

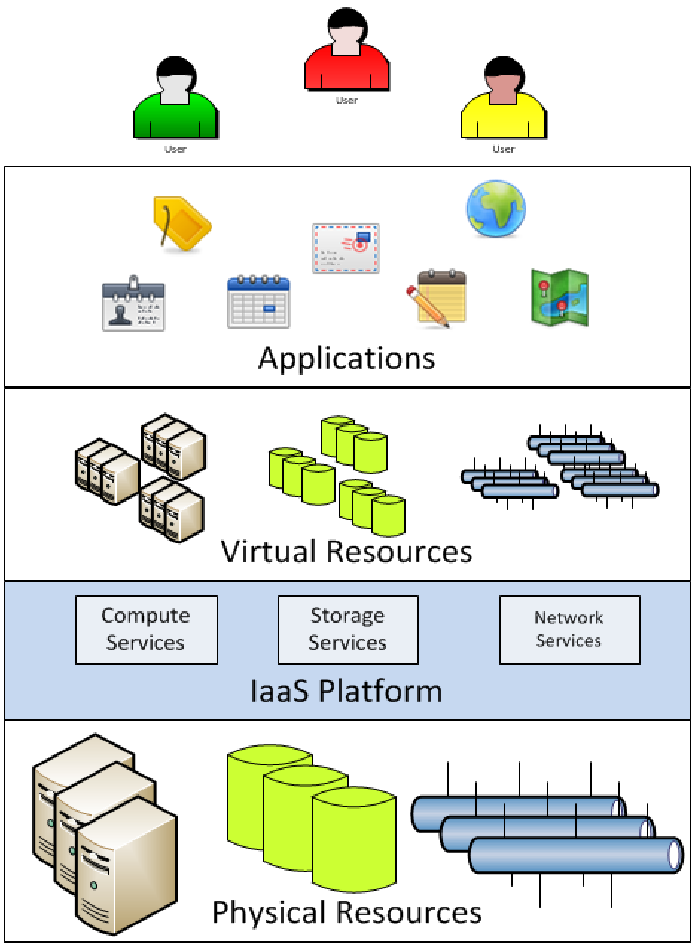


Figure 3- IaaS Cloud Platform

At the bottom of the stack, the physical resources, or physical infrastructure, like physical machines (computing resources), physical disks (storage resources) and physical networks (connection resources) are located.

The IaaS platform is a software layer that provides services for creating virtual resources (virtual machines, virtual disks and virtual networks) that can be used instead of their physical counterparts in order to deploy and run applications. Virtual resources are provided as services meaning that they are created when needed, used to run applications and removed when the application is not anymore needed. The actual computation happens at the physical level but physical resources and applications are not tightly bound to each other. This makes it easier to reuse the physical infrastructure for several purposes, usually at different times.

The combination of the IaaS platform and the underlying physical resources is defined as a **cloud**. In general a **private cloud** is managed by a single organization that is usually the owner of the applications which run on top of it.

The **OpenStack** IaaS Platform is a good choice for fulfilling the project requirements. In addition to being an IaaS Platform (i.e., a software for implementing a private cloud), it is also a project and a community where ideas and code can be shared and leveraged by all participants. In fact, the OpenStack project is backed by an independent Foundation supported by several Corporate Sponsors like Hewlett Packard, IBM, Cisco, Ericsson, Intel, AT&T, Redhat, RackSpace, etc. and a global community with more than 7,000 members representing 850 unique organizations across 88 countries. All source code is freely available under the Apache 2.0 license.

HP ICC has a very strong experience on OpenStack and other open software solutions that implement IaaS platforms (e.g. Eucalyptus and OpenNebula) and will leverage all this knowledge for the benefit of the project.

The following picture shows a very high-level view of OpenStack architecture:

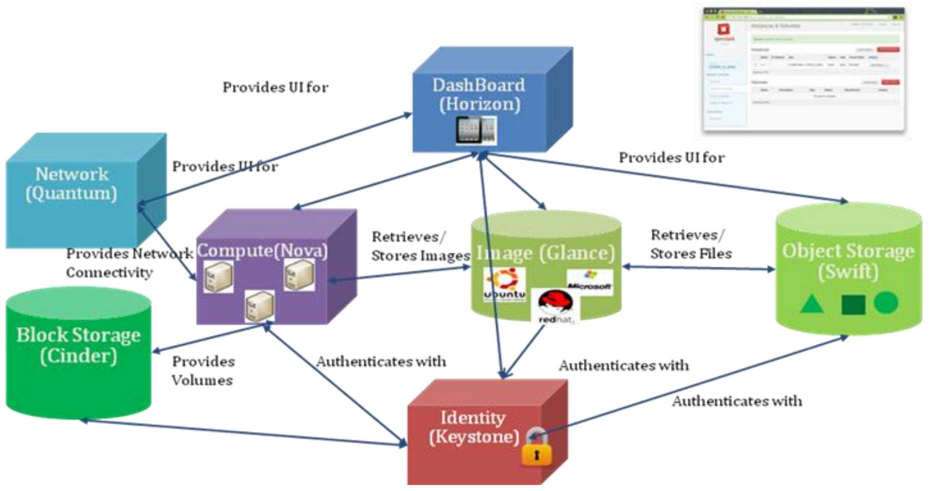


Figure 4 - OpenStack IaaS Cloud Platform

OpenStack is composed of the following modules mapping the fundamental IaaS services:

- **Nova** provides computation services (Virtual Machines)
- **Neutron** (formerly **Quantum)** provides networking services (Virtual Networks).
- **Cinder** provides storage services (Virtual Disks)

In addition it implements some “ancillary” modules:

- **Horizon** provides a web front-end for managing and controlling purposes
- **Glance** implements a catalogue for storing virtual machine images
- **Keystone** implements authentication and authorization.

OpenStack provides a documented and open Application Procedure Interface (API). This is a very important aspect in general - and for project in particular - in order to fully benefit of cloud computing. As a matter of fact, using API it is possible to design and implement tools for the automation of tasks needed when applications are scaled-up or down (software upgrade, application installation and configuration, etc.).

3.1.3 HP Cloud Trial Configuration

HP is hosting an OpenStack Infrastructure-as-a-Service cloud inside its labs in Milan, deployed on a BladeMatrix enclosure, using 8 Blade BL240 Servers with 2 quad-core processors and 24 GB RAM each. A small Storage Area Network connected though optical link provides approx 4 TB of storage.

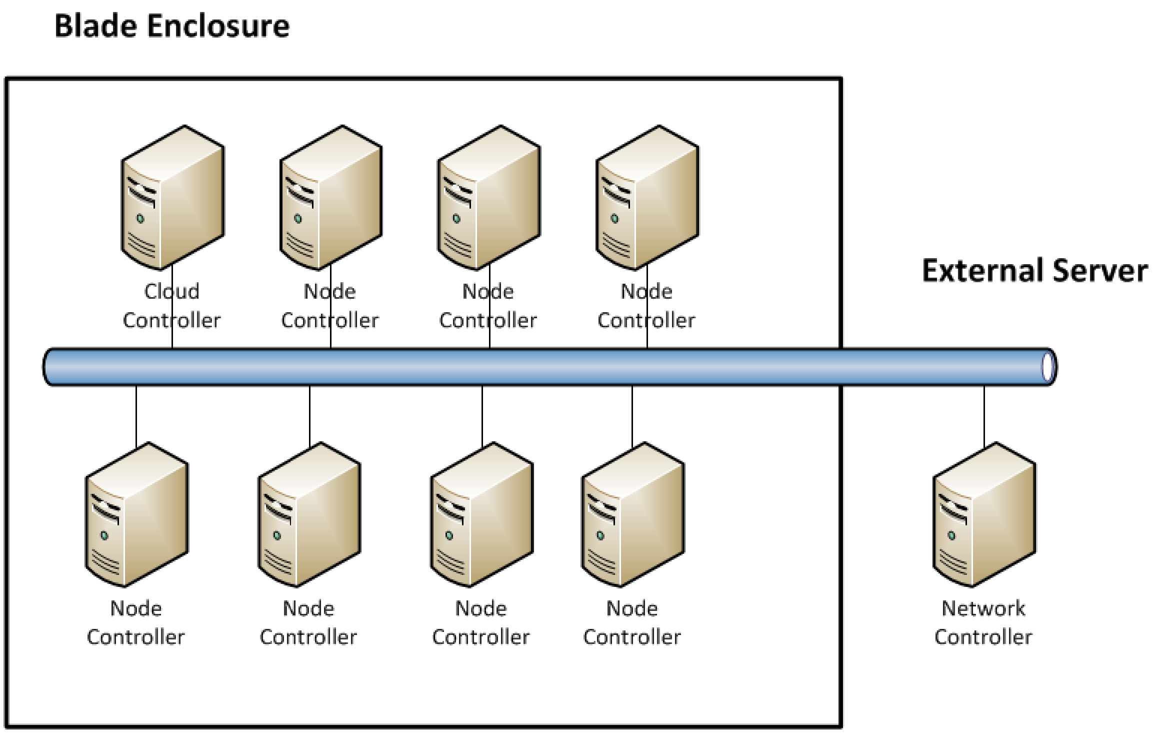


Figure 5 - HP Cloud Trial Configuration

3.1.3.1 Compute Services

Using OpenStack dashboard (a Web Application), it's possible to create Virtual Machines by selecting the between the following options:

- Size: virtual resources allocated to the VM
  - Tiny: 1 virtual cpu + 512 MB Ram
  - Small: 1 vCPU + 1 GB Ram
  - Medium: 2 vCPU + 2 GB Ram
  - Large: 4 vCPU + 4 GB Ram
  - Extra Large: 8 vCPU + 8 GB Ram
  - etc…
- Image: Operating system and basic software – HP can create images for specific requests (after technical validation)
  - Ubuntu 12.0.4 server LTS 32 and 64 bit
  - Other Linux distributions
  - Some Windows releases

Please note that for images based on Windows O.S. a check need to be done for due to licensing regulations.

3.1.3.2 Network Connectivity

The lab Cloud of HP is connected to an intranet LAN:

- The Lab network is NOT on HP Corporate network, but it's accessible from HP Corporate network through a VPN (only for HP employees);
- The Lab network supports WiFi inside HP IIC offices (only for HP employees);
- The Lab network has outbound connectivity to the internet without web proxy;

- The lab network has NO inbound connectivity from the internet (the firewall is outside HP IIC control).

Virtual machines created inside HP OpenStack cloud are by default all connected inside a virtual LAN (managed by Nova Network component). To make a VM accessible to the Lab network, an explicit operation needs to be performed from OpenStack dashboard, to assign a "public" address (public here means inside Lab network).

3.1.3.3 Advanced Networking Features

The latest OpenStack version is Folsom, with the Nova Network component, that doesn't offer great flexibility. For the time when iJOIN trials will start, the OpenStack installation will be upgraded to Grizzly release, including the advanced Neutron component for networking that will enable the following features:

- The dynamic creation of second level sub-networks connecting only a set of user selected VMs
- These second level networks are implemented by embedding OpenVSwitch [6] inside each cloud worker node (hypervisor nodes)
- The dynamic creation of gateways between second level sub-networks

The OpenVSwitch configuration contains "performance" related parameters for:

- Rate limits (outbound bandwidth for VM network interface)
- Rate bust and peak values for short periods of rate limits
- Currently no "latency" related parameters have been implemented

3.2 Joint Access and Backhaul Testbed

3.2.1 Testbed Concept

The 60 GHz frequency band provides up to 9 GHz of unlicensed bandwidth and is well-suited for wireless links that require data rates of several Gbit/s. Within iJOIN, this technology is intended for high capacity wireless backhaul on the "last mile" between the RANaaS and an iJOIN small cell (iSC). For this kind of application, the wireless channel is typically a line-of-sight channel. This allows to use a simplified transmission scheme, e.g., without channel equalization, to maximize the achievable data rate while keeping the hardware-effort at a minimum.

The demonstrator presented in this section shows the feasibility of such a system design, scaled to be used in a lab environment. It realizes a real-time transmission of binary data utilizing the 60 GHz band with a data rate of multiple Gbit/s over a short distance of approximately one meter. It consists of a transmitter and a receiver, each of which comprises a hybrid 60 GHz frontend with directional antennas and a digital baseband processor that is implemented on an FPGA (Field Programmable Gate Array) platform. The simplified system concept builds upon 1-bit data converters at the transmitter and receiver, thus enabling a low-power, low-complexity integrated circuit solution in future. The system design is scalable and not limited to the 60 GHz frequency band. For a real-life BH link for transmission over a longer distance the transmission power and antenna gains would have to be increased, yet the architecture and performance should be comparable to the downscaled demonstrator.

The objective of the joint access and backhaul testbed is to evaluate different candidate technologies (CT) in an experimental setup with a real, potentially erroneous backhaul link. For this, simulated backhaul links in a simulation of a CT, which runs on a standard PC in Matlab or C++, are replaced by the 60 GHz demonstrator, forming a hardware-in-the-loop platform as depicted in Figure 6:

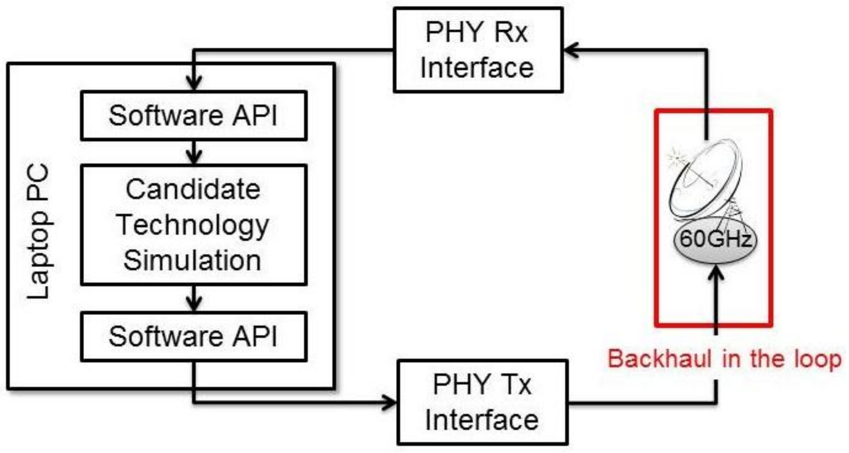


Figure 6 - Backhaul-in-the-loop platform

The interface between PC and demonstrator is provided by an Ethernet link, limiting the real time data rate between the PC and the hardware platform. To ensure easy integration of the hardware demonstrator into the software simulation of the CTs, a simple API will be provide, that handles the transmission over the 60 GHz link.

The platform is still under development therefore the exact architecture is subject to change. The setup described in this document is preliminary and shows an exemplary working implementation that will be optimized in the future.

3.2.2 Physical Layer and Digital Baseband

The considered physical layer architecture [2] shown in Figure 7 uses massive parallelization to achieve a data rate of up to 5 Gbit/s with differential QPSK (quadrature phase shift keying) modulation. A data sequence of symbol rate 3.456 GHz is split into 2 parallel data sequences corresponding to the two complex dimensions of the QPSK modulation. Each sequence carries 32 parallel (convolutional) codewords with a mother code rate of 1/3. This reduces the processing rate to only 108 MHz for each codeword, but at the cost of 64 en-/decoders.

The differential QPSK modulation enables frequency flat phase equalization at the receiver using the analog carrier recovery proposed in [3]. A residual phase-ambiguity of multiples of 90° is resolved in the digital domain. The bandwidth follows the channelization plan of the ECMA-387 standard [4] with bonding of two channels.

The data is transmitted in a packet-wise manner, with separate packets for the in-phase and quadrature-phase path. The payload of the packets is variable with a default size of 97280 bit. It is preceded with a preamble consisting of three parts: an analog training sequence (2048 bit); a digital training sequence (1024 bit) and a MAC layer signalling field (1024 bit).

Considering the latency when transmitting a single frame, the transmitter data path introduces a delay of 93 clock cycles (0.86 µs). The receiver data path introduces a delay of 424 clock cycles (3.9 µs), where the main part originates from the Viterbi decoding.

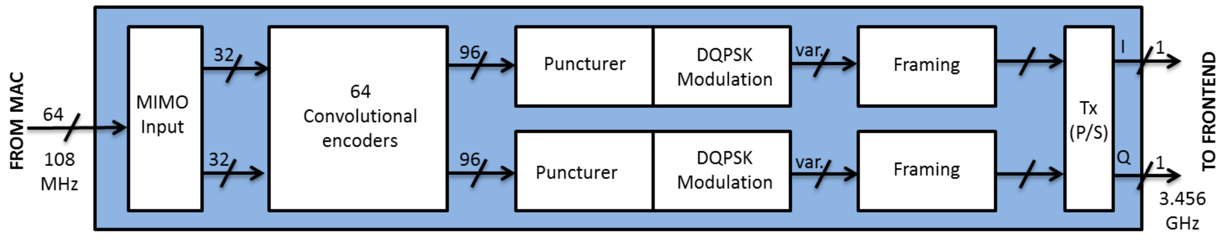
The transmitter and receiver data paths are implemented on off-the-shelf FPGA platforms, each comprising 6 Altera Stratix III and one Altera Stratix II-GX. The integrated transceiver interfaces of the latter serve as the 1-bit data converters that connect to the 60 GHz frontends. The FPGAs are clocked at 108 MHz.

Since the demonstrator uses a simple 1-bit analog-to-digital-converter, it is not possible to calculate the SNR of the 60 GHz channel. The only channel state information available is the bit error rate before or after decoding. If the distance and code rate are appropriately chosen, the bit error rate is usually zero, meaning that the demonstrator provides an error free link.

An overview of the most important parameters is given in the following table:

Characteristic	unit	value
Carrier frequency	GHz	61.56
Bandwidth	GHz	4.320
Symbol rate	GHz	3.456
Modulation		DQPSK
Coding		Convolutional, mother code rate 1/3
Preamble length	Symbols	4094
Payload length	Symbols	97280
Data rate @ code rate ¾	Gbit/s	5
Output power	dBm	10
Maximum range	m	~1
Processing latency	µs	4.76

Table 1: Baseband parameters of the 60 GHz demonstrator



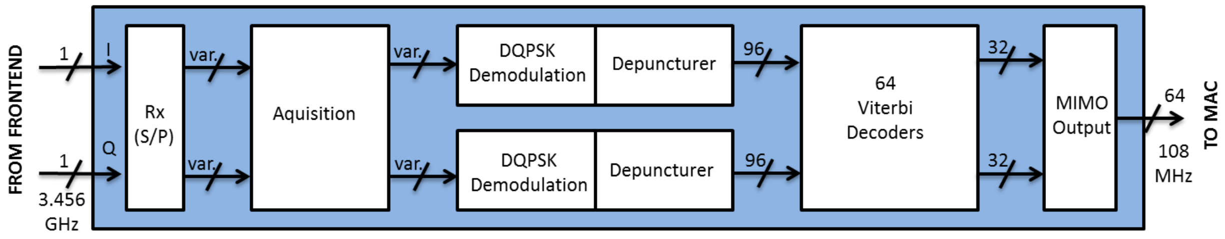


Figure 7 - Digital Baseband Architecture of Transmitter (top) and Receiver (bottom) of the 60 GHz Demonstrator

3.2.3 60 GHz Frontend and Antennas

The RF-Frontends have been fabricated from commercial of the shelf bare die components. Both the transmitter and the receiver frontend have a direct mixing, zero-IF architecture, as shown in Figure 8. The baseband bandwidth of the frontends ranges from 0 to 3 GHz.

The upper limit follows from the intermediate frequency (IF) in/-outputs of the mixers. A 7.5 GHz local oscillator (LO) signal has to be supplied from an external source, which is then quadrupled and fed into sub-harmonic mixers. The output power of the transmitter is 10 dBm under default operation conditions. The receiver uses the same mixer architecture and the same LO supply chain as the transmitter.

Since no strong blocking or interference signals are expected for a line-of-sight transmission in the 60 GHz frequency band, the received signal can be filtered after being amplified by an low-noise amplifier. This ensures a noise figure below 7 dB. The overall power consumption for the transmitter is below 900 mW and less than 600 mW for the receiver. As antennas two directional horn antennas are used.

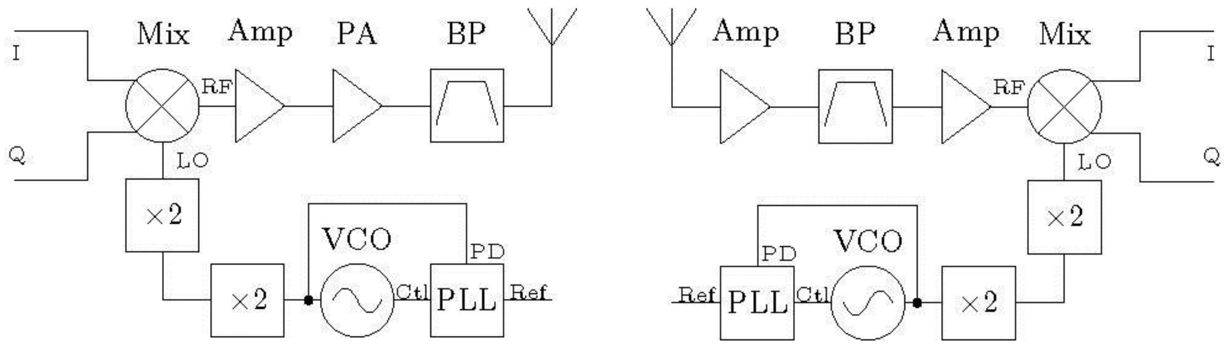


Figure 8 - Analog Frontend of the Transmitter (left) and Receiver (right) of the 60 GHz Demonstrator

3.3 SDN Testbed

3.3.1 Motivation

The iJOIN network wide control mechanisms follow a Software Defined Networking (SDN) model, in which a controller entity - the iJOIN network controller (iNC) - is in charge of orchestrating and configuring the different logical network entities so the desired functionality is achieved in the most efficient way, while benefiting from a joint backhaul and RAN optimized design.

In order to be able to prototype, showcase and partially assess the performance of the design mechanisms, a specialized SDN-based testbed will be deployed.

3.3.2 Testbed Description

The following figure shows an example of the layout of the testbed:

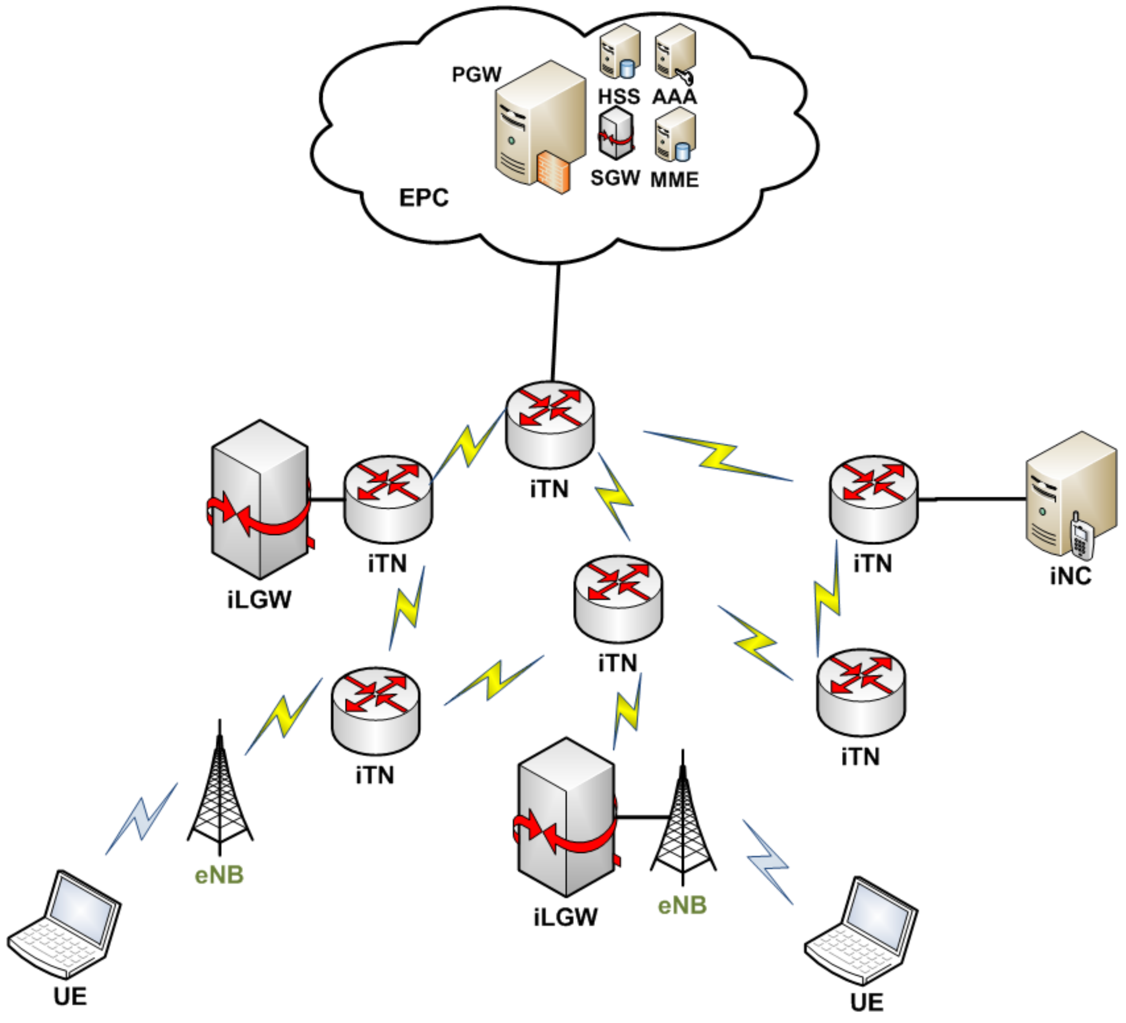


Figure 9 - SDN testbed layout

The layout basically comprises several iJOIN transport nodes (iTNs) - some of them with a collocated iJOIN local gateway (iLGW) functionality, some eNBs (some of which can also be virtual eNBs), at least a couple of user equipments (UEs), an emulated evolved packet core (EPC) and an iJOIN network controller (iNC).

Every iJOIN network logical entity (i.e., iTNs, iLGWs and the iNC) will be implemented on Linux (most likely using a kernel 3.x). The EPC functionality will be partially emulated depending on the final requirements of the demonstrator (i.e., the mechanisms that are actually designed, implemented and showcased). The internal backhaul links will be implemented using IEEE 802.11 and/or 802.3 technologies, as the focus of the testbed is on the network control, not on the actual wireless technologies. Similarly, the eNBs will be emulated, most likely using IEEE 802.11 access points.

Regarding the hardware, laptops will be used for the UEs, a regular PC/laptop for the iNC, and for the iTNs/iLGWs/eNBs, either laptops, regular PCs or even small routers running Linux (such as the Linksys WRT54GL) will be used.

3.3.3 Demonstrator Capabilities

The testbed will have a size which is sufficient to prototype and showcase the main iJOIN network control functionalities, which are: distributed mobility and anchoring, smart path management & routing, and network wide energy optimizations.

The following figure shows an example of potential demonstrator, in which a UE is initially attached to a eNB and has some traffic (red dashed line) locally anchored at the iLGW on the left side of the network (leftmost subfigure). Then, the network decides to move the UE due to energy optimization reasons (in order to be able to switch off some network equipment). Once the UE has moved, existing traffic is still anchored at the original iLGW (middle subfigure), but new sessions (e.g., green dashed line) are anchored at a different iLGW (the one at the bottom), as shown in the rightmost subfigure. The path between the UE and the respective anchors is optimally computed by the path & routing iJOIN mechanisms.

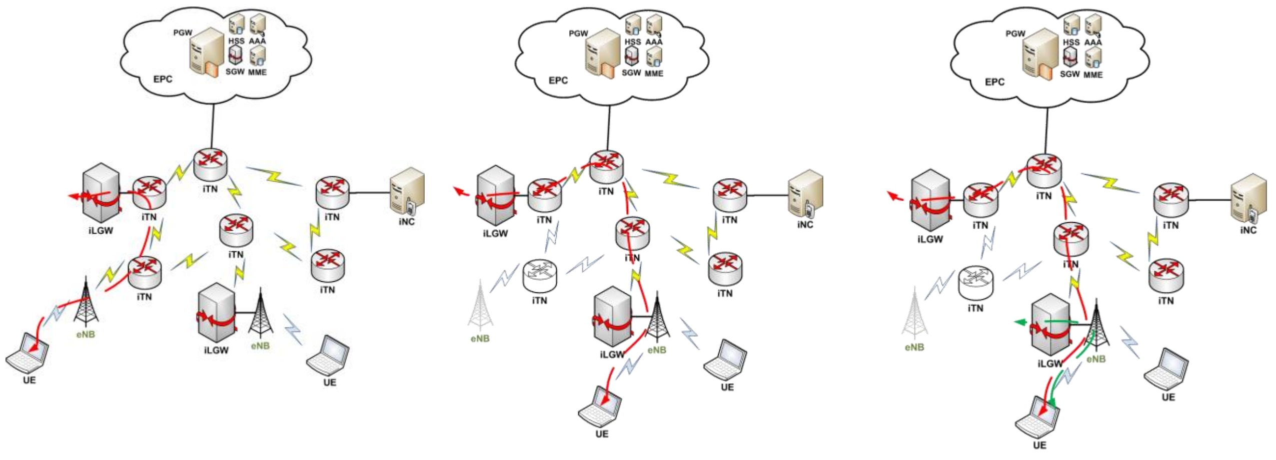


Figure 10. Example of Network Control Demonstration

4 Implementation Guidelines

4.1 RANaaS Testbed

4.1.1 Testing iJOIN RANaaS in the Cloud

As reported in Section 3.1.1, RANaaS testbed is built-up of two conceptual layers: an IaaS platform and a set of RAN Services that are implemented as computer programs. The IaaS platform provides the computing resources; RAN programs provide the logic of the RAN Services.

RAN Services are actually obtained by the integration of several RAN Service Elements:

$$\text{RAN Service} = \sum (\text{VM} + \text{RAN Program})_i.$$

Technically speaking, a RAN Service Element is obtained by preparing a VM image (see below) that contains a RAN Program: when a RAN Service Element is needed, a VM containing the corresponding RAN Program is started. Starting all the RAN Service Elements in a service, the required service is provided. This implementation takes advantage of a feature implemented by the OpenStack IaaS platform: the possibility to create VM images, save them on a catalogue and reuse them for starting VMs.

A VM image, or image for short, is a file containing a Virtual Machine's contents and can be considered as the boot disk counterpart of a physical computer. When a VM in the cloud is going to be started, a VM image needs to be selected and the VM boots from it. A running VM behaves the same way a physical computer that boots from a disk with the same contents of the VM image.

Generally speaking, a VM image must contain an Operating System (e.g. a Linux distribution) for booting but can also contain some application program for implementing a specific function.

For example, it is possible to create a VM image with a database installed on it; when a VM is started using such an image, it actually implements a machine running a database.

In iJOIN, this feature is used for creating RAN Services: the idea is that for each function in a service, a VM image is created with both an Operating System and a RAN program installed on it. VM images are then stored in the image catalogue of the IaaS platform and used for booting VMs that provide the RAN Service installed on them.

The following picture summarizes the steps of the process for loading and use a VM image:

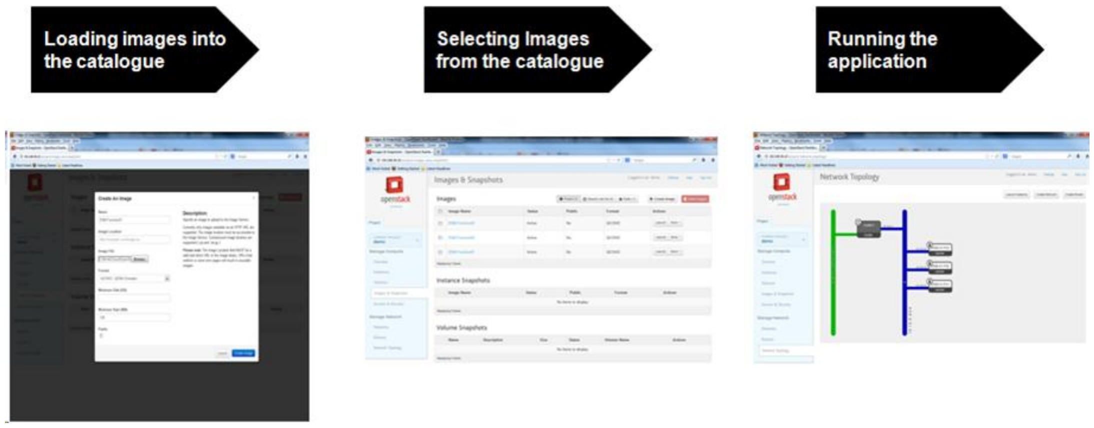

Loading images into the catalogue
Selecting Images from the catalogue
Running the application

Figure 11- Creating and Using a RAN Services

The picture above shows how to load and run a VM using Horizon, a web-based user interface provided by OpenStack but it should be noted that these functions can also be obtained using a 'programmable interface'. OpenStack provides both an Application Programming Interface (API) and a Command Line Interface (CLI) that make it possible to automate procedures. In particular, the possibility to automatically start and stop VMs (and, consequently, RAN Services) is the key point of the RANaaS concept.

Some considerations need to be done on the implementation of RAN services. It's fundamental to identify which RAN functions are to be provided as services and implement corresponding computer programs with the related logic/algorithm. Every single program is deployed on a VM therefore if it requires additional software elements for running (e.g. runtime libraries, interpreters, etc.). Furthermore, these components need to be installed on the same VM image. A RAN program will probably need to communicate with ‘external entities’ like iSCs, other RAN programs, etc. For the RANaaS testbed, the communication can be implemented using some communication middleware (e.g. XML RPC [7]).

In order to parallelize function execution, it would be necessary to split monolithic programs currently used for testing purposes (e.g. Matlab programs) into several ‘traditional computer programs’ implemented in some computing language of choice (i.e. C, C++, etc.) which may be invoked separately without binding from other similar software pieces.

4.1.2 The Testing Environment

A VM can run independently on a cloud. The minimal requirement is to have a hypervisor (installed on a physical computer) that runs the VM simulating the real hardware. From this perspective, RAN Service Elements and RAN Services could be prepared and tested on PCs equipped with a hypervisor. Then, they could be integrated with the IaaS platform that provides mechanisms for creating (i.e. instantiating) several RAN Service copies in a cloud computing environment.

One option is to use KVM hypervisor [8] on Linux for creating RAN Services. When the implementation is complete, iJOIN partners provide HP with the corresponding VM images for the integration with the cloud IaaS platform. As an alternative, Hewlett Packard can provide an **OpenStack Training Appliance** (OTA) that implements, on a single single node, an “openstack cloud in a box” with the same interface as the one in HP IIC lab.

OTA implements an “All-in-One OpenStack Node” meaning that all the OpenStack services run in a single Virtual Machine. It is a very basic deployment that can be used for developing software, for functional testing and for simple demonstrations. Please note that production environments, but also moderate complex demonstration environments, are composed of several physical machines (also named as nodes) running a subset of OpenStack services each.

The appliance runs OpenStack without disturbing the software already installed on the hosting PC that, to some extent, will ‘perceive’ the appliance as a different and autonomous machine. Users will be able to connect to the appliance using a web browser or a terminal application like PuTTY from the desktop of a PC. In reality you will run a VM Player on your PC. On top of the VM Player, you will start the appliance with an Operating System (Ubuntu Server 12.04). On the Operating System, there’s OpenStack that implements the cloud. When you start a Virtual Machine in the cloud, you actually obtain a kvm Virtual Machine Player running on the appliance’s operating system that also runs OpenStack. Kvm runs the virtual machine in the cloud. This scenario is summarized by the following picture:

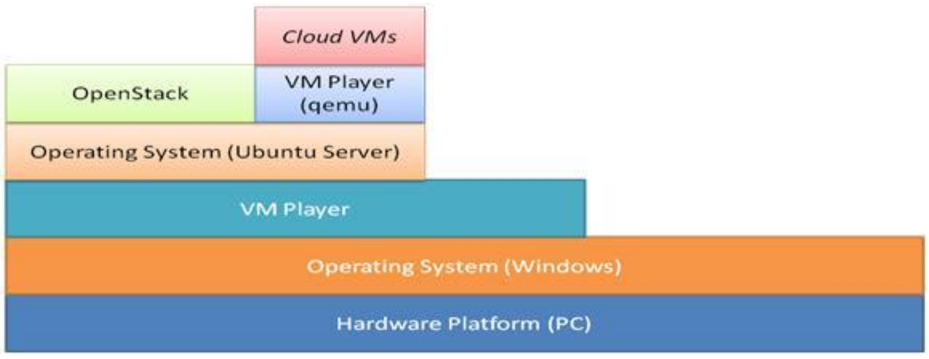


Figure 12 - OpenStack Training Appliance

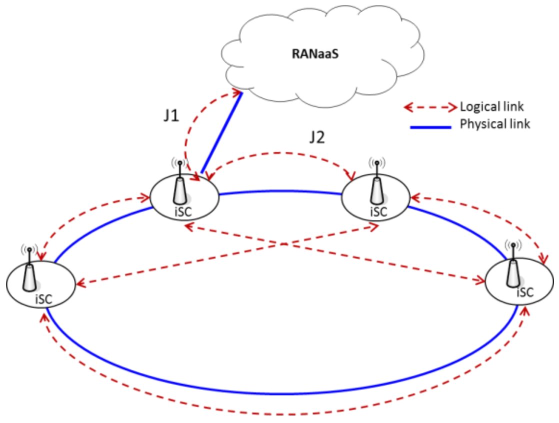
It should be noticed that there are several layers on top of PC hardware and it is not advisable to use deployments of this sort in a production installation. On the other hand, this OpenStack deployment can be used for gaining some confidence before facing complex installations and/or advanced functions.

4.2 Joint access and Backhaul Testbed

As already mentioned in Section 3.2, the testbed is still in development and could be adjusted to certain needs of CTs that will be implemented. The following sections give an example on how an implementation of the demonstrator as a backhaul-in-the-loop could be implemented with a CT.

4.2.1 Application Example: CT 2.1 In-Network Processing

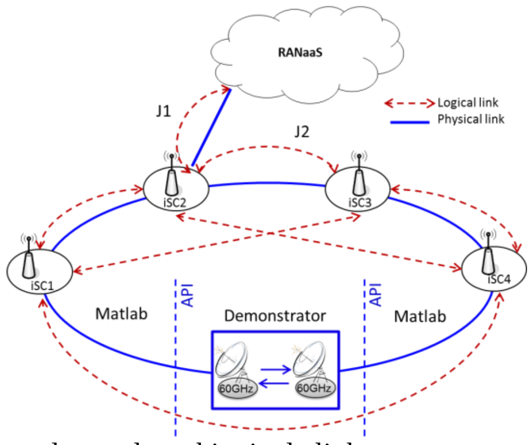
We use CT 2.1 as an example how the 60 GHz demonstrator can be used in CT simulations:



In CT 2.1, several iSCs exchange information about UEs' received signals and one of them forwards information to the RANaaS. The iSCs then try to find a consensus on the content of the UE transmissions. The access link is not depicted above to enhance clarity. For this example, we assume that all J1 and J2 links are based on 60 GHz links.

The CT is evaluated by Matlab simulations, where the backhaul is simulated as ideal. As an example, assume that iSC1 needs to exchange a vector of bits `x` with iSC4. The code for the bit vector `y` that iSC4 receives would be

```
y=x;
```



The demonstrator now could be used to replace this single link:

A simple API will be used to include the demonstrator into the transmission. The code to transmit the vector `x` would be

```
y=sendVia60GHz(x);
```

The function `sendVia60GHz` sends the vector to the demonstrator, places it into a MAC frame which is filled up with dummy bits, transmits it via the 60 GHz link and sends it back to Matlab, potentially introducing bit errors if the channel quality is low. The demonstrator will continuously continue to transmit dummy bits in times between function calls. The received vector can now be further processed in Matlab.

A simple dummy code for testing without the demonstrator would be:

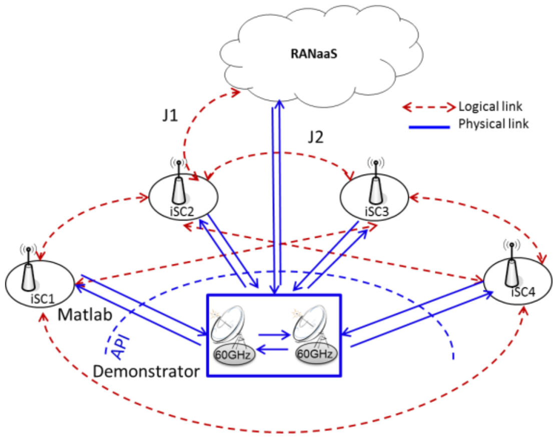
```
function y = sendVia60GHz(x)

      y=x;

end
```

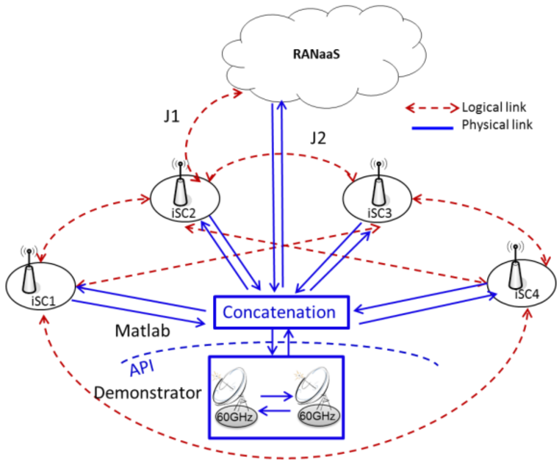
Alternatively, the dummy function could introduce bit errors to simulate an erroneous transmission. It could also include a sleep time, if the real delay of the demonstrator needed to be simulated.

For a demonstration of the CT, it is furthermore beneficial, to not only replace one link with the demonstrator, but several:



As there is only one demonstrator, all links face the same channel during transmission.

In CT2.1, information is processed iteratively and in each iteration step, information is exchanged between iSCs. Each vector `x` to be exchanged contains usually only a few bits. Since every call of `sendVia60GHz` introduces a certain delay due to setup times in the hardware, it would be more efficient to send multiple information vectors over the link in one call. Instead of calling `sendVia60GHz` for each exchange between two iSCs, all information to be exchanged in one iteration step could be concatenated and sent with one call, reducing the delay and also simulating simultaneous transmissions.



The function `sendVia60GHz` transmits only raw bit data, so the Matlab simulation would be responsible for concatenating the bit vectors before transmission and separating them according to destination after reception (i.e., "logical link" stays in Matlab). Alternatively, the function `sendVia60GHz` could be encapsulated to provide such functionality. Furthermore, the function could also return additional information, e.g. the bit error rate as estimated by the hardware during the frame transmission (the frame might be much longer than the original bit vector `x`).

It would also be possible to provide a function that returns the current estimated BER on the demonstrator link. However, this value would not change much, since the channel of the demonstrator is very static. To demonstrate the feasibility of the CT for different channel conditions, the link distance or code rate of the demonstrator could be increased.

4.2.2 Requirements for Development and Evaluation Parameters

In order to ensure an easy integration of the candidate CTs, the following information on each candidate CT is required for developing the testbed:

- Number of 60 GHz links to be replaced by demonstrator
- Approximate number of bits per transmission/ function call
- Simultaneous transmissions/concatenation possible?
- Delay requirements on demonstrator (if any)

The following parameters of the testbed can be evaluated, either during operation as backhaul-in-the-loop or in separate measurements:

- Bit error rate
- Code rate
- Maximum data rate: payload (excluding signalling overhead and coding) and total
- Energy consumption
- Delay
- Link budget

4.3 SDN Testbed

The SDN testbed will be based on OpenFlow as southbound protocol, using the reference Linux User-space and Kernel-space Switch available at http://www.openflow.org/. For the controller, different options would be analyzed later in the project, but a good candidate is POX [8]

4.3.1 Application example: CT 4.3 Joint Path Management control

In order to provide an example of how the SDN-based network control demonstrator can be used for CT prototyping and performance assessment, we use CT 4.3 as an example.

In CT 4.3, the network continuously monitors the status of the backhaul and the RAN and dynamically computes the best path from a given iSC and the exit point in the backhaul towards the anchoring entity (which might even be a backhaul node, if traffic offloaded at an iLGW).

Current OpenFlow specification provides some mechanisms to report on the status of the network, but it is likely that some additional extensions will be implemented by iJOIN. Based on these periodic reports (from iTNs, iSCs and iLGWs), the iNC is able to keep an up-to-date view of the network (topology and load), which is used to compute the best possible path between a given pair of ends (from the iSC to the anchoring entity).

The Joint Path Management control algorithm runs on the iNC, resulting in a set of forwarding rules to be configured and installed on the different involved nodes in the path. This forwarding state, which is as this state foreseen to be configured based QinQ is then configured on the nodes, also using OpenFlow. Note that a first version of the demonstrator could make use of outband OpenFlow signalling, meaning that there exists a direct connection between the iNC and the rest of the network entities, used exclusively to run the OpenFlow protocol (reports and configuration). This is planned to be replaced by inband signalling at a later stage of the project.

The OpenFlow protocol will also allow monitoring in real time of the paths computed by CT4.3.

5 Summary and Conclusion

The main purpose of WP6 is to provide an environment for evaluating, testing and demonstrating technical solutions for the implementation of the iJOIN system architecture.

In this document, the partners of the consortium provide the architecture of the testing enviroment that is built-up of components aimed to simulate elements and/or functions of the iJOIN reference architecture.

At the time of writing, the testing evironment is built-up of the following testbed components:

- Radio Access Network as a Service Testbed
- Joint Access and Backhaul Testbed
- Software Defined Network Testbed

It is worth noticing that the testing environment can be considered as a "development platform" providing basic services identified as fundamental features needed for supporting the iJOIN architecture. This approach results in a high flexible environment that can be used for testing several competing solutions.

This document describes how the components are implemented, the services they offer, the relationships between them and how they are expected to be used in the project context.

The future work would be to investigate which of the candidate technologies would be demonstrated over these test platforms. Different work packages have already started investigation of the technologies which are suitable for demonstration. Decoder implementation from WP2 is under consideration over RANaaS testbed. In-network processing and intercell-interference cancellation are being considered for demonstration over joint access and backhaul testbed. Joint path management control is a promising candidate for demonstration over SDN testbed.

Acknowledgements and Disclaimer

This work was partially funded by the European Commission within the 7th Framework Program in the context of the ICT project iJOIN (Grant Agreement No. 317941). The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the iJOIN project or the European Commission.

References

[1] C. J. Bernardos, A. de la Oliva, J. C. Zuniga, and T. Melia. Applicability Statement on Link Layer implementation/Logical Interface over Multiple Physical Interfaces. Internet Engineering Task Force, draft-bernardos-netext-ll-statement-01.txt (work-in-progress), March 2010.

[2] G. Fettweis, F. Guderian and S. Krone, “Entering the path towards terabit/s wireless links”, Design, Automation Test in Europe Conference Exhibition (DATE), March 2011.

[3] A. C. Ulusoy, G. Liu, M. Peter, R. Felbecker, H. Y. Abdine, and H. Schumacher, “A BPSK/QPSK Receiver Architecture Suitable for Low-Cost Ultra-High Rate 60 GHz Wireless Communications,” Proceedings of the European Microwave Conference (EuMC10), September 2010.

[4] Standard ECMA-387: High Rate 60 GHz PHY, MAC and HDMI PAL, 2008.

[5] OpenStack Foundation http://www.openstack.org/foundation/companies/

[6] OpenVSwitch site http://openvswitch.org

[7] XML RPC site http://xmlrpc-c.sourceforge.net/doc/libxmlrpc.html

[8] NOX and POX control platforms for Software Defined Networks http://www.noxrepo.org/