NOKIA

Nokia Networks

MEC
Mobile Edge Computing

Aalto University, T-110.5130 Mobile Systems Programming

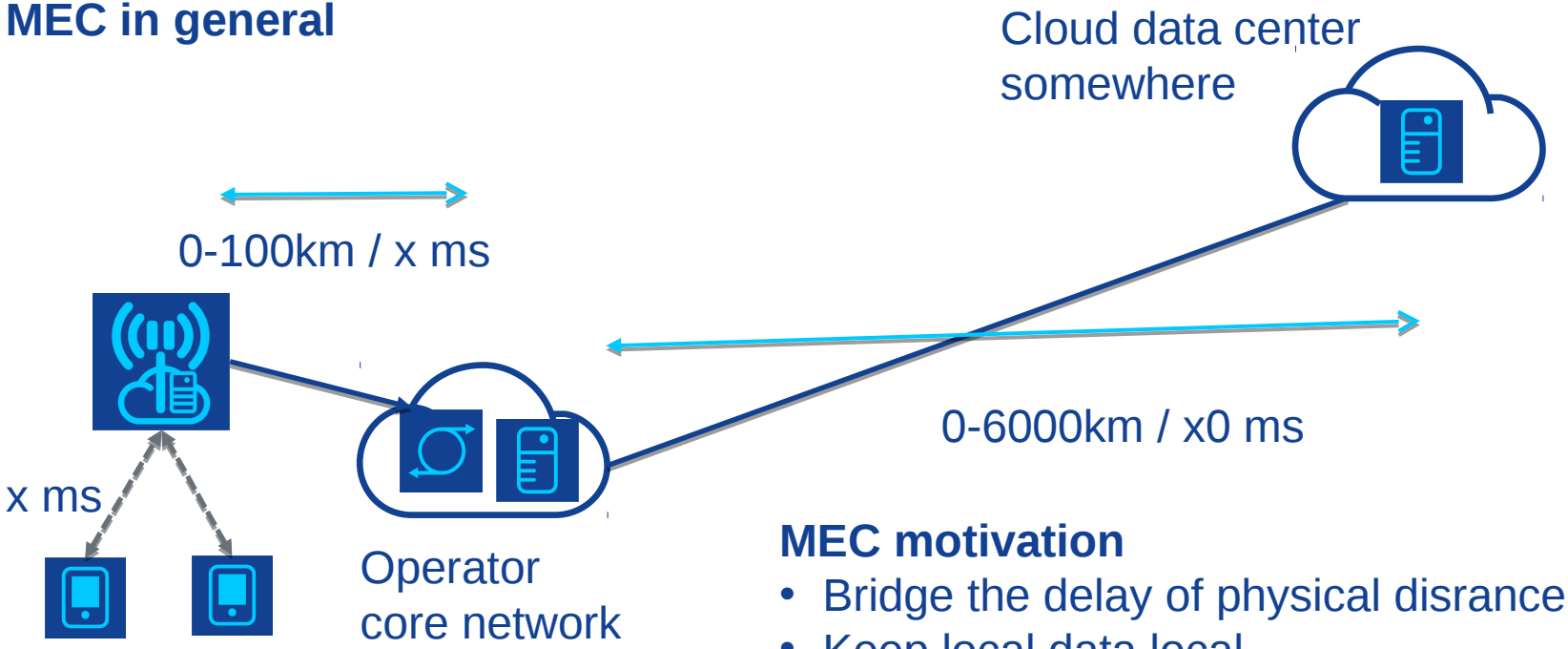
- Timo Knuutila (timo.knuutila@nsn.com), Petteri Pöyhönen (petteri.poyhonen@nsn.com)

- 14-01-2015

Agenda

- MEC in general
- Nokia approach to MEC
- Sample applications
- Network environment for the course
- MEC Programming in RACS environment

MEC in general



MEC motivation

- Bridge the delay of physical disrance
- Keep local data local
- Lower long distance bandwidth requirement

Nokia approach to MEC

- Nokia Liquid Apps concept
- AppFactory
- Application and software management
- Samples

Liquid Apps: Why at the mobile edge ?

- Content at the place where people connect
- Process data at its source before it gets big
- Immediately capture value from real-time network data to contextualize applications
- Maximize speed and interactivity from being as close as you can get to the customer

- Seamless mobile network integration
- Telco-grade robustness and security
- State-of-the-art middleware for highly distributed application environments

Radio Access Cloud Server (RACS)

Optimized for standard 3GPP functions

Built around the telco lifecycle

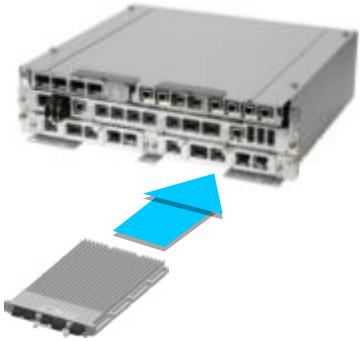
Data in/data out functionality

Signal processing

Fully programmable

Datacenter-like flexibility

Computing, storage and contextual data extraction



Edge computing

Nokia approach to MEC

- Nokia Liquid Apps concept
- AppFactory
- Application and software management
- Samples

AppFactory

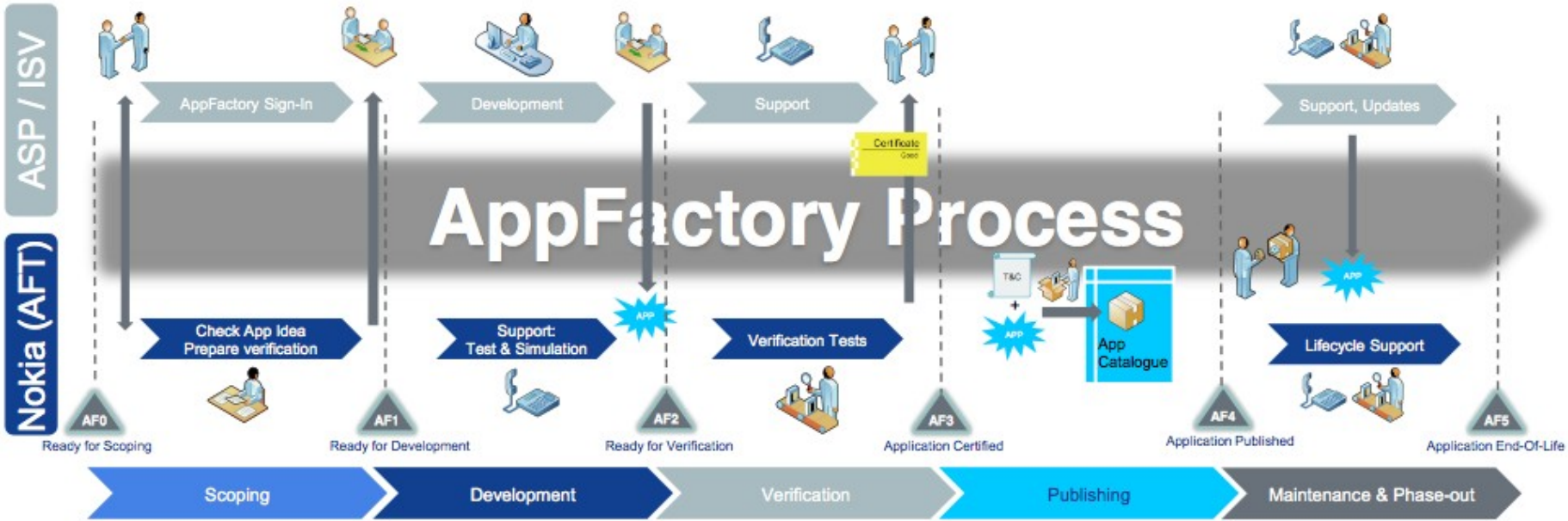
- AppFactory process for 3rd parties to create LiquidApps running in RACS environment.
- Decouples the applications and the RACS platform releases.
  - Different life cycles according to different factors.
- Well defined process ensures the SW quality both from application developer and Nokia point of view.

AppFactory

- Each application uses the same set of APIs according to their needs.
  - Additionally, there could be application specific APIs for out of band communication for instance.
- AppFactory supports 3rd party developers through the application development process and provides testing facilities.
  - No need for 3rd parties to invest own testing infra.

AppFactory Process
ASP/ISV engagement

AFT AppFactory Team
ASP Application Service Provider
ISV Independent Software Vendor
AST Application Screening Team
T&C Terms and Conditions



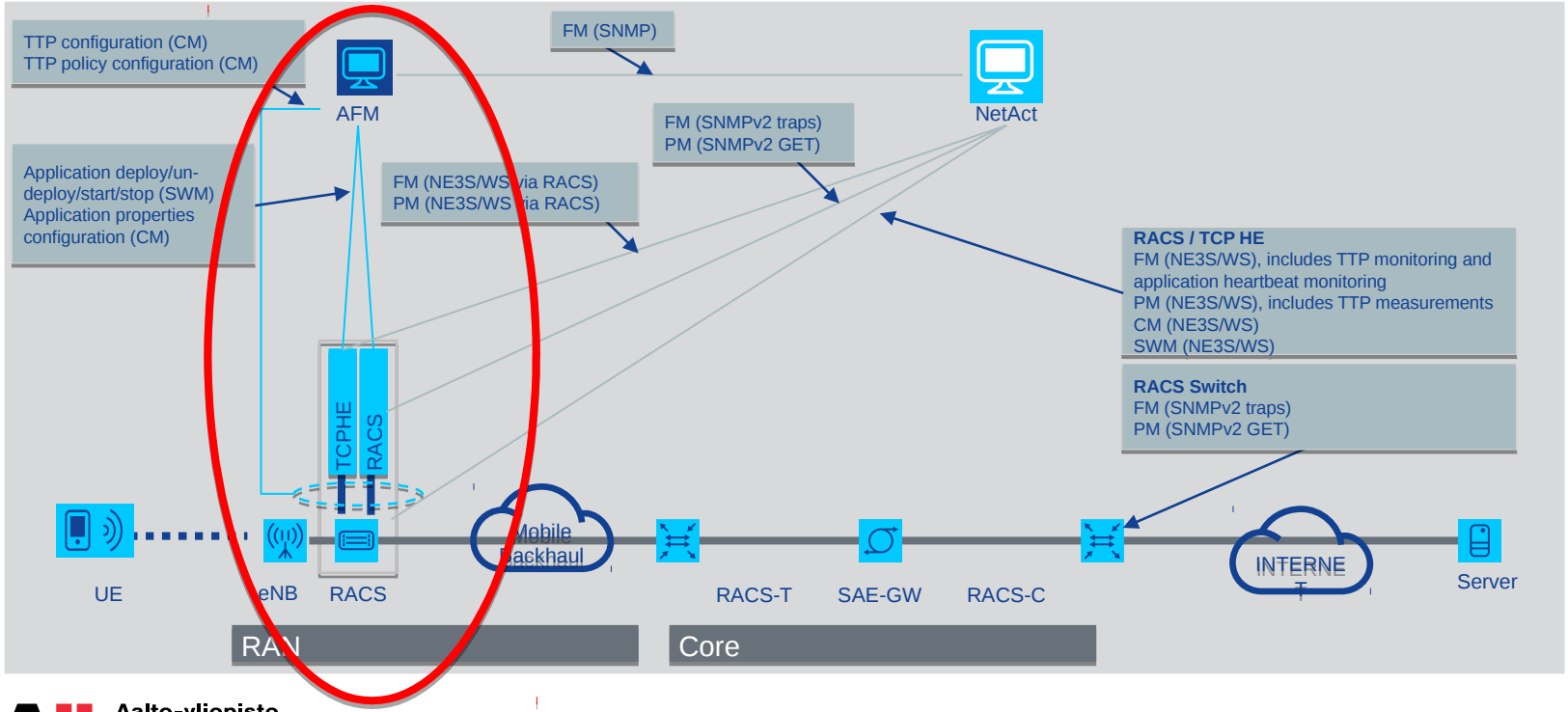
Nokia approach to MEC

- Nokia Liquid Apps concept
- AppFactory
- Application and software management
- Samples

OSS for Liquid Applications

FM= Fault Mgmt
PM= Perf. Mgmt

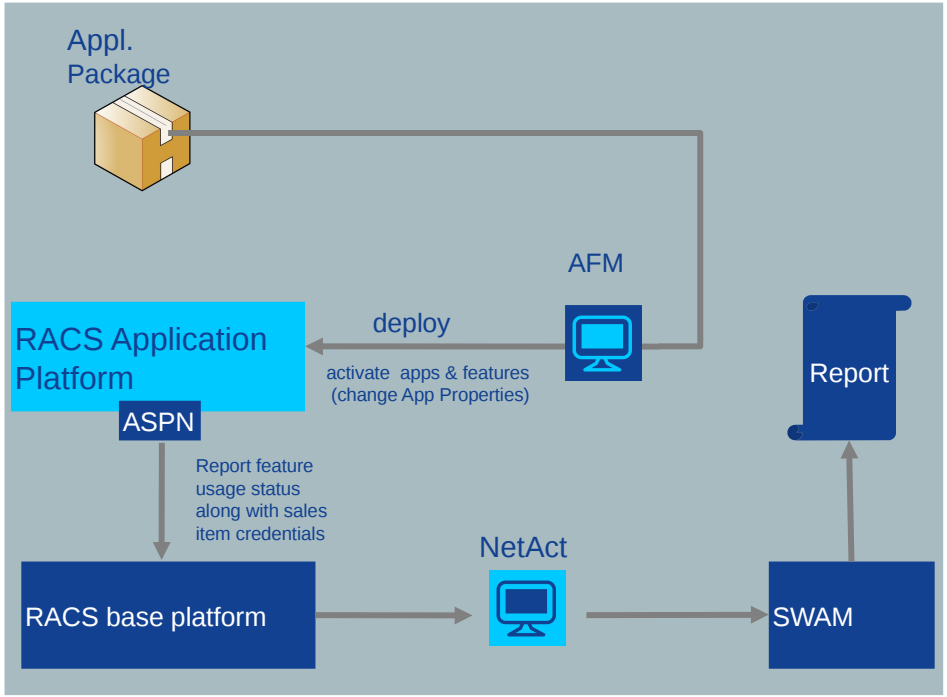
AFM= Application Framework Mgmt



SWAM Integration for Liquid Applications with AFM

Automated Charging of Application deployment

- Centralized Software Asset Management for RACS with NetAct License Manager
- Asset Reporting of Applications deployed in RACS throughout the network
- Activated feature list, feature ID and feature name
- Deployed Applications current count
- Delta count
- Reporting and report forwarding



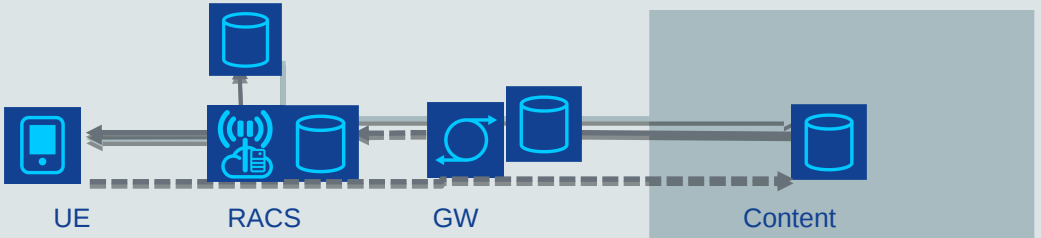
SWAM= SW Asset Mgmt

Nokia approach to MEC

- Nokia Liquid Apps concept
- AppFactory
- Application and software management
- Sample apps

Animated

Acceleration – application area overview



Large object delivery with **Content Extender** (optional Core caching)

Small object delivery with **Site Accelerator** through eNB edge caching or local break-out

Radio **throughput** guidance and real time **positioning** for service optimization

Network environment for the course

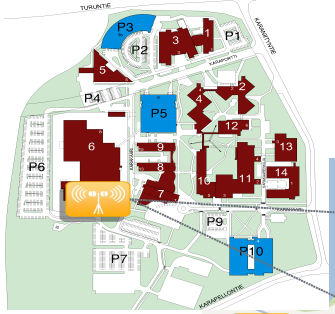
- NetLeap Network
- Emulated network

NetLeap / Network environment for the course

- Joint initiative with Nokia Networks and Aalto IT.
- Network for research and development use in mobile environment. Users get unlimited usage of LTE capacity and features for application development and testing.
- The network uses 2.6GHz frequency in Otaniemi. Dedicated indoor coverage is in OIH, Startup Sauna and Computer Science Building.
- Using this LTE network requires special SIM-cards that can be obtained from Aalto IT.

NetLeap Network

Karaportti Campus SEC



Cloud Edge Computing (RACS)

Innovation lab Espoo

Network management

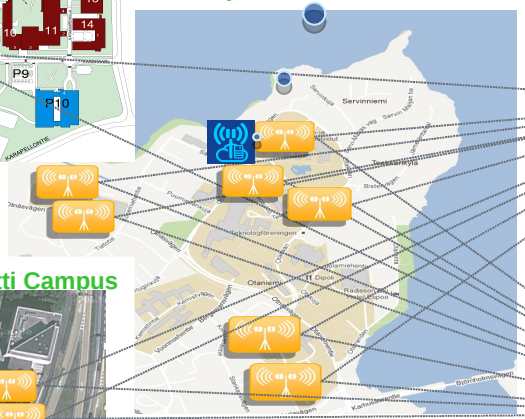
UFM

NetAct

CAM

MME pool

Netleap network in Otaniemi campus



Netleap Cloud OSR Openstack MME & GW

Säterinportti Campus

2nd Core Site OSR Openstack MME & GW

vCloud IMS/TAS*

EETelcoCloud

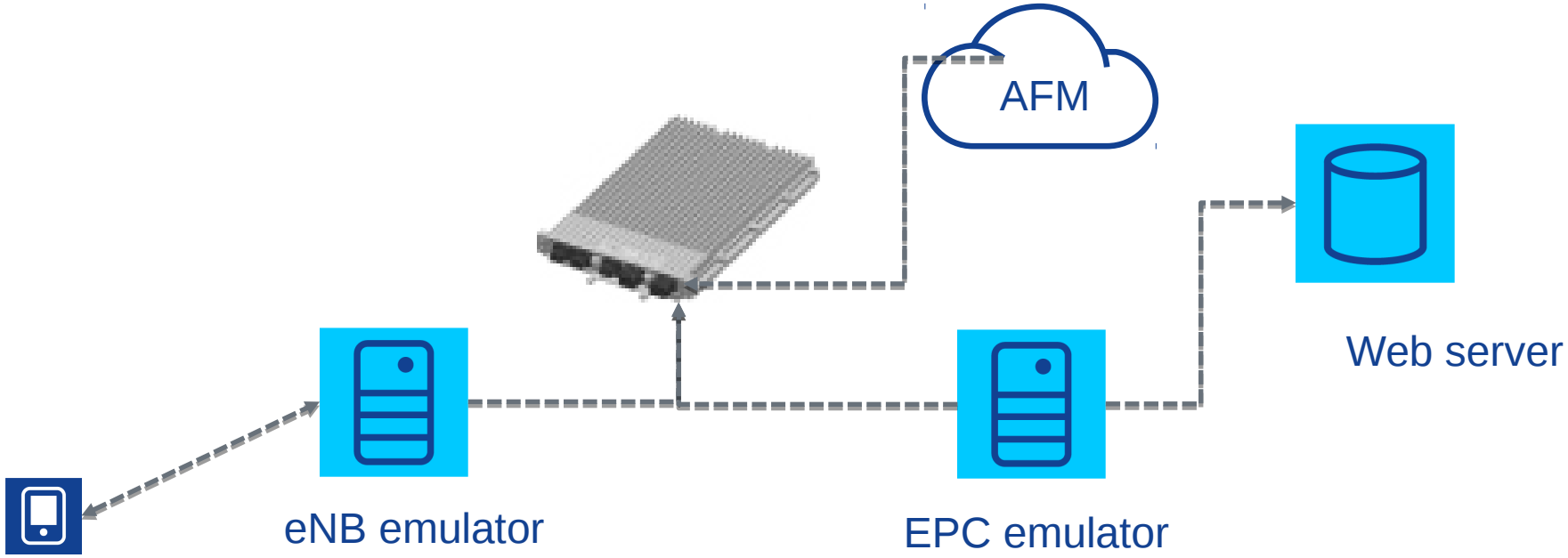
*IMS/TASQ3 2014

NOKIA

Network environment for the course

- NetLeap Network
- Emulated network

Emulated mobile network



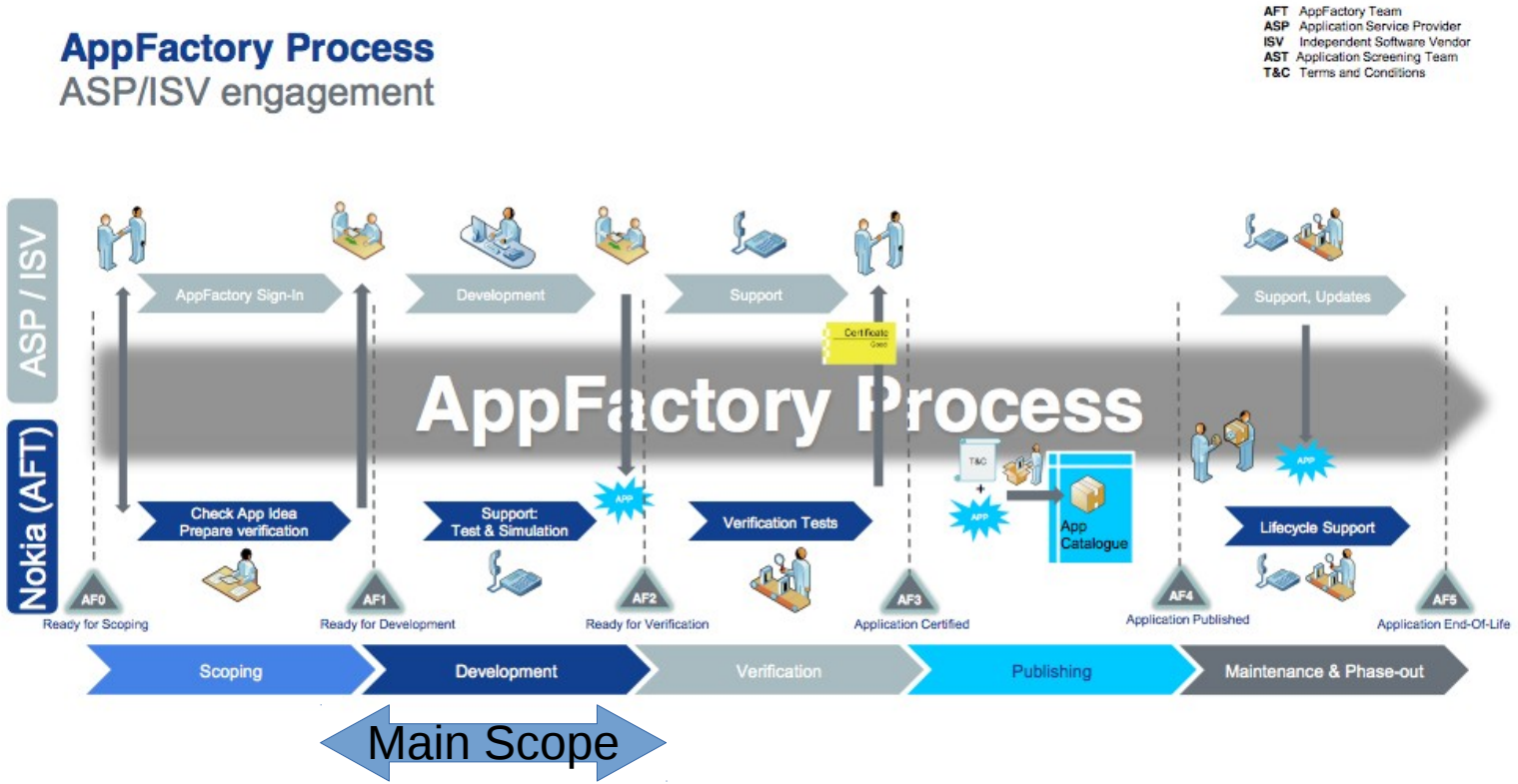
Agenda

- MEC in general
- Nokia approach to MEC
- Sample applications
- Network environment for the course
- MEC Programming in RACS environment

MEC Programming in RACS environment

- Work Scope in AppFactory Process
- RACS Application Overview
- RACS Services
- Application Types
- User Plane Service
- Application Integration
- RACS Application Specifics
- Development Process
- VM Development Environment
- Application VM
- Development Support

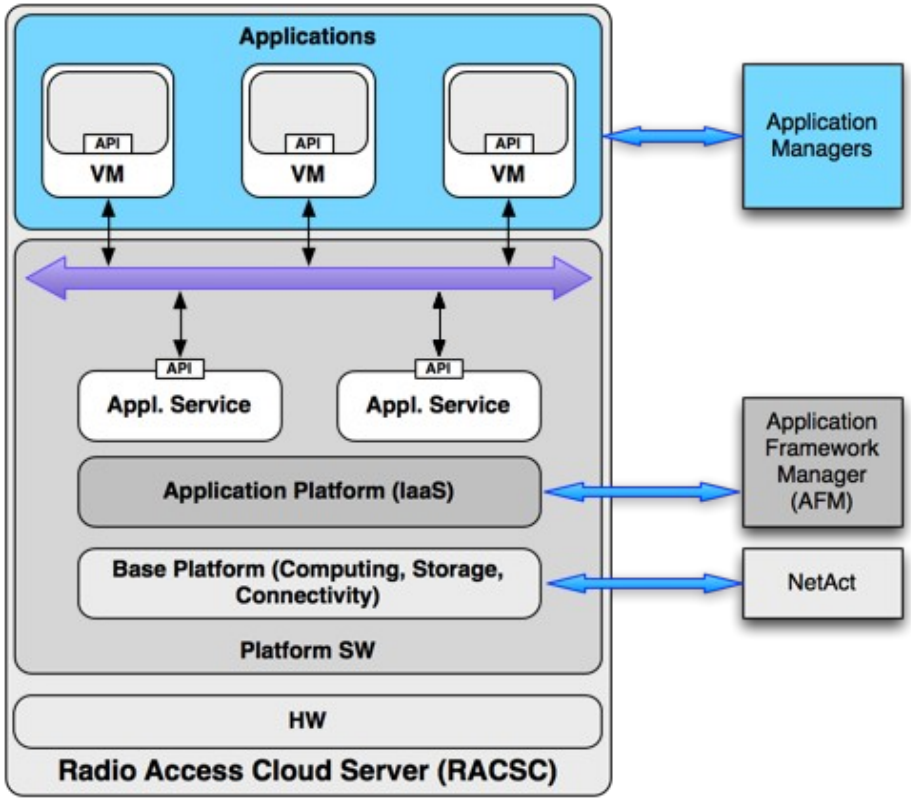
Work Scope in AppFactory Process



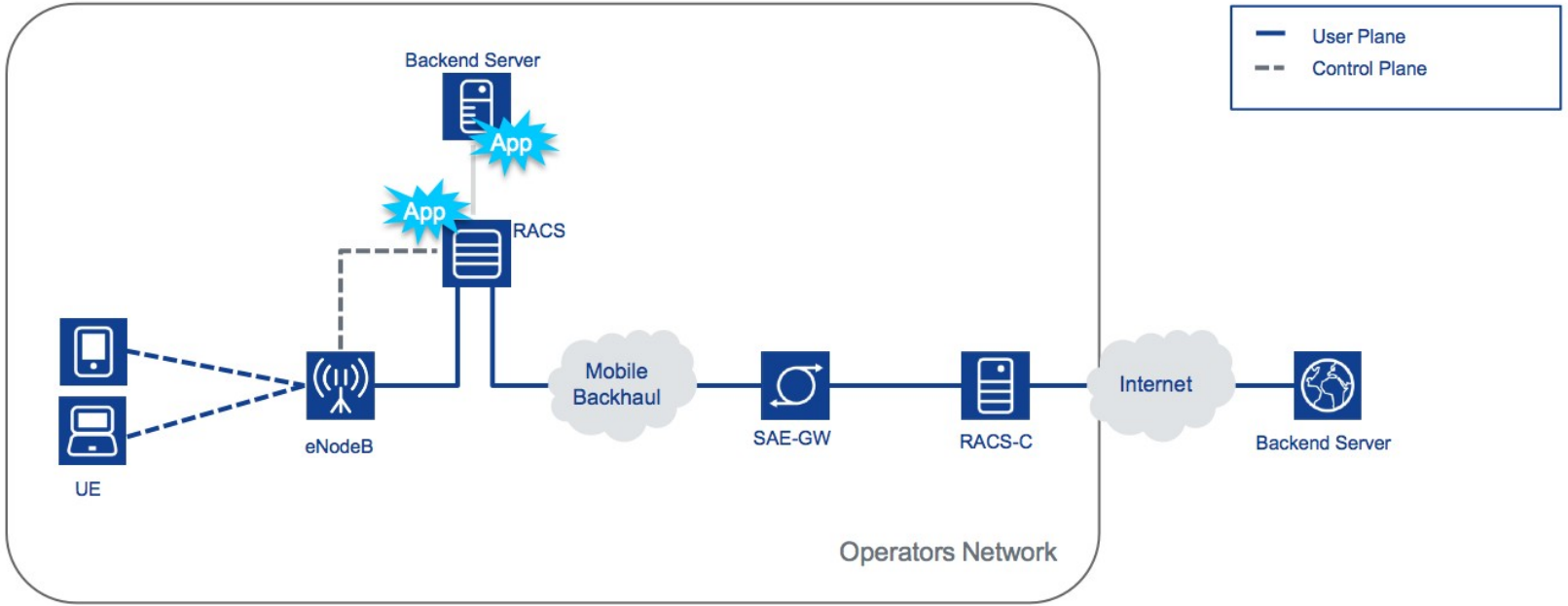
MEC Programming in RACS environment

- Work Scope in AppFactory Process
- RACS Application Overview
- RACS Services
- Application Types
- User Plane Service
- Application Integration
- RACS Application Specifics
- Development Process
- VM Development Environment
- Application VM
- Development Support

RACS Application Overview



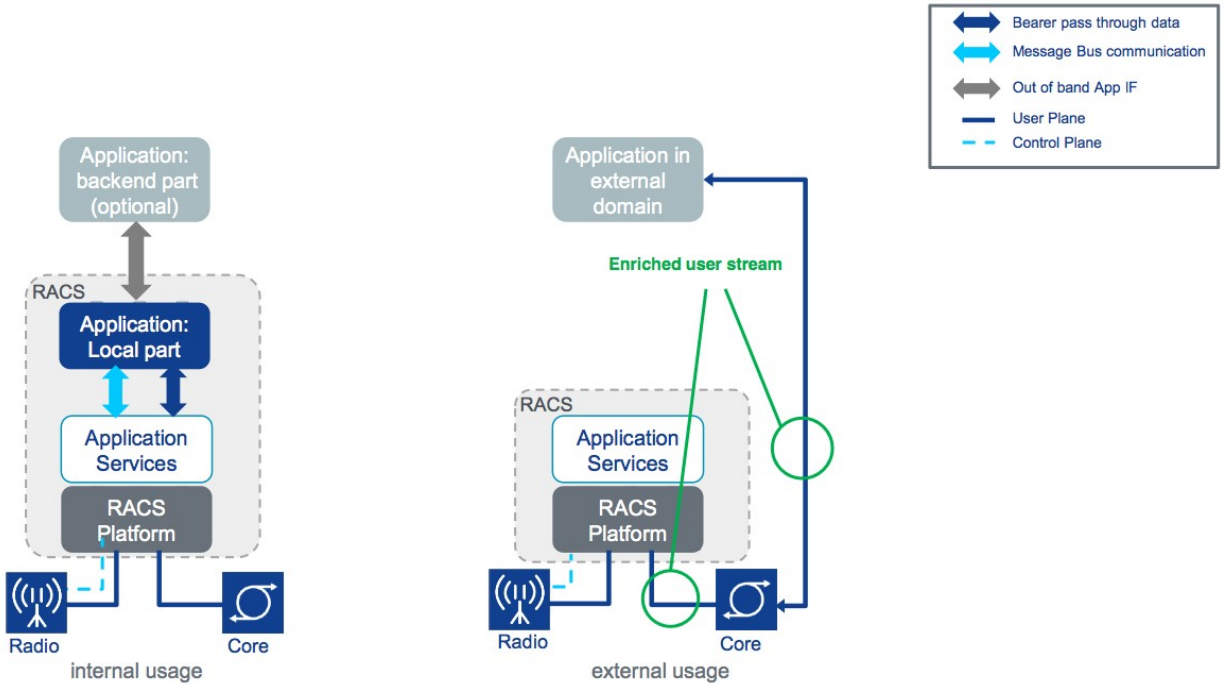
RACS Application Overview



- Backend Server could be located at the operator network or it could be hosted by a 3rd party, i.e., somewhere in Internet.

RACS Application Overview

- External usage is an example of how to use inbound signaling to communicate with an external application.
- Alternatively, out of band signaling, i.e., a dedicated signaling channel, could have been used.



MEC Programming in RACS environment

- Work Scope in AppFactory Process
- RACS Application Overview
- RACS Services
- Application Types
- User Plane Service
- Application Integration
- RACS Application Specifics
- Development Process
- VM Development Environment
- Application VM
- Development Support

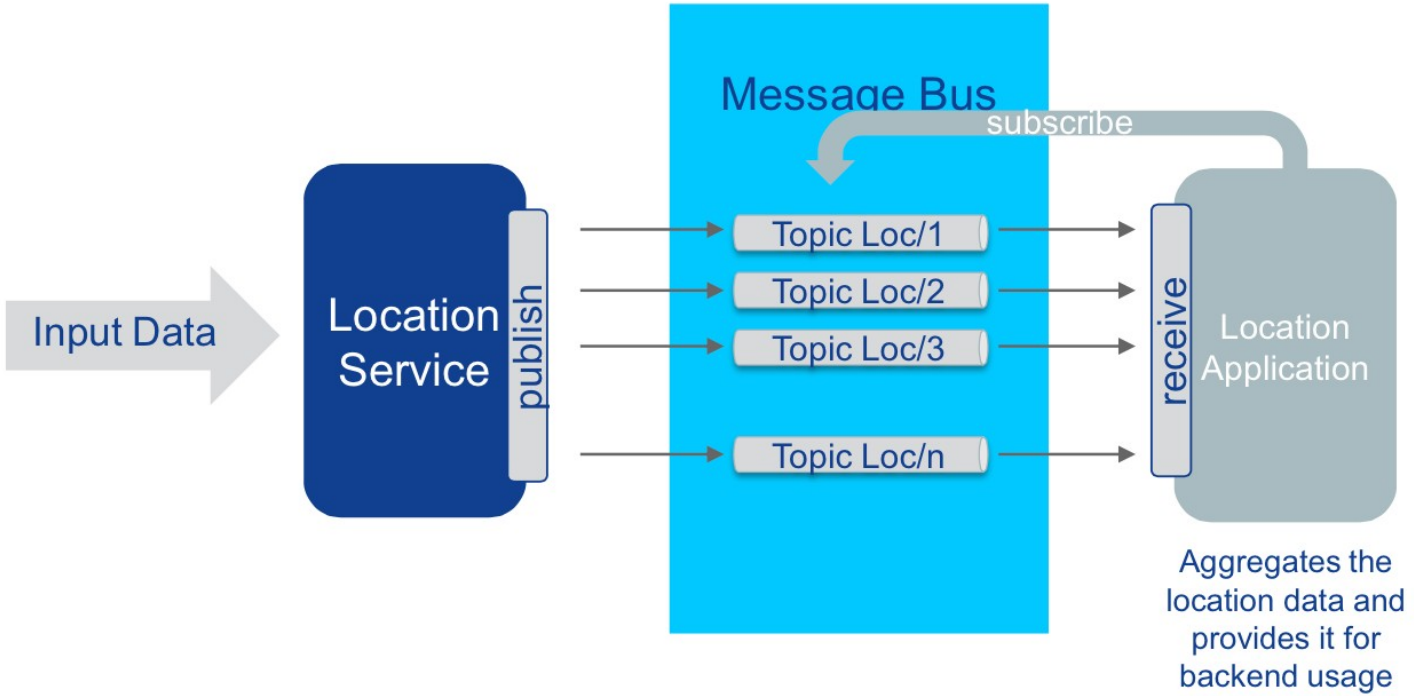
RACS Services

- An application can use RACS specific value added services;
  - Cell related services
    - Number of users
    - Cell load
  - User related services
    - Cell ID
    - Throughput Guidance (TG)
  - Location related services
    - User location
    - Users in area
- Based on the UE's IP, various UE related info can be fetched using these services.

RACS Services

- Services available via MQTT (MQ Telemetry Transport)[1] based Message Bus (MB).
- MQTT a lightweight broker-based publish/subscribe messaging protocol.
- Application subscribes the preferred topics and then starts to listen new events.
- Service (“producer”) publishes new events to MB via which they are delivered to the subscribers.

RACS Services



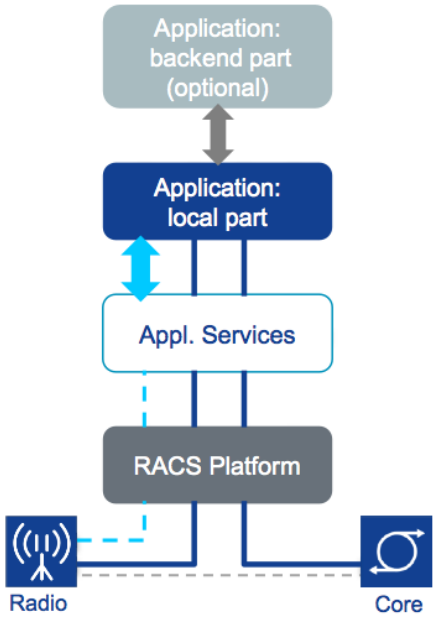
MEC Programming in RACS environment

- Work Scope in AppFactory Process
- RACS Application Overview
- RACS Services
- Application Types
- User Plane Service
- Application Integration
- RACS Application Specifics
- Development Process
- VM Development Environment
- Application VM
- Development Support

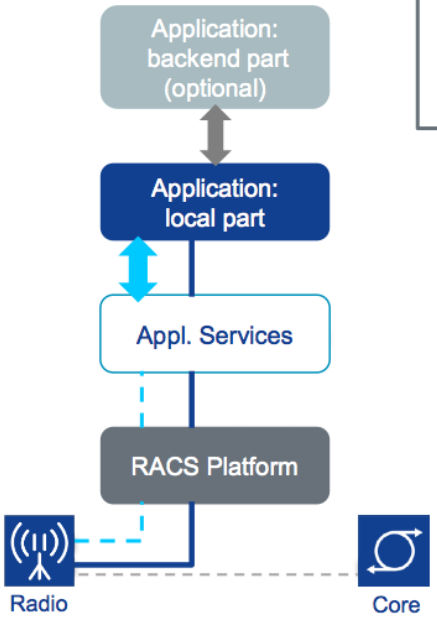
Application Types

- Transparent/pass-through applications.
  - User plane UpLink (UL) and/or DownLink (DL) traffic is intercepted by an application.
  - Intercepted packets can be monitored, modified, etc.
  - Data session is between UE and server.
  - Offloading rules (application specific) define what traffic is rerouted to the VM.
- Terminating application.
  - An application acts as a server towards UEs.
  - An application has a unique IP address reachable outside of the RACS and has FQDN(s) in the operator's DNS.

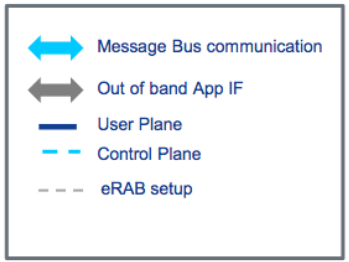
Application Types



Transparent/pass-through applications



Terminating Application

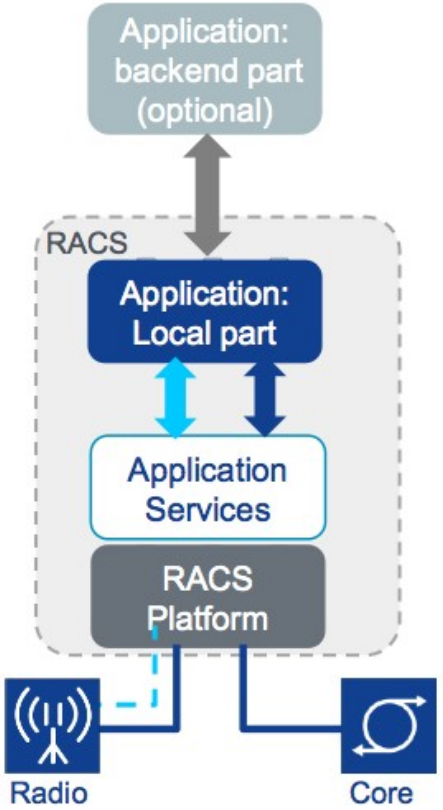


MEC Programming in RACS environment

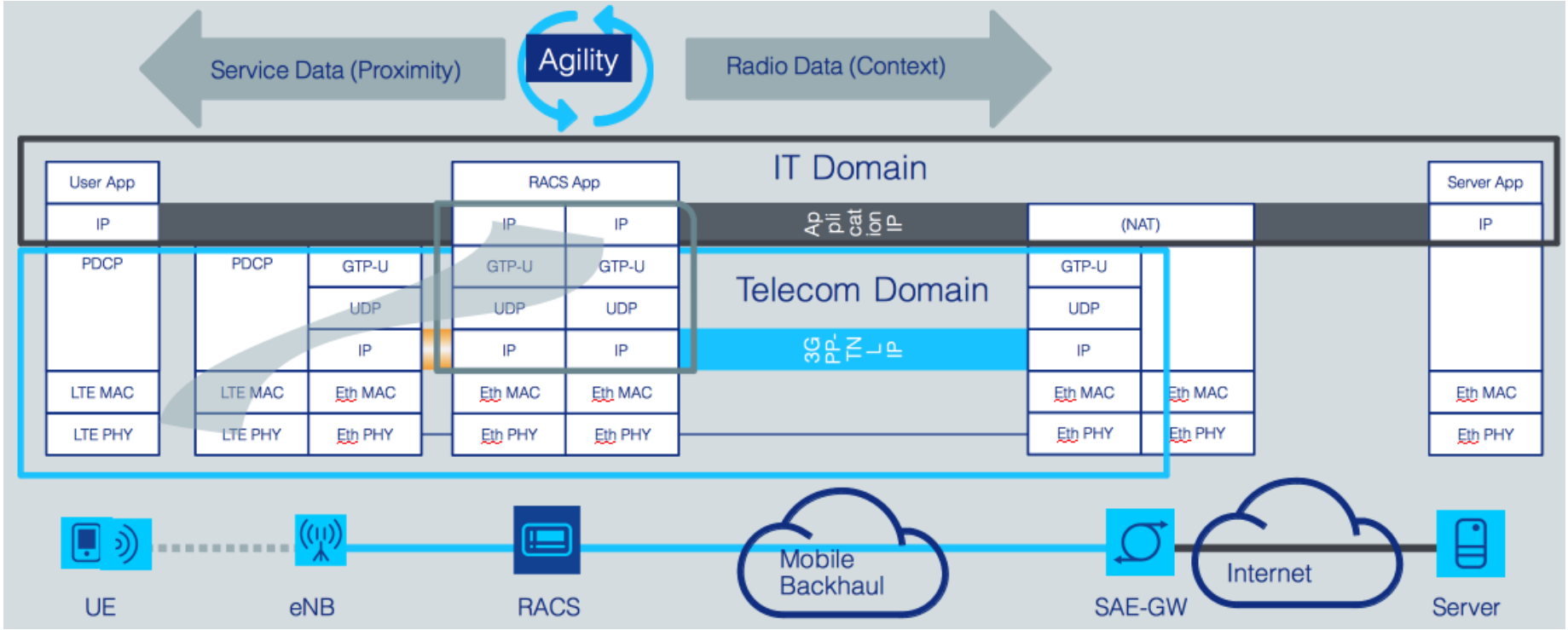
- Work Scope in AppFactory Process
- RACS Application Overview
- RACS Services
- Application Types
- User Plane Service
- Application Integration
- RACS Application Specifics
- Development Process
- VM Development Environment
- Application VM
- Development Support

User Plane Service

- Bridges 3GPP SIPTO (Selected IP Traffic Offload) to VMs, providing a Virtual NIC (vNIC) for applications.
- Pass-through applications can intercept up- and downlink traffic, potentially modifying or shaping it, and sending back to original PDP context.
- The IP routed format delivers end-to-end packets from/to UE "as is", without any outer encapsulations.



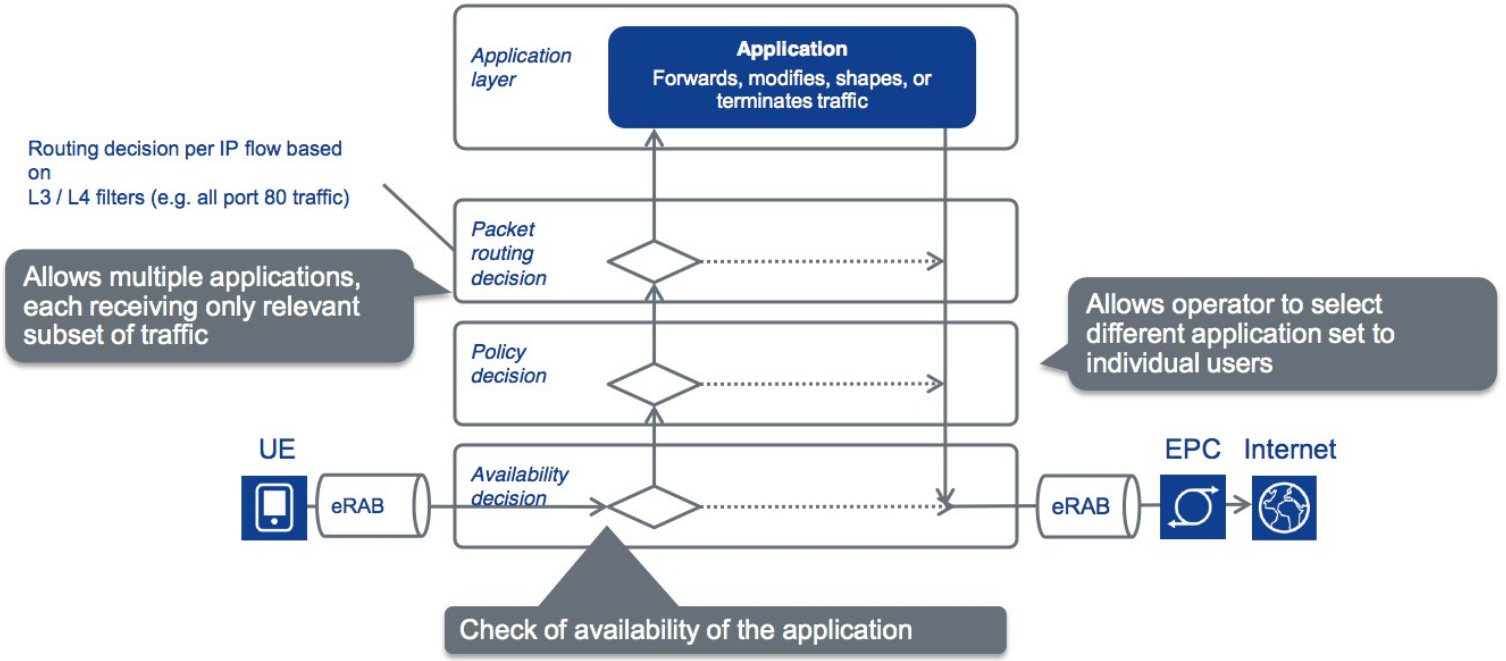
User Plane Service



Aalto-yliopisto

NOKIA

User Plane Service



- Offloading rule could be as simple as <4==ip_version && proto==TCP && (80==dst_port || 8080==dst_port)> for uplink traffic
- More fine grained rules based on L4+ info has to be implemented in the application.

MEC Programming in RACS environment

- Work Scope in AppFactory Process
- RACS Application Overview
- RACS Services
- Application Types
- User Plane Service
- Application Integration
- RACS Application Specifics
- Development Process
- VM Development Environment
- Application VM
- Development Support

Application Integration

- All applications must send a heartbeat signal at a predefined rate.
  - UDP based protocol (example Python client available).
- Sending this heartbeat indicates that the application is not only available, but also running correctly and either continuing to process work or preparing to do so and not in need of reset.
  - The heartbeat does not indicate that all services provided by the application are ready to process work immediately so the heartbeat should not be used to indicate service availability.
- In the event of an application failure the heartbeat function ensures that:
  - The correct action to restart the application is taken.
  - This may result in an escalation, if the application cannot be restarted.
  - Bypass any application traffic until the application is available again.
- Application VM also must synchronize itself using NTP.
  - Example Python client available.

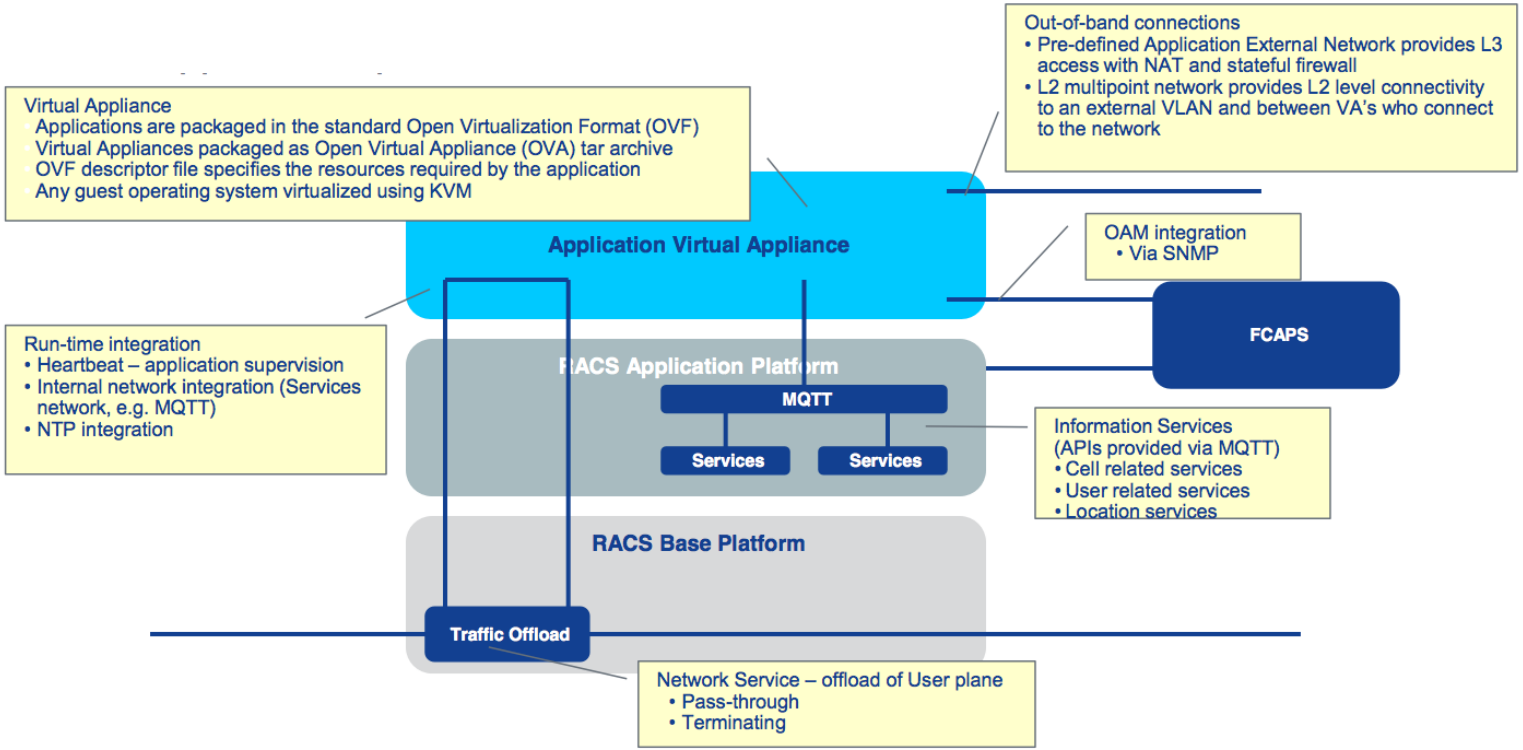
Application Integration

- Application specific XML files define various application properties like:
  - Application identity related info.
  - Offloading rules (what traffic is offloaded).
  - Network resources (vNICs and their configuration).
  - Computing (CPUs, etc.).
  - Storage (storage capacity, mount points, etc.).
  - Port forwarding (“X->Y”, i.e., packets outside of the VM destined to port X should be forwarded to VM's port Y).
    - For instance, to access application VM's SSHD, instead of using port 22, one would use something like 29001 port instead.

MEC Programming in RACS environment

- Work Scope in AppFactory Process
- RACS Application Overview
- RACS Services
- Application Types
- User Plane Service
- Application Integration
- RACS Application Specifics
- Development Process
- VM Development Environment
- Application VM
- Development Support

RACS Application Specifics



MEC Programming in RACS environment

- Work Scope in AppFactory Process
- RACS Application Overview
- RACS Services
- Application Types
- User Plane Service
- Application Integration
- RACS Application Specifics
- Development Process
- VM Development Environment
- Application VM
- Development Support

Development Process

- Liquid App's VM is based on KVM (Kernel-based Virtual Machine).
- Each (Liquid App) application runs in a dedicated VM.
- Applications can be developed in any programming language as long as it runs on a standard KVM's VM.
- For multi-process applications, some parts can even run outside of RACS, i.e., in a backend server.
- VM should run RedHat based distro; CentOS 6.5 is recommended.
- VM creation environment can be also non-RedHat based distro like Ubuntu.
- VM creation environment (e.g., Ubuntu OS) can be a VM itself as long as nested virtualization is supported by the used hypervisor.
- Note that, for instance, VirtualBox does not support this.

Development Process

- When designing the application, the following aspects should be considered;
  - Internal communication over vNICs,
  - Communication with external/3rd party entities and
  - Application type (transparent or terminating).
- What traffic DL/UL (in terms of src/dst IP address spaces, src/dst ports, protocol types, etc.) should be offloaded.
  - These packet policy rules are part of the application specific configuration.
- Application specific FW rules.
- Restricted resources for a VM;
  - A single logical CPU and
  - 1GB RAM.

Development Process

- For transparent/pass-through applications, it is application specific decision on how to intercept incoming data from vNIC(s) and deliver it to the entity processing it.
  - For instance, various packet filter mechanisms, e.g, BPF (Berkley Packet Filter), can be used.
- Where the intercepted packets are processed, i.e., user space vrs. kernel space, is again application specific decision.
  - Additionally, the intercepted packets could be processed in backend server(s).

MEC Programming in RACS environment

- Work Scope in AppFactory Process
- RACS Application Overview
- RACS Services
- Application Types
- User Plane Service
- Application Integration
- RACS Application Specifics
- Development Process
- VM Development Environment
- Application VM
- Development Support

VM Development Environment

- We have tested CentOS 6.5, Ubuntu 12.04 and Ubuntu 14.04.
  - Other main distros like Fedora, Mint, Debian and so on should also work assuming that required SW packages/tools are available (see below).
- For creating a new VM devel OS;
  - Install Ubuntu 14.04 (a basic installation).
  - Install extra packages (isomd5sum, libvirt0, libvirt-bin, libvirtodbc0, python-libvirt, virt-manager, virt-viewer, virtinst, kvm, python-software-properties and libguestfs-tools).
  - Install mosquitto (http://mosquitto.org/download/).
  - Update guestfs appliances ($>sudo update-guestfs-appliance).
  - Edit /etc/libvirt/qemu.conf and set “user” & “group” as "root" and restart libvirt-bin ($>sudo stop libvirt-bin; sudo start libvirt-bin).

MEC Programming in RACS environment

- Work Scope in AppFactory Process
- RACS Application Overview
- RACS Services
- Application Types
- User Plane Service
- Application Integration
- RACS Application Specifics
- Development Process
- VM Development Environment
- Application VM
- Development Support

Application VM

- Applications are packaged as defined by the DMTF Open Virtualization Format (OVF) specification V2.0.
- Packages are provided as an Open Virtualization Archive (OVA) file (OVF Specification 2.0 (DSP0243 2.0.0)).
- The OVA file is packaged as a TAR archive, and contains the following files:
  - An OVF descriptor (.ovf XML file).
  - An OVF manifest file (.mf), which lists the SHA-256 digests of all of the other files in the OVA, as specified by the OVF specification v2.0.
  - An OVF certificate file (.cert), which contains a certificate, along with a signature of the manifest as defined by the OVF specification v2.0.
  - One VM disk image in QCOW2 format. This must always be a boot disk image. There must not be additional data disk images.
- Additionally, if the application uses any MQTT services, then an authorization policy file should be included with pathname auth/acl.xml.

Application VM

- Basically the (simplified) VM creation procedure goes as follows:
  - Create a basic VM image, e.g., perform automated (using kickstart) CentOS 6.5 minimum installation.
  - Mount the created VM.
  - Copy the application and all needed SW packages, i.e., all application SW dependencies not already present in the basic installation.
  - Construct application config file.
  - Encapsulate everything into OVA file.
- All these steps can be scripted (as done for instance by a sample GoOn app):
  - $>sudo ./build-app.sh --name GoOn --in /home/test/CentOS-6.5-x86_64-bin-DVD1.iso
  - $>sudo ./build-ova.sh --ovf GoOn.ovf --image GoOn.qcow2 --privatekey ca.root.key --certificate ca.root.crt

Application VM

- Once the application VM is created, it can be locally mounted and accessed;
  - 1. Define the virtual machine .
  - “$>virsh define <App_Name>.xm”l, where App_Name is the name of your application, e.g., GoOn.
  - 2. Start the virtual machine running your application image.
  - “$>virsh start <App_Name>”.
  - 3. Start the virtual machine console.
  - “$>virsh console <App_Name>”.
- To stop and undefine the VM;
  - 4. Stop the virtual machine running your application image.
  - “$>virsh shutdown <App_Name>”.
  - 5. Undefine the virtual machine.
  - “$>virsh undefine <App_Name>”.
- Alternatively, you can also use the GUI (“$>virt-manager”) to do the same things.

MEC Programming in RACS environment

- Work Scope in AppFactory Process
- RACS Application Overview
- RACS Services
- Application Types
- User Plane Service
- Application Integration
- RACS Application Specifics
- Development Process
- VM Development Environment
- Application VM
- Development Support

Development Support

- Nokia provides Applications Developer’s Guide - a comprehensive user and reference manual for RACS application design & implementation.
  - [Reference to be provided]
- There is a sample app (GoOn), which is terminating “echo server” application using a one RACS's service via MQTT.
  - Verification is done using R&D certificates.
  - Provides a working example of RACS integration.
  - [Reference to be provided]

NOKIA