

Bringing the telecommunication solution to the next level

Check how the experts do it

Bringing the telecommunication solution to the next level. Check how the experts do it.

Strzegomska Street 36

53-611 Wrocław
Reception 1st Floor, Green Towers B
Opening hours 8:00-18:00
Phone: +48 71 777 3800
Fax: +48 71 777 30 30

Lotnicza Street 12

54-155 Wrocław
Reception Ground Floor, West Gate
Opening Hours 8:00-18:00
Phone: +48 71 777 4002
+48 71 777 4001

Szybowcowa Street 2

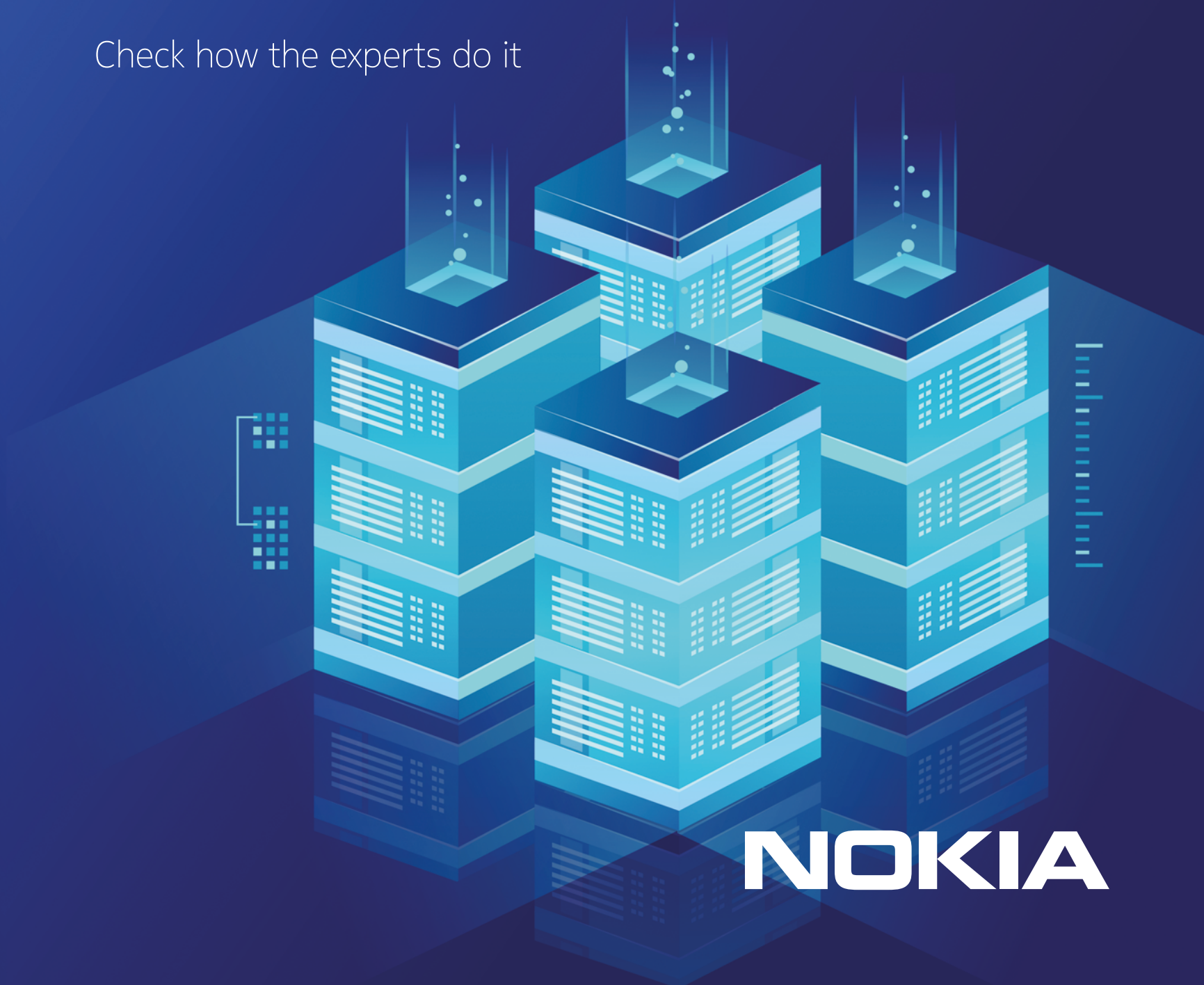
54-130 Wrocław
Reception Ground Floor, West Link
Opening Hours 8:00 – 18:00
Phone: +48 71 777 4000

e-mail: kontakt@nokiawroclaw.pl
www.nokiawroclaw.pl

© 2019 Nokia

NOKIA

NOKIA



Dear Readers,

I am very pleased to present you fourth part of Nokia Book – another edition of articles written and published in the European Software and Engineering Center in Wrocław, with invaluable contribution of our specialists.

“Bringing the telecommunication solution to the next level. Check how the experts do it” is a collection of 16 articles covering various topics from testing techniques, tools and development directions. You can read here about theoretical aspects as well as real-life case studies and challenges for the future.

Besides significant aspects for experienced engineers, this book has also great educational value for beginner testers. I believe that all enthusiasts of telecommunication will find the articles interesting.

I would like to thank the authors for their passion, hard work and outstanding contribution to that project.

I wish you a pleasant read,

Bartosz Ciepluch

Head of Nokia Networks European Software
and Engineering Center in Wrocław



Testing areas and techniques

4		
6	1.1 — Łukasz Parzybut	What should be tested? Tests vs Real World
10	1.2 — Rafał Łukasik	Run-to-completion model in Data Plane processing
16	1.3 — Mariusz Brzóska	Benefits of software testing using the Scrum framework. How to avoid the Cargo Cult?
24	1.4 — Kamil Musiał	Wireshark application in telecommunication area
32	1.5 — Tatsiana Viarbitskaya and Grzegorz Czaiński	CI Through Project Life Cycle
40	1.6 — Przemysław Ratajczak	Automation of Software Testing in LTE Environment
46	1.7 — Mariusz Zamłyński	Throughput testing techniques and challenges in modern LTE-Advanced Pro networks
52	1.8 — Łukasz Grządko	Primality Tests



Laboratory testing tools

62		
64	2.1 — Bartłomiej Radwan	Network tools – whim or must? Overview of the network tools used in the R&D labs
70	2.2 — Patryk Furman	Hardware resources monitoring with Cacti
76	2.3 — Tomasz Szandala	One play to rule them all: write Your first Ansible playbook



Tests and Laboratory development directions

82		
84	3.1 — Grzegorz Juszczak	Application of OpenStack Technology for Managing R&D Virtual Machines in 5G Infrastructure Laboratory
90	3.2 — Wojciech Zmyślony	Passive Intermodulation Cancellation
96	3.3 — Łukasz Szewczyk and Bartłomiej Radwan	Virtual Environment Manager Cloud solution. How to create fully automated R&D laboratory
100	3.4 — Łukasz Michna, Grzegorz Dygoń and Stanisław Pospiech	Migration towards future Data Center Networking
110	3.5 — Adam Chamera	Using Artificial Intelligence for radio performance test result analysis

Testing areas and techniques



1.1

Łukasz Parzybut
What should be tested?
Tests vs Real World

6

1.2

Rafał Łukasik
Run-to-completion model
in Data Plane processing

10

1.3

Mariusz Brzóska
Benefits of software testing using
the Scrum framework.
How to avoid the Cargo Cult?

16

1.4

Kamil Musiał
Wireshark application
in telecommunication area

24

1.5

Tatsiana Viarbitskaya and Grzegorz Czański
CI Through Project Life Cycle

32

1.6

Przemysław Ratajczak
Automation of Software Testing
in LTE Environment

40

1.7

Mariusz Zamłyński
Throughput testing techniques
and challenges in modern LTE-Advanced
Pro networks

46

1.8

Łukasz Grządko
Primality Tests

52

What should be tested?

Tests vs Real World

Łukasz Parzybut
Verification Engineer
MN 4G RAN



1. Introduction

We live in an era of an unusually dynamic development of technology. Things are getting smaller, more and more accurate right before our eyes. There is progress in every field of science.

We can cure more effectively, anticipate disasters and warn people about them. We can prevent a lot of things that once tormented mankind. We improve our lives in harmony with Mother Earth.

Technology gives us the possibility of global communication. It only takes an eye blink to travel from one place to another. Thanks to the global network and its availability, smart minds from around the world can exchange data, results, and ideas. Because of this possibility we are able to check the weather, make a beautiful photo and share it with friends all around the world. Everything thanks to a device that fits into your hand.

Behind the phenomena are people with passion and knowledge who want to develop the world. They create, invent, and design new products to make our lives easier. However, each of them, before putting their work into our hands, must make sure that it works as planned. If you were not sure that everything was solid, resistant, and safe, the astronauts definitely did not want to board the shuttle of the Apollo 11 mission.

And here is where the testers come along.

2. What is testing?

Testing is a comprehensive issue and covers many fields of science. We can test air quality, efficiency of solar panels, a new treatment method, software. Testing consists of individual components, tests. So, what is a test? According to the dictionary definition:

Take measures to check the quality, performance, or reliability of (something), especially before putting it into widespread use or practice¹

The word refers to many things and could refer to other fields of science. In the article, we will refer to the cases of testing the software and devices on which the software is located.

3. Goals of testing

To answer the question „What should be tested?” we should start with general testing goals. In the IT sector, we have a lot of diversity

¹ <https://en.oxforddictionaries.com/definition/test>

when it comes to languages in which the code is written. Languages differ in syntax, complexity, and purpose. For example, it would be difficult to write a batch code per processor in HTML. However, each software must be tested before it goes to the client. The question arises: should the testing criteria of different software groups be the same? The answer is no. However, if there are different criteria, are the goals changing?

The answer to this question is as follows: the goals do not change. The main purpose of software testing is to detect faults, program imperfections, incompatibilities with the expected results. Their detection can prevent many dangers: from losing customer’s trust, even to causing damage to the health of the person who uses this software.

Often it is time-consuming to correct the program and it may affect the customer’s receipt of the product after the set deadline. However, it is worth taking care of its quality because it can be an asset in the overall assessment of trust in the supplier.

4. Rules of testing

We can distinguish seven basic principles of testing:

1. Exhaustive tests are not possible

It would be an ideal situation if our application would be tested in every possible way and in all conditions. Then we would be sure that our product has no right to be advertised. Is it possible? No. It would take too much time and time is usually limited. The solution is to focus on those parts that are most frequently used and vulnerable.

2. Defect clustering

At some stage, the software becomes very complex and each developer knows the situation in which one small error causes an entire avalanche of bugs. The best practice is to identify very risky modules and focus on them.

3. Pesticide paradox

This rule applies to the situation in which we have a field, and that the plants growing on our field could yield a good crop, we must provide them with protection against pests. However, when we will try to remove pests with only one insect repellent, they will finally become resistant to it. Similarly with tests, if the test will be carried out in the same form, the program will finally become resistant to it. It is therefore important to design new tests or edit the old ones.

4. Testing shows a presence of defects

The description of this principle is very simple: it is virtually impossible to find an application that has no errors. And testing has these glitches to reveal. When the application passes all the tests positively, there is a very good chance that some bugs still exist.

5. Absence of an error

This principle of testing refers to the belief that if our software has passed all the tests and challenges, it is ready to be released to the market. The fact that the project, program, or website does not have any defects does not mean that it meets the requirements of the users.

6. Early testing

It is important that software tests are performed at every stage of its life cycle and not only at the end. The later the error is found, the more likely it is that it will bring more errors. Taking care of it can save you time and resources.

7. Testing is context dependent

Programs, applications, and websites differ from each other depending on their purpose. The blog website will be tested differently than the banking system. It is important to match the way and methods of testing to the level of risk.

5. Model of testing

The previous paragraphs familiarized us with the goals and principles of testing, these are the bases that every tester should know, remember, and most importantly refer to during their work. In this part I would like to introduce more detailed issues related to software testing. The first thing to know is the V model. The V model is the most commonly used model of software testing and it shows different testing phases in good way. It requires the creation of two groups of people who participate in the project: a team of developers and a team of testers. It consists of verification phases and validation phases separated by the implementation phase. The verification phases include:

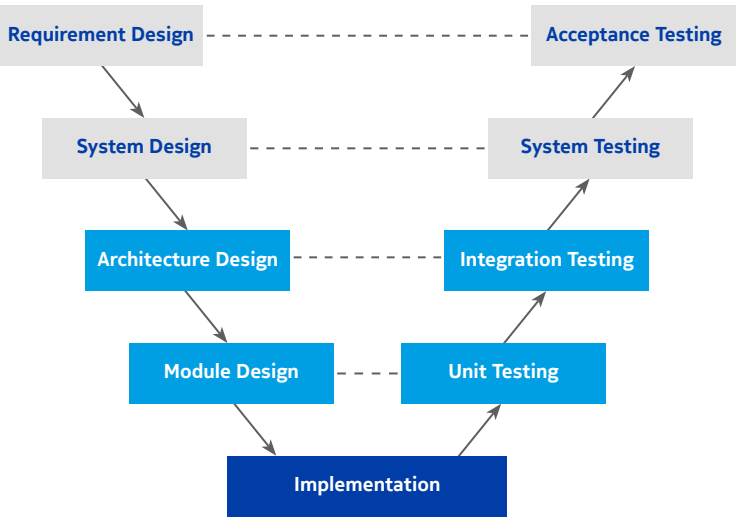
- **Requirements analysis**
During the analysis of the requirements it is determined what the ideal product should be capable of. Its functionality, interface, security, and performance is determined. It is saved in the so-called user requirements document. During this stage, acceptance tests are designed.
- **System design**
During the system design stage, the ways in which a given functionality will be achieved are determined, and whether it is possible to implement it. The document relating to user requirements is edited. System tests are also designed at this stage.
- **Architecture design**
This planning phase is responsible for setting high-level requirements. Lists of modules, relationships between them, dependencies, databases, technologies, and integration tests are saved.

- **Module design**
Here, a detailed outline of each module is designed, as well as a way of communication, internal communication interfaces and API, and a way of showing errors. In addition, unit tests are designed.

Then the implementation phase follows. Another is the validation phase, during which the testers take up work, and it consists of:

- **Unit testing**
The lowest possible level of testing in the V model. It is responsible for verifying whether each module has the right output, how it reacts to an incorrect input, and is it safe and secure.
- **Integration testing**
Integration testing refers to checking whether individual modules communicate with each other and whether they can coexist with each other.
- **System testing**
System tests are designed by the customer team. It is the client who determines what the requirements are, what the program should be able to do. All applications should show functionality, resistance, and safety of use. Performance tests, strength tests, regression, and so on are used here.
- **User acceptance testing**
The last stage of testing. The tests composed by end users are validated here. Realistic inputs are used. This is to show whether the product is ready to be launched on the market.

Figure 1



Model V is just one of many models. These include primarily prototyping, the RAD model, Rational Unified Process (RUP) and agile, iterative-incremental models, and many, many others. Model V has been presented here to show different types of tests, at various stages of the program's life cycle. The choice was subjective.

6. What should be tested?

I think that after reading everything above, we can conclude that the answer to the title question is not trivial. We know that there are many different stages, we know what the general principles are and why tests are needed at all. However, what should be tested?

It should be remembered that it is very difficult, if not impossible, to anticipate every possibility with which the customer will use the final product. In the case of devices, even the ambient temperature can have a big impact. Due to the multitude of factors, testing everything in every possible configuration takes infinite time.

In my opinion, each project has different goals, different requirements, and a different degree of risk towards the final recipient. After gathering all this into one, the final picture of our work emerges and based on this image we should edit the appropriate tests. For a blog website, they will concern the correctness of adding comments and the speed of switching between subpages. For the shuttle, it will be the strength of the material. In my opinion, the requirements that the product should meet and the safety of its use should be tested.

7. Testing VS real life

Testing is an inseparable part of every project, it depends on the validation phase whether the product will be implemented by the customer. Test design starts at the beginning of the project from the so-called acceptance tests, by system tests for integration tests and unit tests.

To be able to compare testing with real conditions, we should ask ourselves: what are the real conditions? Real conditions are those in which our program will be used by end users.

The customer defines how the end user will use the program. Based on this definition, tests are designed starting from those at the highest level, that is the acceptance tests. Based on the highest level, testers design what is below and they determine how the whole machine should work. Its weaknesses are identified and protected accordingly. Individual modules are developed. Testing aims to show that the software is consistent and robust, and works in a certain way.

Every stage of validation tries to imitate the real conditions. There are test cases that imitate normal use but there are also others

that expose the program to overload. Different variations of input, different numbers of events, different combinations. However, we must remember a few principles: absence of an error and exhaustive tests are not possible.

For example, a car manufacturer must be sure that every component of the car works properly, is in place, and cooperates properly with other elements. The person must know that each component meets certain requirements. All parts are put together into a whole car, which interface is a possibility of driving. However, as testers we cannot say where the rider will drive, we can only check its resistance to the range of conditions that might be met. In real life we are dealing with many combinations so that the simulation would last indefinitely. Therefore, the most frequent and the most critical ones are selected in the testing process.

About the author

I have studied physics with programming specialization at the University of Wroclaw. During the studies I began a part-time job in embedded focused company, where I learnt the software and hardware skills. After almost two years, I decided to change the way of my life and try as a tester in Nokia. Now with our team we verify the working features. Personally, I am interested in low-level programming, mathematics, and telecommunication. After work I love road cycling.

Łukasz Parzybut
Verification Engineer
MN 4G RAN

Run-to-completion model in Data Plane processing

Rafał Łukasik
Engineer, Software Development
MN ECE



Introduction

Multi-core processing is a vital part of the modern computational architectures. It is present in huge server clusters, personal computers, and smartphones. Parallel processing allows applications and services to use computing resources in an energy-efficient manner while being executed uninterruptedly and independently from processes executed on other cores.

The key advantage of multi-core architecture is its provision of a central processing unit (CPU) resources, especially important in cloud computing. As a result, CPU resources can be shared by different processes based on their demand. These capabilities are provided at a lower cost than the cost of the core, which is powerful enough to handle all the processing in one process [1].

Multi-core architecture of general purpose processors has also been proposed as an answer to the increasing demand for computing resources of the mobile networks and mobile devices for example, in form of mobile edge computing (MEC). In MEC, control plane (C-plane) and user plane (U-plane) processing is moved to the computing cloud located in the close proximity to the source of data.

The processing required by the data plane can be implemented on a multi-core general purpose chip, although in terms of computing, data plane has specific requirements. In general, the data plane handles the arriving traffic in the current network node. The traffic is processed and sent back to the network. From the outside, such network node looks like a single super-core which processes packet by packet, while being constrained by the packet rate [2].

Due to this constraint, especially nowadays when high data throughput is a primal demand, the packet processing must be deterministic and straightforward. There is a need to identify and optimize a critical path which is taken by most of the packets inside the current network node [2].

This article presents specific aspects of data plane applications running in a multi-core environment. In the following sections, the article elaborates in detail on the typical aspects of packets processing and how different demands are addressed by the run-to-completion model, which can be utilized for the design of data plane applications.

1. The optimization problem and the supporting model

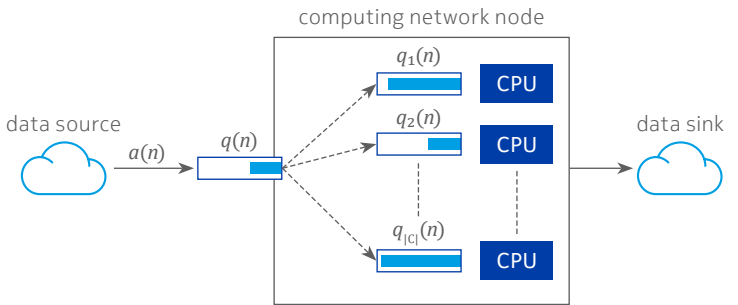
1.1 System modeling

Let us consider a network composed of data source, computing network node, and data sink (terminal) with a queue between the source and the node. Network node is hosting a data plane application in

a multi-core environment. Each core executes the same set of instructions on packets read from the same input and written to the same output stream [2]. The processing of packets from an input queue is distributed across cores inside a node based on scheduling and CPU load balancing policy. However, on purpose, we do not want to limit this analysis only to a specific network architecture; the number of data sources and terminals, and I/O queues can be different. The only rule is that the arrival packet flow from the input must be forwarded to the output.

A typical server consists of cores which are numbered and has a homogeneous architecture. Each core has the same computing power. Each transmission path has an associated queue or buffer for packets, since processing of each packet takes time (see [Figure 1](#)). Consequently, a number of packets is serviced in n th time interval. In [Figure 1](#), $a(n)$ is a stochastic process of the transmission and there is some average transmission rate of incoming packets [1].

Figure 1 Computing network node with parallel processing



1.2 CPU load

Depending on the current packet arrival rate, scheduling, and CPU load balancing policy, the packets can be processed on one of the cores with a certain rate [1]. Such processing takes time, therefore causing CPU load. Hence, there is a maximum load which can be obtained in each time interval of the transmission. And obviously, there is a maximum rate of packets which can be achieved by causing such load.

1.3 Queuing model

Every time interval indexed by “ n ”, $a(n)$ packets are transmitted by data source. Let $q(n)$ be a queue length at the data source. Since the transmission goes on, a number of packets in a buffer changes. Arriving packets are queued and stored in the buffer. At the same time, packets are dequeued from the buffer and serviced by CPUs with a specific rate. This process should balance the size of the

buffer. Otherwise, the transmission would have to be broken, packets would have to be dropped and eventually lost.

At each process running on core, the arriving packet can be stored into another buffer before it is processed. In **Figure 1**, $q_1(n)$, $q_2(n)$, ..., $q_{|C|}(n)$ denote the queues length of the core. The number of packets in the queue depends on the current state of that queue, arrival rate of the packets, and processing rate of the core.

The total service rate at the network node is a sum of processing rates of used cores.

1.4 Problem formulation

There is a need to decide how much of the overall transmission will be handled by each available CPU in the server. The control action at each core includes a selection of the CPU load [1]. This allocation is a subject of CPU load balancing policy. For example, the scheduler in the network node may allocate a different fraction of CPU time in a given time.

The state of the network node which should be considered, while deciding about the load balancing policy, includes the data rate from the source, and the current number of packets in buffers. In our case, it is reduced to the decision about the number of packets processed by each core in a current time interval.

1.5 Solution

The presented model was defined to simplify the analysis of the optimization problem which we face while designing a packet processing application in a multi-core environment. It clearly shows that the main actors are data packets and cores [1, 2]. Thus, the highly optimized data plane software should utilize the abstraction of these two things.

2. Run-to-completion

Run-to-completion is a scheduling model in which each task runs until it either finishes or explicitly yields control back to the scheduler for example, to use a hardware accelerator. In multi-core environment, run-to-completion is considered as a variant of cluster model [2]. It combines several cores into a cluster which executes not fragmented, single staged, non-preemptive program. Such clusterization is very often referred as partitioning [1, 2].

As presented in the system model from Section 1., one of the ways to achieve higher overall performance is to focus on the efficiency at which each packet is processed entirely by one core [1, 2]. Since

the run-to-completion model already defines the entity of a single stage execution, it can be easily mapped to packet processing.

2.1 Processor affinity

To get the notion of a processor in our model, a popular technique of processor affinity (also known as CPU pinning) can be applied [3]. It binds a specific physical core or cores to a specific process. In that way, process data always remains in cache, thus avoiding cache misses [3]. Depending on the environment, processor affiliation can be used to achieve better performance.

2.2 Polling

As the packet arrival rate increases, the CPU load must also increase. If the cores serve interrupts, the high load leads to the context switching overhead and results in performance degradation. Instead of using an interrupting serving routines for dequeuing packets from buffers, the polling mode is often used to achieve extreme efficiency. The poll mode driver is implemented as a process pinned to the CPU core and it periodically checks the receive queues for the arriving packets to be dispatched and serviced by the single routine, which runs uninterruptedly until it finishes. Because the polling process constantly checks the queues, the CPU is constantly loaded. In addition, the packet processing is executed in the same context as polling, so the context switching overhead does not occur.

2.3 Data structures for packet processing

Several types of data structures can be independently identified for the data plane model:

- Packet descriptors – data structures per packet
- Data structures per flow/connection
- Current context, state for the processing state machine, statistics counters – data structures per protocol [2]

In general, the access to data shared between the cores must be protected against race conditions.

Since the whole packet is processed by a single core in a run-to-completion model, packet descriptors are not shared [2]. Hence, they do not require a restricted access.

In ideal situation, it is desired to be able to let all processes running concurrently. Touching the same memory inherently adds invisible contention [4]. Thus, in applications, which have requirements of a very high performance, it is allowed to duplicate the same context data in different places of memory. The access to one copy is

then restricted only to one core. It is a convenient and theoretically simple solution, especially assuming that processes are only reading common data and do not modify them. Otherwise, a different solution should be considered to avoid handling of asynchronous events coming from different cores.

There is an extensive research conducted on queuing and shared data structures. Generally, algorithms for concurrent data structures, including FIFO queues, fall into two categories: blocking and non-blocking [4]. Blocking algorithms allow a slow or delayed process to prevent faster processes from completing operations on the shared data structure [4]. Non-blocking algorithms guarantee that if there are one or more active processes trying to perform operations on a shared data structure, some operations will complete within a finite number of time steps [4]. On asynchronous (especially multi-programmed) multiprocessor systems, blocking algorithms suffer from significant performance degradation when a process is halted or delayed at an inopportune moment [4]. Possible sources of delay include processor scheduling preemption, page faults, and cache misses [4]. Non-blocking algorithms are more robust in the face of these events, hence they are an absolute preference in data plane applications.

Some of the algorithms proposed in the research papers assume the existence of non-delayed processes attempting to perform operations on shared data structures [4]. These operations must be assumed to be completed in a finite time. It directly shows how vital is the attempt to design a straightforward and deterministic packet processing in the run-to-completion model.

Despite of the analysis of the particular solution, it is worth to recognize that some of them utilize the idea of the procedure which is inherently non-preemptable. Such operation is called atomic [4]. Execution of atomic operation is virtually like calling of single instruction of the processor – it will always complete without interruption.

In extreme situations, it can be assumed that the whole packet processing operation is atomic. Hence, the processing of the next packet originating from the same flow cannot be started before the previous one is completed, no matter whether on the same or other core. In consequence, the concurrent access to the flow data can be avoided. In addition, such approach supports the other need explored in the next section: packets belonging to the same flow which are arriving to the node must go back to the network in the same order.

2.4 Parallel packets processing from a single flow

The atomic queue [5] allows the processing of the packets originating from a single flow, one at a time. It does not mean that the packets from other flows cannot be processed concurrently with

these packets. It is now a matter of CPU load balancing policy to allocate processing resources efficiently considering the additional restriction.

It is easy to notice that this solution works inefficiently, especially when we consider only one existing flow.

Hence, being supported with applying different techniques of avoiding the concurrent access to the shared data, it is worth to explore more sophisticated techniques of keeping packets in order.

The most widely used method to achieve this is to assign a unique sequence number to each packet when it enters the processor, allow the packets to be processed out of order, and have them reordered basing on their sequence numbers just before their return to the network [2]. It means that it should be allowed to execute concurrent processing of packets and then spend additional CPU time for reordering. Unfortunately, it violates the principle of the presented run-to-completion model which tries to only keep the notion of the packet and the processor.

This article shows only how much can be done to efficiently utilize general purpose architecture of processors for data plane applications optimized for very high throughput. In the next section, the article presents the comparison between the run-to-completion model with a model which uses general purpose fair scheduling.

3. An alternative approach

As seen till now, by introducing the run-to-completion model, we maximize the overall packet rate in a data plane application which is running on a multi-core machine. In the long term, the goal is to maximize a joint utility of data processing throughput and to minimize the power consumption [1]. The economical usage of power (computing) resources is not addressed by the run-to-completion model.

In other words, cores which are affiliated to packet processing processes are not used efficiently when packet rate is much lower than the maximum value. CPU time is therefore consumed by the polling process which checks the queue state whether there is a packet to be serviced. It creates artificial CPU load and, when it takes too long time, it might appear simply unprofitable because occupied cores cannot be used for other tasks or cannot be disabled.

In addition, the processor affinity (Section 2.1) requires pinning of the process to the physical CPU what disables hyper-threading (Intel's proprietary solution). It is done not to share internal resources of physical core. This optimization technique should be considered only when it is confirmed that the processor affinity improves the data plane application performance in a particular case.

Constraints, applied due to the presence of data shared between processing stages, can cause additional inefficiency in some conditions and leave processors loaded but also unused for example, the atomic queue and single flow problem discussed in Section 2.4.

The prohibition of preemption, mainly made to avoid context switching overhead and to make data forwarding process more predictable, seems to be a root cause of the constant load of a fixed number of CPUs in a presented solution.

The main goal of data plane processing is to avoid packet loss. If packet rate is sufficiently low and there are no real-time constraints, the preemption and fair scheduling can be considered.

Preemption and fair scheduling should allow to efficiently use available CPUs by data plane application and other programs which can run on the same core. Nevertheless, mixing data plane and control plane applications associated with the same network brings additional risks and should be avoided [2]. The preemption caused by a higher priority control plane application process can lead to longer packet queues, longer delays, and congestion. The preemption of a control plane process by the data plane delays the analysis of asynchronous hardware and network events, and results in analyzing them when they are already obsolete for example, the link that was previously reported down might be up by now [2].

The use of standard tools and loose coupling with the hardware has obvious advantages in software engineering. The decision of introducing high performance data plane optimizations should be made by also considering software development phase.

Conclusion

The highly optimized data plane application design has many specific aspects which are not present in the control plane and other types of applications. Nevertheless, due to the rise of multi-core processing which utilizes general purpose architecture, it is a must to face these problems and define appropriate solutions. The tight coupling of the data plane with hardware platform, that is physical CPUs and accelerators, does not make it easier, as the virtualization of the application obviously brings additional processing overhead. The scalability of data plane applications is also a matter of question.

The run-to-completion model of data plane application tries to address this variety of issues. It does not mean that it solves them completely or must be strictly applied when it comes to the packet processing. However, it reveals the potential of being a basis for efficient and to some extend scalable solution deployed on the popularity gaining clouds.

References

[1] M. Kangas, “Stability Analysis of New Paradigms in Wireless Networks”, University of Oulu, 2017
[2] C. F. Dumitrescu, “Design Patterns for Packet Processing Applications on Multi-core Intel® Architecture Processors”, Intel Corporation, December 2008
[3] J.Donovan,K.Prabhu,“BuildingtheNetworkoftheFuture:Getting Smarter, Faster, and More Flexible with a Software Centric Approach”, CRC Press, 2017
[4] M. Michael, M. L. Scott, “Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms”, University of Rochester, 1996
[5] Nokia Siemens Networks, “Open Event Machine”, 2013 [ONLINE] Available at: https://github.com/zlim/open_event_machine [Accessed 20 May 2018].

About the author

I graduated from Wroclaw University of Technology where I studied Electronics and Telecommunications. For 1 year, I’ve studied also Wireless Communications Engineering in University of Oulu in Finland. I work as R&D Engineer in MN ECE. With my team, we develop User Plane Software for 5G cellular network. In my work, I combine my programming skills in C++ with knowledge about telecommunications. I constantly learn from others and on my own mistakes.

Rafał Łukasik
Engineer, Software Development
MN ECE

Benefits of software testing using the Scrum framework.

How to avoid the Cargo Cult?

Mariusz Brzóska
Engineer, Software Development
4G RAN



1. Short description of the Scrum framework

Fundamentals, roles, artifacts and values of the Scrum framework have been described in the Scrum Guide, however there are many other publications which illustrates the framework in a more detailed way. Following the Scrum Guide: “*Scrum is not a process, technique, or definitive method. [...] The Scrum framework consists of Scrum Teams and their associated roles, events, artifacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum’s success and usage. The rules of Scrum bind together the roles, events, and artifacts, governing the relationships and interaction between them.*” [1]

The Scrum Team consists of a Development Team, a Scrum Master and a Product Owner. The Scrum Master is neither a project manager nor a team leader; he is rather a servant-leader and a facilitator for the Scrum Team. His main responsibilities include:

- Removing barriers between development and the customer/Product Owner
- Improving productivity of the Development Team
- Coaching the Development Team and other stakeholders in the scope of understanding and using Scrum
- Facilitating Scrum meetings

The Product Owner is responsible for the Product Backlog. One of his responsibilities is presenting Product Backlog items to the Scrum Team. He ensures common understanding of Product Backlog items by the Scrum Team. He decides on the priorities of the items.

The Product Backlog is a prioritized list of tasks which need to be completed for a product, including features, enhancements, fixes and improvement actions. It should be continuously refined to be as precise as possible, which is the main aim of the Product Backlog Refinement meetings.

There is also a lower level backlog- Sprint Backlog. It consists of Product Backlog items for certain Sprint, which provides a new working functionality for the product- the Sprint Goal. The main goal of Sprint Backlog is visibility of the status of the work that needs to be done in a Sprint. The status is updated during the Daily Scrum.

Scrum defines a couple of solutions to maximize the productivity and regularity, while at the same time minimizing the number of meetings which need to be organized for the Scrum Team.

The main solution is the Sprint. It’s a relatively a short period of time during which the increment of a product is created. The increment is commonly understood as a sum of work done by a team fulfilled Definition of Done criteria, which provides releasable feature.

Sprint Planning meeting help to define the Sprint goal, which is a short description of what the Scrum Team wants to achieve during the particular Sprint. It allows to understand WHY team is building the next increment. The team translates high-level user stories into smaller tasks, which go to the Product Backlog. Sprint goal is achieved by completion of Product Backlog items.

Figure 1 B[2]

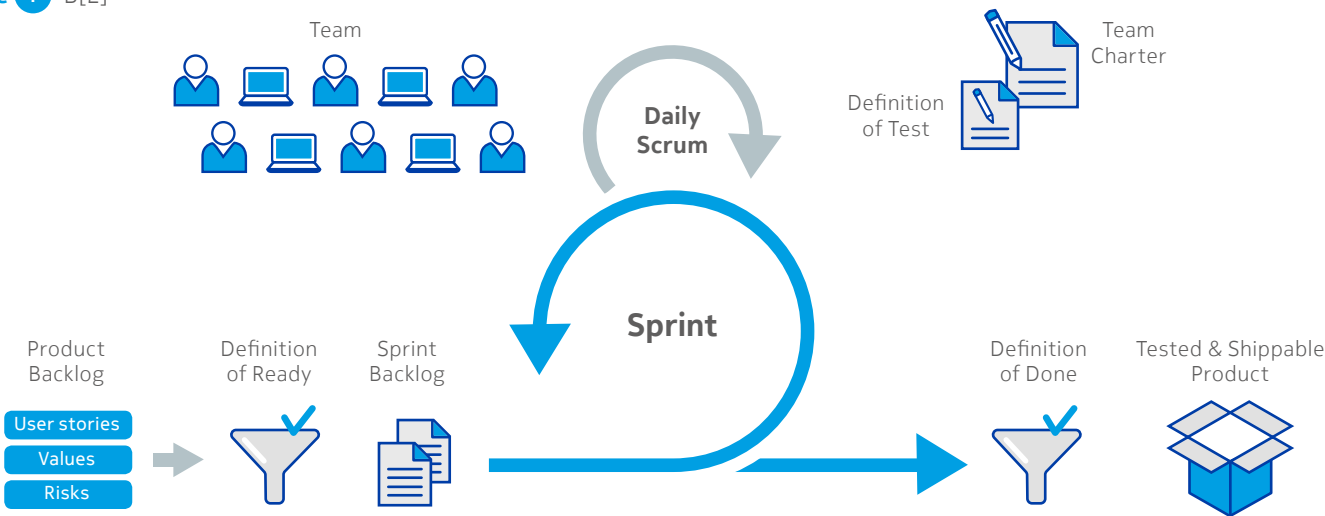


















Figure 2 The team’s Scrum board[3]

User Stories this Sprint	Tasks To Do	In Progress	Done
			 
	 		
	  	 	

The next important part of the Scrum framework is the Daily Scrum. It is an about 15-minutes long daily meeting. Each team member can share information regarding:

- What did I do yesterday that helped the Development Team meet the Sprint Goal?
- What will I do today to help the Development Team meet the Sprint Goal?
- Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?

These questions help to plan the next 24 hours of the Sprint; however, sometimes additional, more detailed questions should be asked. It is a very valuable meeting, which improves knowledge sharing inside a team.

At the end of a Sprint, the increment should be verified by stakeholders. This is achieved during a Review meeting. Then assessment about what has been done during the Sprint is made. Development Team may provide demo of the delivered increment. Based on the feedback during the Review meeting, a Product Owner can adapt Product Backlog for the next Sprint. An item is considered “Done” when the set of rules called Definition of Done is fulfilled.

As continuous improvement is a part of Scrum, feedback about the most recent Sprint is highly desirable. Sprint Retrospective Meetings help to identify and improve weak points of interpersonal relations, processes or tools inside the team. They are an occasion to discuss technical and interpersonal issues and to comment if something went well.

2. How does the testing process work in Scrum and what are the benefits of having a tester in a Scrum Team?

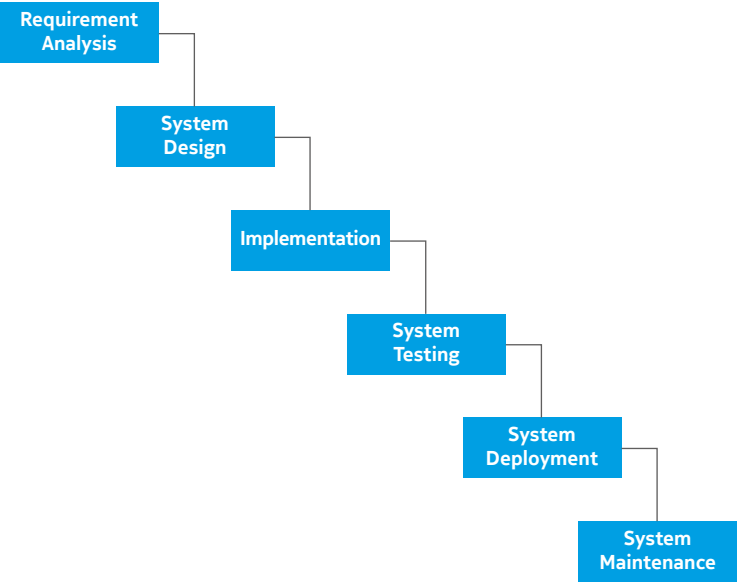
Test management process using the V-model or the Waterfall model is quite different than with Scrum. In traditional methods the leading tester, test manager or test architect is responsible for planning the test strategy, scope, etc. He is leading the team and managing testing tasks. The main disadvantage of these models is that they are not flexible – especially the Waterfall. The next drawback is that SW bugs are detected at the late state of product development. Nonetheless, these methodologies are still in use in some projects and fit well for them.

A Scrum Team has no dedicated test manager, because it should be a self-organized and cross-functional group of people. Cross-functional doesn’t mean that each team member has all competences needed for developing a product end to end. It rather means that all team members are focused on delivering a working product and synergy of their skills from different fields improves the product quality. So, testers in a Scrum Team can be an added value and they can help to improve quality of the final product from the early phase of product life cycle.

One of the key roles of a tester in a Scrum Team is helping with defining user stories, Definition of Done and Acceptance Criteria. If a team is working in ATDD (Acceptance Test Driven Development) methodology, then during the planning and design meeting, at the very beginning of the Sprint, all Scrum Team members should take part in defining of Acceptance Criteria (AC). They are considered as a set of requirements that SW must meet, to be accepted by the customer or the Product Owner. New features are developed and test cases are designed based on these requirements. The most popular template of defining Acceptance Criteria is: “*Given... When... Then*”. This is the generic form of AC definition: with given context or circumstances, when an action is carried out, then a result is observed. The biggest advantage of this method is simple language which guarantees common understanding of the criteria. There is no split between the testers and the developers in the Scrum Team, however somebody within the development team may have testing skills and then take on the tester’s role. Often testers are in possession of detailed domain knowledge which can significantly improve quality of the created criteria.

Quick information about the level of quality of the developed software is very important. In the traditional model, where separate development and testing team are working on a product, tests are run after the software is ready. It takes some time before a tester can analyze test results, find the responsible development team and give them feedback. In Scrum Team feedback is much faster. It is possible due to team members’ location but also thanks to test automation and Continuous Integration practice. Tests runs even

Figure 3 Waterfall model[4]



after each commit to the common code repository. There are several levels of testing under the Continuous Integration process: unit tests, component tests, integration tests and system tests. Results from the execution of them are clearly visible on the dashboard and can be tracked. There is no need to wait for the end of execution of test cases on the previous stage. Tests might be executed simultaneously on all levels. Test cases created by testers are often more complex than those designed by developers as unit tests. Integration test cases might discover bugs in interactions with other system components and interface misalignments. Automation gives quick daily feedback about code quality, but not everything can be automated.

In practice, when a tester raises a bug report, an investigation on the development side needs to be performed. Quality of the bug report has significant impact on the problem resolution time. Relevant information about an observed issue needs to be provided for the developer. For that reason, many organizations introduce a template for bug reports. It forces good practices and improves common understating of the problem on tester’s and developer’s side. Sometimes it is not enough and additional questions need to be asked. When there is communication between two or more teams, there is always some time waste regarding answering the questions, especially when considered teams are located apart from each other. In a Scrum Team this problem is significantly reduced. Due to the shared location of the team, all misunderstandings regarding test environment and scenario can be resolved immediately. Moreover, delivered corrections might be tested rapidly, so feedback to the

developer is given straight away. It significantly improves the velocity of SW developing process and reduces costs for bug fixing. I’ve encountered a quote, which gives a sense of test automation process, regarding the team where it is performed: “*Also common is the test automation group zombie. This zombie is the practice of assigning test automation to a dedicated team of test automators. The appeal is that we can keep developers focused on writing new code instead of writing and maintaining automated tests. The danger is that test automation inevitably lags development, so feedback from testing is delayed in a way that significantly reduces its value.*”[5]

The next great benefit of the presence of a tester in a Scrum Team is his knowledge of software testing and strategies. Based on testing training like ISTQB and experience, testers can train the rest of the team in the quality assurance process. Developers often don’t have detailed knowledge about system operations or hardware. Testers may bring this information to the team and thanks to that quality of the final product is better.

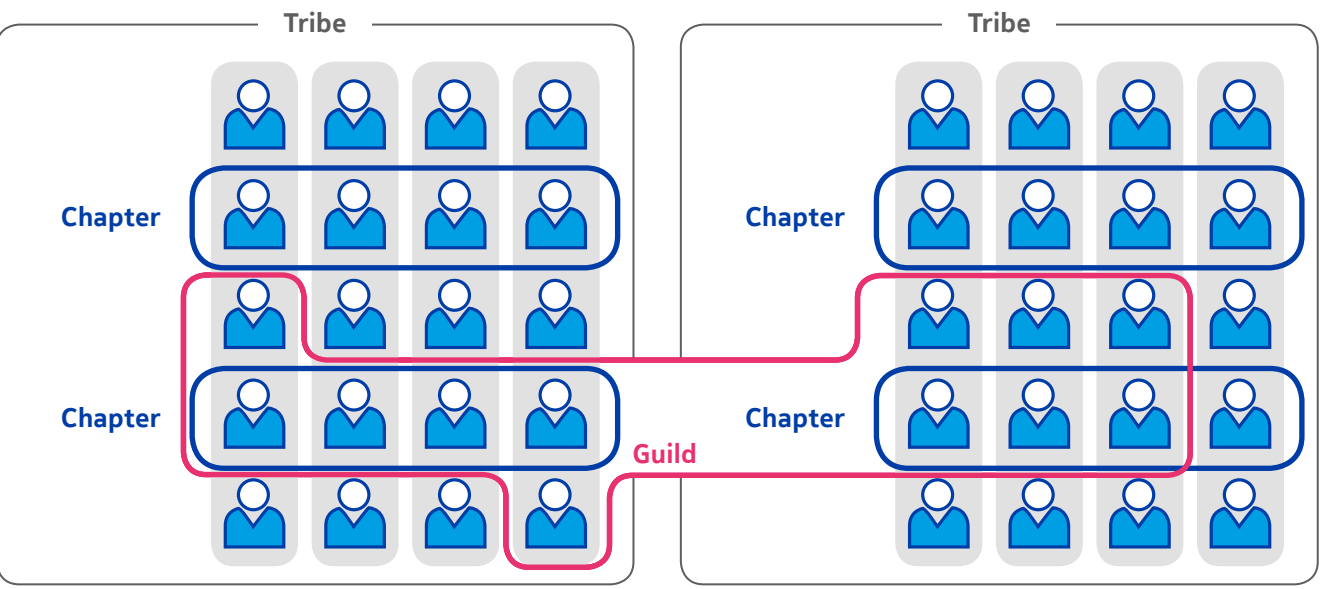
While it is an advantage, it might be also a drawback. One of the biggest drawback is that developers and testers make the same assumptions in their work. They sometimes have the same attitude to challenges in daily tasks. It is not a problem when they need to order a pizza for a lunch, but it is when they must design code and test cases to check this code. In software testing it is expected from testers that they are independent and objective. It isn’t so simple when they are working in a Development Team.

They may unconsciously try to fit to the developer’s vision of the product. For that reason, some test scenarios might not be covered due to influence of developers, or even not recognized. Testers may also affect decisions about software design within a team. For example, they can suggest selection of a simpler algorithm regarding a procedure, because of the smaller testing effort which must be spent on test automation. Sometimes it could be dictated by environmental issues, like tool availability. As the next disadvantage of testing without dedicated team we can identify the additional effort for synchronization of testing tasks across Scrum Teams. In this case some additional meeting might be required for assurance, that there is no gap in test coverage and teams are synchronized.

3. The Spotify model as a next step in managing the workflow?

From time to time new solutions for making project management more effective, or improving organizational structure of the company are emerging. Sometimes they are entirely new, sometimes just modification of older solutions. One of them is the so-called Spotify model. It was developed in the Swedish entertainment company Spotify Technology SA. As it was said by Marcin Floryan,

Figure 4 Spotify organizational structure[6]



one of the chapter leaders in Spotify, the Spotify model should be considered rather as an organizational snapshot of the engineering culture, than a model as such. But where does this idea come from? How has it evolved into the today's form? At the beginning, Spotify employees used the Scrum methodology to manage their workflow, but they have abandoned this framework.

Spotify adapts Agile for their needs and, as it is visible in the **Figure 4**, some new terms have been introduced regarding organization of teams. All Scrum practices became optional. Scrum Master is transformed in this organization into an Agile Coach to indicate his role as a servant leader. A Scrum Team became a Squad, also cross-functional and self-organized, but a more independent team, with end to end responsibility for their part of the product. The Product Owner prioritizes their work like how it was done in Scrum.

These changes shall entail more autonomy for teams, which results in better motivation of employees and improves teams' velocity. Because of such set-up, it is important to communicate the common vision of the product and ensure that high level of alignment between Squads is kept. Otherwise each Squad will go in a different direction and ultimately organization will fail. This is the role of a Tribe leader in cooperation with Product Owner. The Tribe is a set of Squads and is applied in larger organizations.

Interdependency gives an opportunity to use multiple tools and practices across the Tribe. Each Squad within the Tribe can develop

different methods and tools in their work. Then they may discuss it with other Squads and propagate their solutions across the Tribe. It is kind of a lightweight process of examining and selection of optimal tools and practices.

There are also mysterious names for other groups called Chapter and Guild. The Chapter should be considered as a group of people having responsibilities from a certain area, for example Developers or Testers. They meet to synchronize and discuss challenges regarding their competence areas. The Guild is a kind of a virtual team or task force. It brings together people from different Squads and Chapters, and their aim is to provide a solution for specific problem or deliver some improvement. It is one of the mechanisms of information flow between Squads working on a bigger feature together. One of the biggest problems in an autonomous team is knowledge sharing with other teams. It is a big issue especially, while these teams are working on one big feature. Bringing them together in a Guild partially solves this problem.

As it was mentioned before, there is a kind of independency and it also applies to team composition. If there is are testers, they should have specific knowledge and skills.

Test automation skills are essential for fast delivery of test results. As continuous testing is considered fundamental for this "model", manual testing is restricted to exploratory tests. Working in close cooperation with developers, knowledge about acceptance test

driven development and white box testing is also more than welcome. As frequent feedback is expected, high level of communication skills is essential for good cooperation within a team. High autonomy of a team must be balanced by result-oriented mindset and high discipline in organization of their own work.

All these solutions are continuously changing and should be tuned to current business needs. As a matter of fact, even Spotify abandoned this model and isn't using it any more in such form.

4. What is the Cargo Cult and why may it have an adverse impact on testers and their teams?

The Cargo Cult began in Fiji in the 19th and early 20th century when local tribal societies met with western people. Then the same phenomenon occurred in Papua New Guinea and other islands of Melanesia. The clearest examples, in my opinion, were described by American soldiers during World War II. As time went by, U.S. Army moved across the Pacific. As more and more troops were going ahead, they had to be supplied. Once Americans have taken island of strategic importance, they started to build a landing strip. Thanks to that, aircrafts could land on the island and deliver supplies. Then another building structures and when the natives first saw all the goods brought by the foreigners, they were shocked and impressed by Soldiers sometimes shared their canned food or other small items with the locals. These gifts were of considerable value to them.

This process was observed by the natives. They thought that all this cargo was sent by gods and it only took acting like foreigners to get it. So, they started building a makeshift landing stripes, fake aircrafts, antennas made of bamboo, etc. They also noticed soldiers' uniforms, various signs on them and tried to copy them. The drill was considered by Melanesians to be a kind of a ritual, and they also tried to imitate that. Local people blindly emulated incomprehensible behaviors and believed that they could succeed.

We can observe similar behavior in many aspects of our life. From daily routines to advanced scientific research. Even Nobel's Prize laureate Richard P. Feynman described this phenomenon regarding research techniques.[7] We still expect that if some mechanisms which are working in other organizations will be imported to our organization, we will succeed. Going back to it is for example deployment of a new bug tracking tool, because it is doing well in other companies. After some time, that there is no place for additional information which we are using for generating our statistics etc. We must realize that without deep analysis of our working environment and comprehension of the new way of working, we won't succeed. Many companies tried to deploy Agile or Scrum and failed. We mainly identify deployment as the next factor of success in scrum an entire organization. For example: it is not possible to continuously

deliver software without continuous testing, so all work using the same methodology. As it was mentioned before, regarding the Spotify model, sometimes it is not necessary and such setup may work in some way, however it is not We are hearing from time to time that some company failed to deploy . There are several reasons why it may happen.

Breaching fundamental principles of Scrum, we cannot expect success at the end, especially with an inexperienced team. For example, if a few past sprints had different length each, it is hard to estimate tasks for the next one, because we do not have a common benchmark. If Scrum meetings are not facilitated by the Scrum Master and other team members, it is destructive for the whole team. Sometimes it is because of lack of the Scrum Master's confidence and respect within the team. Nobody then knows what is the progress and what is going on in the Sprint. On the other hand, if a Scrum Master tries to micromanage the team, then there is no Scrum.

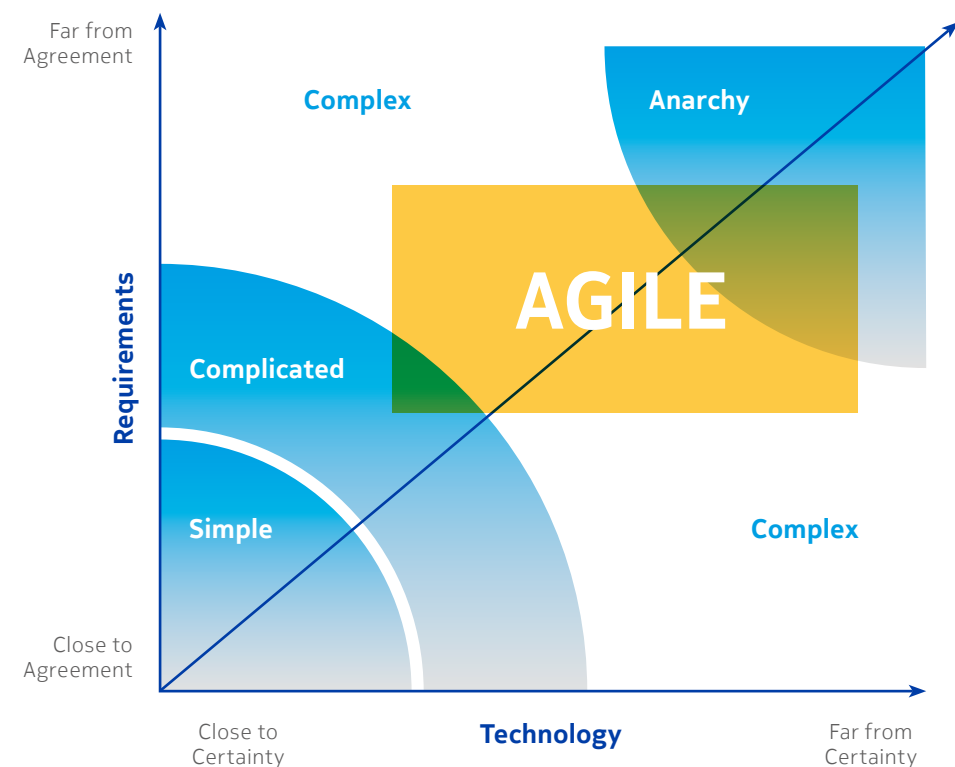
"Stop trying to borrow wisdom and think for yourself. Face your difficulties and think and think and think and solve your problems yourself. Suffering and difficulties provide opportunities to become better. Success is never giving up." It was said by Taiichi Ohno. He was a Japanese industrial engineer and businessman. Author of the *"Toyota Production System: Beyond Large-Scale Production"*. He is the father of the Toyota Production System, adapted and transformed in the United States to a Lean Manufacturing. It is not a plug and play solution and Motorola and BlackBerry found out about it. It is hard to start with Scrum without relevant people. Both engineers and managers must be ready for changes.

5. Why testing in the Scrum methodology is not relevant for some projects?

Scrum is a framework for developing complex products, which cannot be planned and controlled in a traditional way, or it is not effective. It suits fast-changing projects well because it allows it quick response to the customer's needs. Change of scope or requirements is nothing new and each task in the backlog can be re-prioritized.

I would point out two main reasons not to use Scrum: people and project type. If people are against the changes or do not have comprehensive knowledge about Scrum, it is not a good idea to introduce a new framework. Team members must understand Scrum principles and apply them in daily work. Due to close cooperation inside a team, people should be open-minded, communicative, and have relevant technical skills. If we already have the relevant people, we need to consider if the project is worth the changes. So, if the scope of the project is well known and most of the expected changes are almost certain, then there is no need to use the Scrum framework. There is no need to introduce Scrum if your organization is dealing with repetitive projects.

Figure 5 Agile application in the context of project type[8]



There is also a technical aspect. As Continuous Integrations is an important part of the software development process, there must be an environment allowing to make this happen. Before the first sprint starts, relevant tools and hardware must be provided. If there is no possibility to meet this requirement or there is lack of knowledge how to set up the environment, the project will fail immediately.

As the level of uncertainty in the project is rising, there is a higher recommendation to use one of the Agile techniques. It allows better adaptation to the changing environment.

6. Summary

Software testing methodologies have evolved in the past years and continue to evolve further. Good management of the testing process is crucial to provide high quality features. It is hard to describe what an important and resource-consuming activity it is. Good cooperation across the Development Team and continuous synchronization

with the Product Owner helps to keep everything up to date and respond to the customer's needs. Testing takes more than half development time. One of the most significant people in the IT world is a kind of description of this process: *"We have as many testers as we have developers. And testers spend all their time testing, and developers spend half their time testing. We're more of a testing, a quality software organization than we're a software organization."*[9] These words have been spoken by Bill Gates regarding the testing practices in the Microsoft.

Scrum relies on feedback, and testers are the most important source of feedback. It is very important to provide high quality test cases and for that reason they should also be under the review process.

Some may say that developers have unit tests which are sufficient, and testers are not an added value in a development team. Following the V-Model, there are test levels corresponding to development stages and unit tests do not guarantee a high quality of code. Testers have a considerable influence on many aspects of the software

development life cycle. Shortening software quality feedback time and participating actively in relevant team activities has a significant role in an organization. For that reason, test environment and methodology should be continuously improved inside organization, considering all technical and project-related aspects. As IT industry is changing fast and many new solutions are presented, it is not a good idea to introduce each of them in every organization. Any change must be preceded by a deep analysis of the organizational and engineering culture within the company, and then after a positive verification, changes may be incorporated.

References

- [1] The Scrum Guide ©2017 Ken Schwaber and Jeff Sutherland
- [2] Testing in Scrum: A Guide for Software Quality Assurance in the Agile World Tilo Linz Rocky Nook, Inc., 28 mar 2014 – 240
- [3] www.inflectra.com
- [4] <http://www.softwaretestinghelp.com/what-is-sdlc-waterfall-model/>
- [5] Dale Emery
- [6] <https://spotifylabscom.files.wordpress.com/2014/03/spotify-engineering-culture-part1.jpeg>
- [7] <http://calteches.library.caltech.edu/51/2/CargoCult.htm>
- [8] Strategic Management and Organisational Dynamics: The challenge of complexity to ways of thinking about organisations
- [9] Bill Gates

About the author

I have graduated from Wrocław University of Technology in Electronics and Telecommunication. I work as a Software Development Engineer in the 4G RAN department. Our Department develops cutting edge software features for LTE eNodeBs. Thanks to us, state-of-the-art features regarding Internet of Things become a real. I am a part of a Scrum Team however, I'm also experienced as a Software Tester working with the Waterfall model.

Mariusz Brzóška

Engineer, Software Development
4G RAN

Wireshark application in telecommunication area

Kamil Musiał
Continuous Verification Specialist
MN 5G RAN



In telecommunication field, especially in new technology development, the connection problems are often found.

The following article concerns the usage of Wireshark program in Nokia laboratory tests and problem investigation.

1. Introduction

Wireshark is a opensource sniffer – program whose main purpose is to capture and analyze the network data flow. It allows capturing data packets in real time, recording, and decoding them. Due to a large number of add-ons, it can recognize and decode numerous communication protocols. Wireshark includes filters, color coding, and other features that let user dig deep into network traffic and inspect individual packets. Also the *promiscuous mode* is available allows the user to see all the other packets on the network, instead of only the packets addressed to certain network adapter/interface. Many organizations don't allow Wireshark and similar tools on their networks. However, in Nokia Wireshark is a basic diagnostic tool used in many departments.

2. Main functionality

Wireshark offers various useful options depending on how thorough the investigation has to be. After choosing which network interface should be monitored (WiFi, Bluetooth, Local) user sees basic information about captured packets like: time, source, destination, protocol, length. Below, see a typical WiFi captured data stream.

This dataflow capture provides basic information about data traffic on – in this case – WiFi interface, and what happens between the computer and the WiFi router user is connected to.

Except for its basic functionality of capturing, Wireshark can – as mentioned above –analyze packets thoroughly. Using a filter window, user can choose and analyze only packets that came from or go to a specific IP address or packets sent by the specific protocol.

On the screen below the traffic has been filtered and only packets sent / received via DNS protocol are shown. The Domain Name System (DNS) protocol is a 'hierarchically distributed database', which is a formal way of saying that its layers are arranged in a definite order, and that its data is distributed across a wide range of machines (just like the roots of a tree branch out from the main root).

Most companies today have their own dedicated DNS servers to ensure the computers can find each other without problems.

You can notice the detailed information about chosen packet. Following the 7 layer ISO/OSI model, it is possible to analyze each captured frame (which contains mac address) in Data Link layer, packets (which contains IP addresses) in Network layer, protocol details in Transport layer, etc...

Besides simple filtering option Wireshark gives the possibility to combine filters.

In **Figure 3** you can see 3 filters used together: source, destination IP and protocol.

Figure 1 Basic Wireshark dataflow

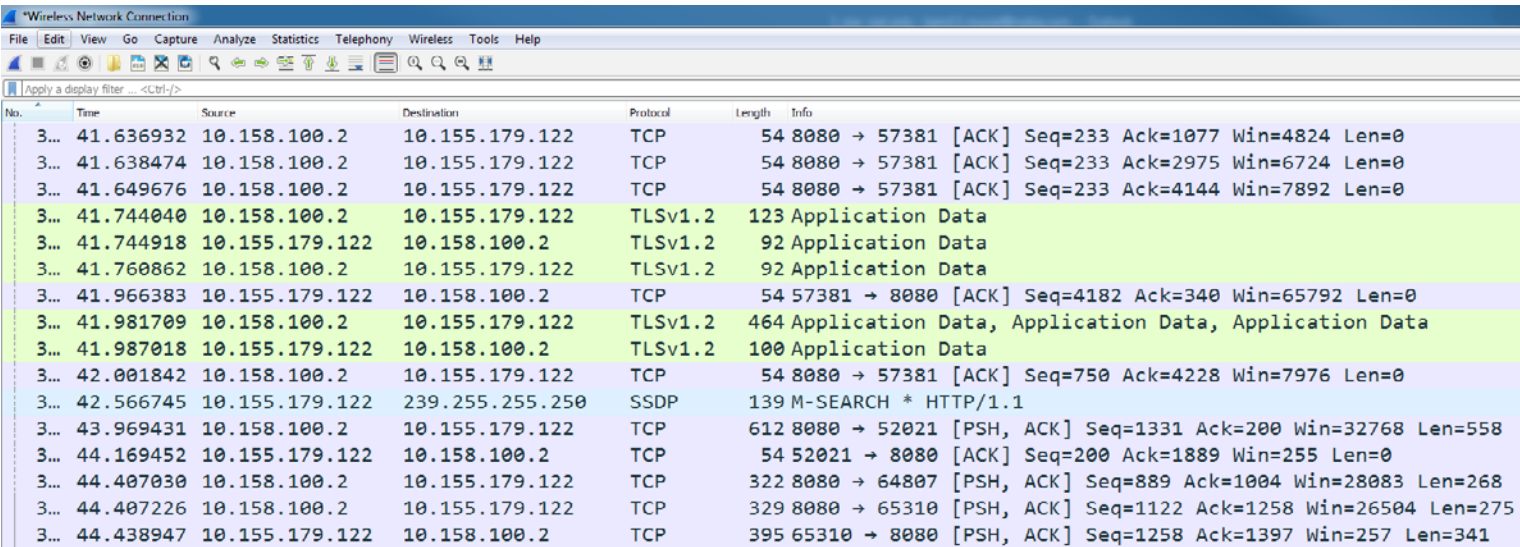
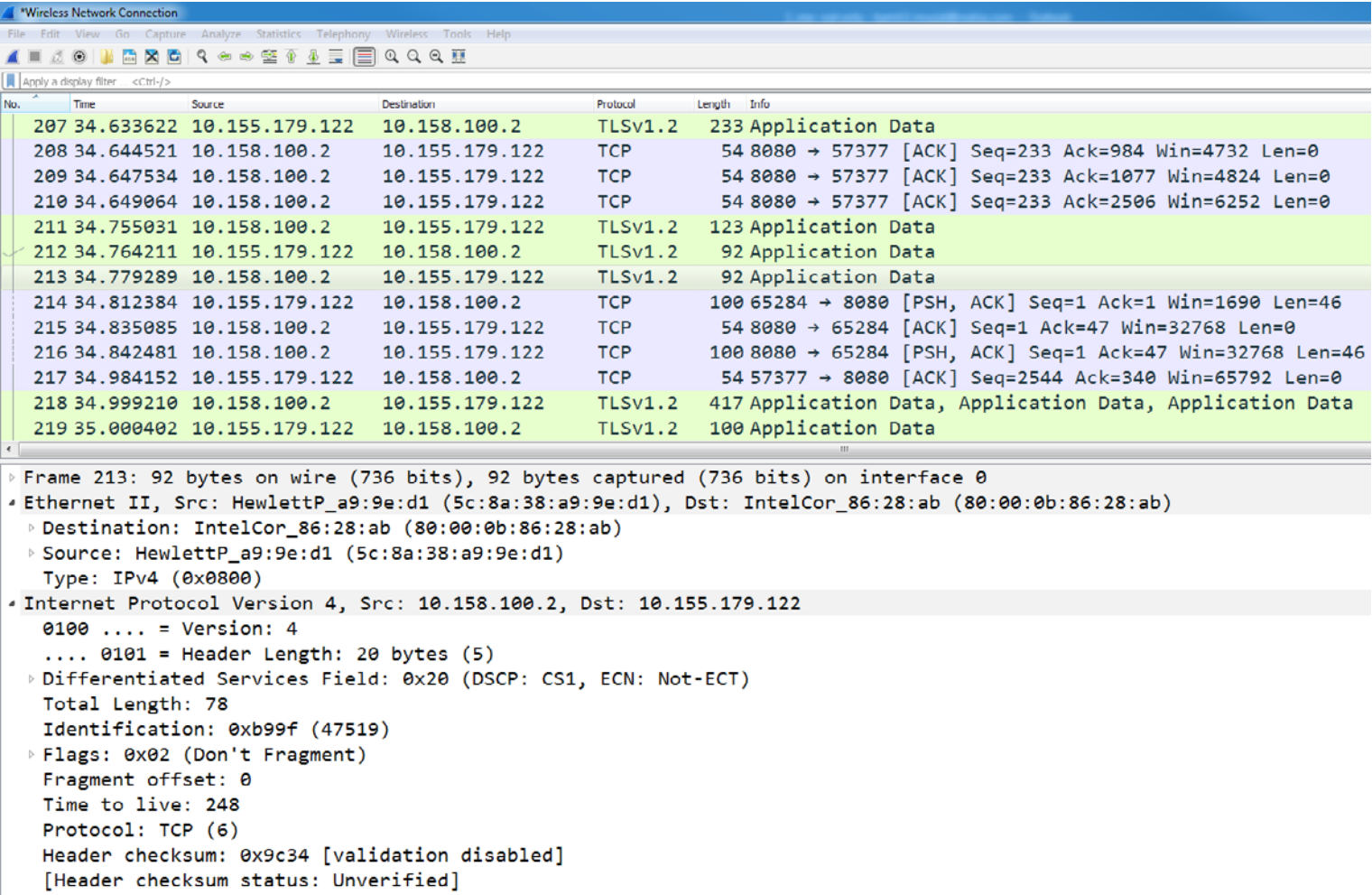


Figure 2 Detailed Wireshark dataflow



3. Nokia cases

In majority of Nokia appliances only low layers (1, 2, 3 and 4) are under investigation. During testing the correctness of connections between specific components must be checked. In case of problems thorough analysis that should be performed in order to find the root cause – the reason, why specific functionality does not work.

3.1 Wireshark in basic BTS connection investigation.

One of the main and most popular Nokia solutions is the BTS (Base Transceiver Station) that consist of:

- **System Module** – main managing computer
- **Radio Module** – device responsible for sending and receiving data to, and from user mobile phone.
- **ALD (antenna line device)** – additional devices improving signal quality

In Figure 4 the basic base station model has been presented.

System Module is connected to MME (Mobility Management Entity) via S1 link, which is a part of EPC (Evolve Packet Core). System Module can also be connected to another System Module via X2 link. System Module is connected with Radio Module via an optical link. Links S1, X2 and link between System Module and Radio Module are physical, optical fibers wherein digital data is transferred.

Figure 3 Filtered Wireshark dataflow

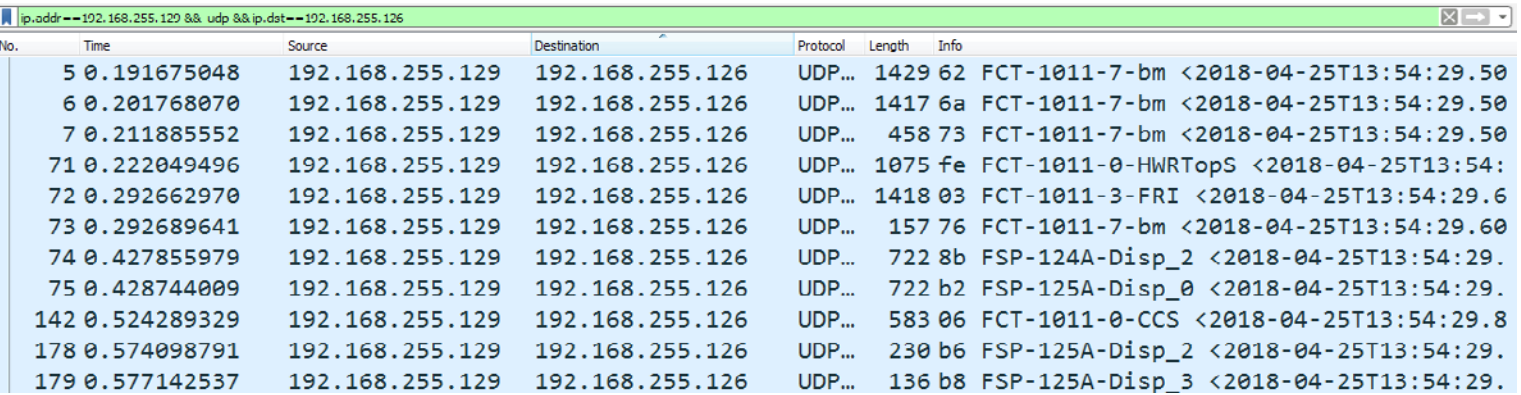


Figure 4 BTS overview

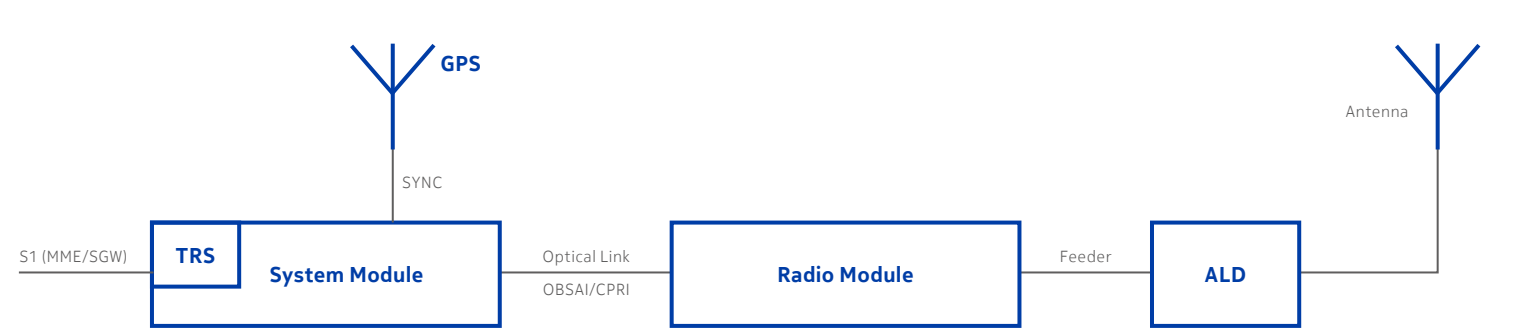
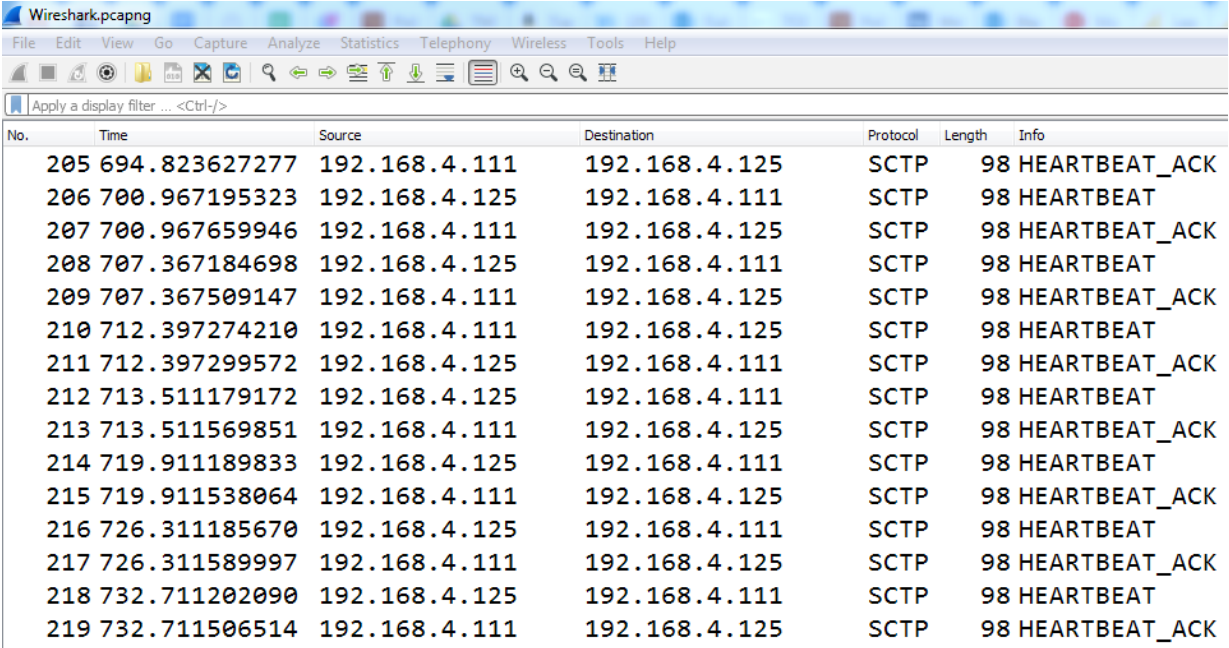


Figure 5 Wireshark filtered dataflow



A heartbeat message in signal processing is a message sent from an originator to a destination that enables the destination to identify if, and when the originator fails, or is no longer available. Usually, heartbeats messages are sent non-stop from the start-up until the shutdown.

In [Figure 5](#) you can see communication between two interfaces with the following IP addresses: 192.168.4.111 and 192.168.4.125. The heartbeat messages are sent and confirmed (ACK) which means that these two devices - in this case System Module and MME – are connected.

3.2 Wireshark in attach procedure.

Nokia, operating in the telecommunication industry – creates and develops LTE and 5G technology. When UE (User Equipment) – subscriber’s device – is switched on, the attach procedure is enabled. Below you can see an example of the attach procedure. On the top there are names of system components taking part in this procedure. Arrows indicate communication process between devices.

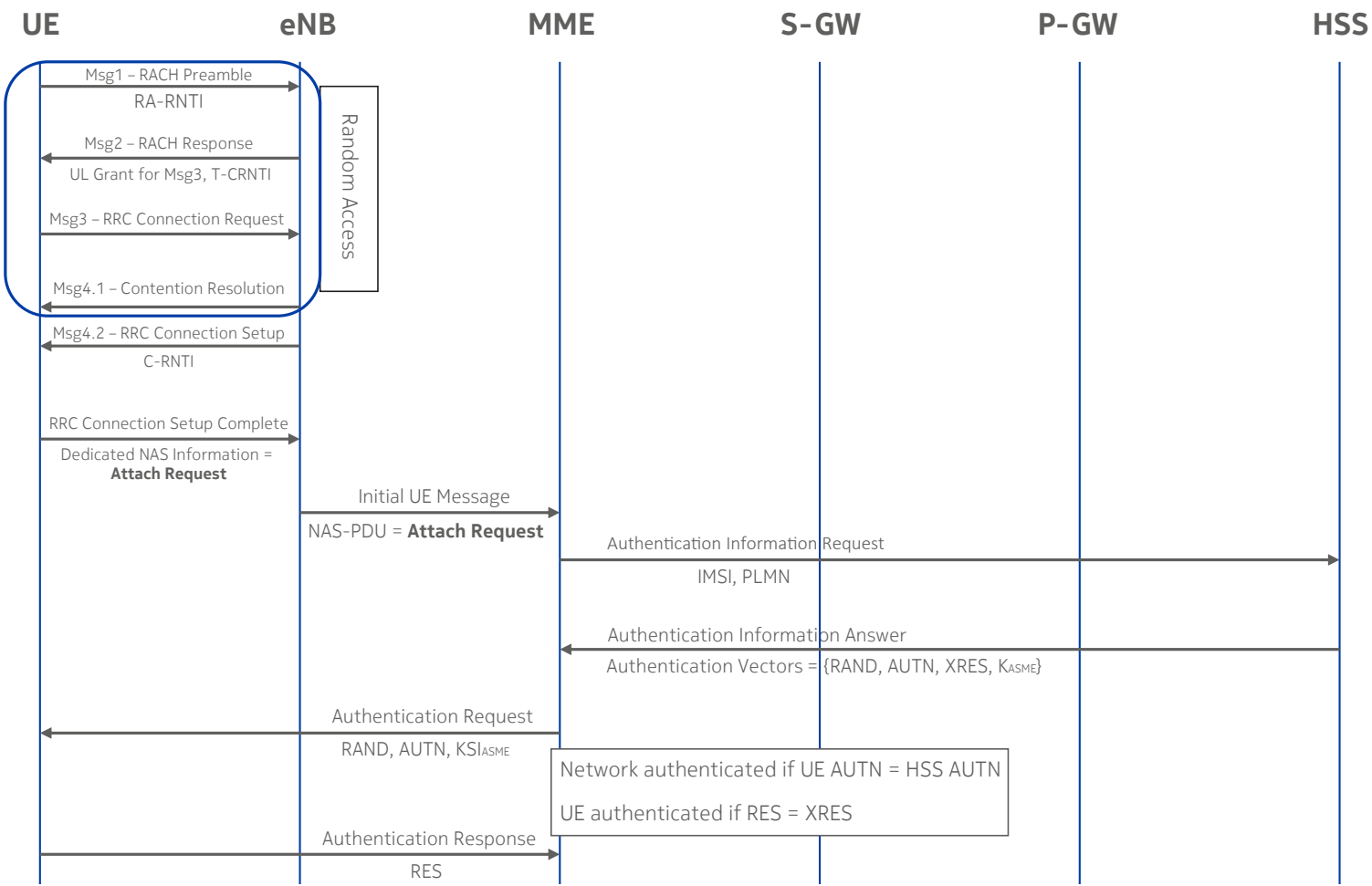
Setting Wireshark to a specific interface (IP address) allows you to capture and decode desired packets.

In the picture above you can see packets from [Figure 6](#). Presented Wireshark measurement captured all packets between eNB or UE and MME. It is clear that messages presented in [Figure 6](#) – Attach Request, Authentication Request and Authentication Response are present and have been successfully performed. Also, in [Figure 7](#) in packet no. 10 you can see that the packet retransmission has been executed. Following such an analysis you can easily investigate the attach procedure to find, where communication is interrupted.

3.3 Decoding – Lua scripts

Lua is a script language used for extending functionalities of various applications. Wireshark, in order to decode packets within specific technology, needs information what this specific packet means. [Figure 5](#) presents Wireshark screenshot with no additional tabs, only the basic information has been shown. In [Figure 7](#), you can see messages names that are characteristic for attach procedure in LTE technology. In order to obtain these decoded packets, the specific lua script was necessary. Nokia has invented and is still developing its own sets of lua sctipts for 5G, LTE, IoT, WCDMA and GSM technology.

Figure 6 Attach procedure



LEGEND:

UE – User Equipment (mobile phone or simulator)

eNB – evolved NodeB (base station)

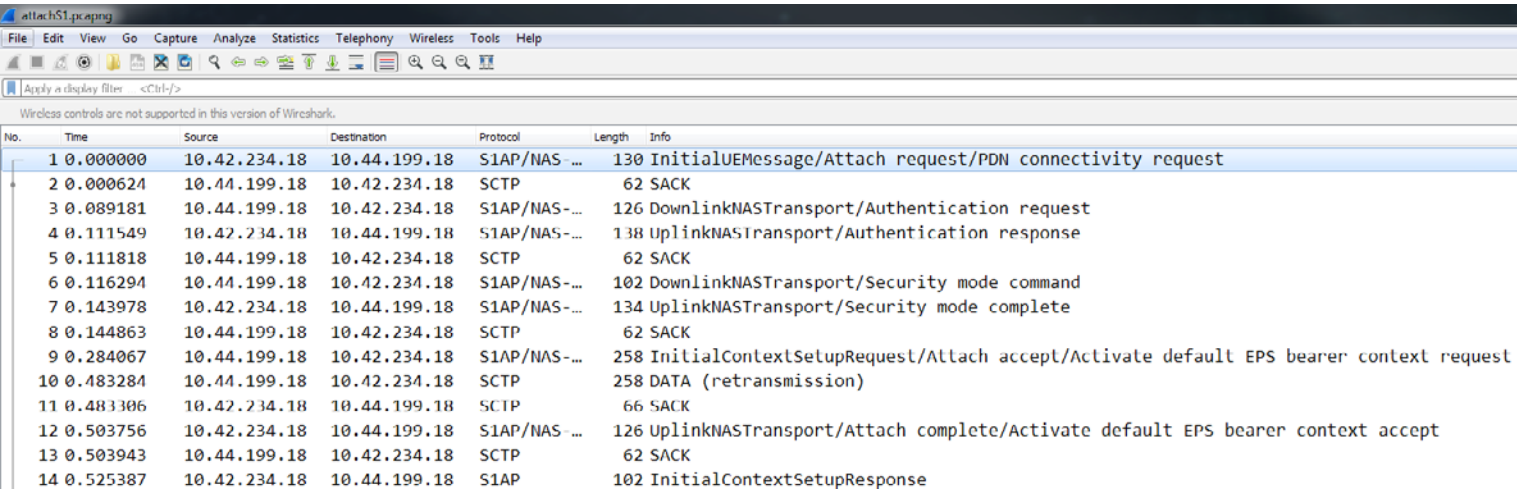
MME - Mobility Management Entity

S-GW – Serving Gateway

P-GW – Packet Data Network Gateway

HSS – Home Subscriber Server

Figure 7 Wireshark attach procedure



4. Summary

In network area, a tool that shows what happens between specific interfaces and investigates problems if necessary - so called - sniffer is mandatory. As shown above, Wireshark lets the user capture and analyze data flow, from basic flow presented in a simple view, to extended and thorough investigation of specific packets, between specific interfaces. Among all sniffers available, Wireshark is an opensource, cross platform, and very customizable tool which gives it a great advantage over other available programs and because of that, it is widely used in Nokia Network Company.

References

- [1] Brown. (1987-12-01). *USA Patent No. 4,710,926*.
- [2] Hoffman, C. (2017, June 14). *How to Use Wireshark to Capture, Filter and Inspect Packets*. Retrieved from [www.howtogeek.com: https://www.howtogeek.com/104278/how-to-use-wireshark-to-capture-filter-and-inspect-packets/](https://www.howtogeek.com/104278/how-to-use-wireshark-to-capture-filter-and-inspect-packets/)
- [3] The DNS Protocol. (n.d.). Retrieved from <http://www.firewall.cx/>

About the author

I've been working for Nokia for 4 years – for the first 2 years as a Software Integration Engineer for the LTE technology and now I work for 5G department as a Continuous Verification Specialist. My job involves testing new functionalities and investigating occurring faults. My responsibility – besides SW integration and verification – is taking part in various internal and external projects like conferences, cooperation with Universities, Nokia Academy and so on. Combination of engineering work and “social” activities is what keeps me going.

Kamil Musiał

Continuous Verification Specialist
MN 5G RAN

CI Through Project Life Cycle

Tatsiana Viarbitskaya
Local Product Owner
MN SR

Grzegorz Czaiński
Verification Architect
MN SR



1. Stepping into an IT project

The main goal of an IT project with destination for commercial market is to create a product from which its producer will benefit. To be competitive, the product needs to meet the following requirements:

- Quick time-to-market,
- Increase in customer satisfaction,
- Ensured adaptability of the delivered solution.

The above-mentioned criteria are equally applicable to projects focused on developing services, standalone applications, embedded software, etc. Meeting final customer expectations requires continuous development which leads to continuous integration. Different aspects of continuous integration have been described in “Modeling continuous integration practice differences in industry software development”. [1] Although the article written by Daniel Ståhl and Jan Bosch is rich source of information, the approach to software development and testing is constantly evolving to face a more and more demanding market.

2. In the beginning, there was chaos

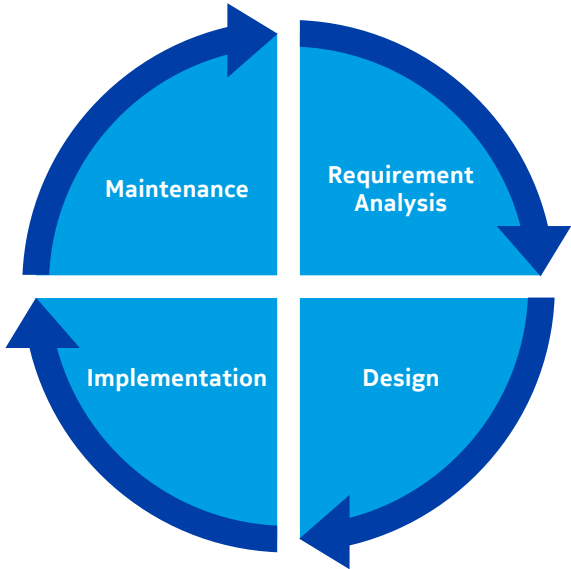
First words describing the origins of the world in Greek Mythology can be applied to any IT project. In the beginning, there is an idea with a general definition of goals. Through time projects become more and more specified to finally reach a point when they’re close to an evolving living organism. Project life cycle can be divided into stages, as shown in [Figure 1](#).

For each project, life stage tests need to be focused on a different aspect to ensure that the final product meets customer requirements. How to implement Continuous Integration (CI) into project life cycle, so it would satisfy all needs of each development stage? To answer this question, it is necessary to first understand what activities should be performed at each stage and how can CI support it.

A bunch of necessary definitions before diving into the project life cycle:

- **Continuous Integration (CI)** – a development practice focused on integrating each code change with a common code repository and executing a set of tests for verification purposes. The primary goal of CI is to detect issues by testing the smallest possible portions of code as often as possible.
- **Shift left** – an approach which allows finding and preventing defects early in the software delivery process. Quality improvement is reached by moving tasks early in the lifecycle (moving left in the project timeline). Testing needs to be performed from the earliest phases of software development. [11]

Figure 1 Project Life Cycle



- **Test framework** – mostly a set of guidelines for creating and designing test cases. It is a conceptual part of automated testing that helps testers to use resources more efficiently. [3]
- **Tools** – mechanisms (e.g., earned value computations) applied to inputs (e.g., task results) to create outputs (e.g., a progress report)
- **Process** – “a series of actions bringing about a result”, while a result is a “concrete outcome.”
- **Project** – a specific set of activities designed to accomplish a singular goal. It has a defined beginning and end in time, and therefore limited scope and resources. [4,5]

3. CI in project life cycle

Based on Continuous Integration and Shift Left definitions, we can say that testing in a project needs to start as early as possible and must be performed even for the smallest changes.

4. Requirement Analysis

At this stage business requirements and a way to meet them are defined. Moreover, it is identified what products need to be delivered, what resources are needed and what set of skills is necessary for completing the work .

Although during the requirement analysis stage main work is performed by project managers, test architects and engineers should use this time for necessary analysis related to tools, frameworks, software development, and testing strategy. At some point, you need to decide whether to create Continuous Integration from scratch, or reuse a known solution. **Tables 1, 2** and **3** present both these approaches, as well as a combination of them.

Table 1 Defining the CI chain with reuse of known solutions

	Advantages	Disadvantages	Recommendation
Reuse of available tools, frameworks, and processes.	Tools functionalities and limitations are well known.	Tool limitations may affect development and testing of functionalities mandatory for new projects.	One-to-one reuse of available resources (tools, frameworks, processes, and other methods) is useful when a new project is based on an already developed and maintained product – usually, a percentage of source code is inherited.
	Tools and frameworks are integrated and ready to use.	Available frameworks and procedures may not be scalable for new requirements.	
	Development and test environment issues have been solved in an earlier project.	Tools and frameworks may have a high entry level for new employees.	
	Engineers are familiar with processes and way of work.	High costs of modernization in case of significant changes in specific project.	

Table 2 Defining the CI chain from scratch

	Advantages	Disadvantages	Recommendation
Creating Continuous Integration based on new tools, frameworks, and process redefinition.	Tools and frameworks can be selected to cover all requirements related to development and testing of project functionalities.	Integration of tools and frameworks requires additional time for solving issues related to test and development environment.	This approach is applicable in two situations. First: when you are starting a new project and there are no resources that could be reused. Second: when you are starting a new platform as a base for further projects. Because introduction and deployment of tools, frameworks, and processes requires significant time and resources, this approach is not recommended for projects with a challenging time-to-market.
	Framework and defined processes suit project needs.	New processes and way of work require time and resources for introduction and successful deployment.	
	Tools and frameworks can be chosen in a way which ensures a low entry level for unexperienced employees.	The legacy test may not be reusable.	

Table 3 Defining the CI chain with new tools and frameworks, as well as process redefinition

	Advantages	Disadvantages	Recommendation
Partial reuse of tools, frameworks, and processes.	Tools with limitations which do not allow for project functionality development and testing can be replaced within an existing framework.	Integration of new tools and frameworks with available resources requires time for solving issues related to development or test environment.	Due to reuse of some elements of the CI chain, it is possible to use this approach in a project with a medium to long time-to-market. Partial reuse is also possible in projects with short time-to-market, but you need to ensure that reused tools, frameworks, and processes don't have any dependencies on elements of the CI chain which you are not going to implement in the project CI.
	Legacy resources which meet project requirements can be reused to save time.		
	Tools and frameworks with a high entry-level can be gradually replaced with more commonly used tools and frameworks.		
	Issues with tool integration are limited to areas where changes have been made.		

The second activity in which engineers need to be involved is reviewing all available project related requirements. That way any limitations and bottlenecks will be discovered at the earliest stage possible and they can be properly addressed during project planning. Every time project related documentation is updated, a review session needs to be performed. Usage of static testing techniques like review, walkthrough, or inspection is quite effective when done by experienced engineers. Because static testing does not require test line or writing source code, it is also considered a cheapest way of detecting issues [6].

5. Requirement analysis exit point

At the end of the requirement analysis stage, you should have visibility of: project time-to-market, functionalities requested by customers, and similarities to already developed projects. Use this knowledge to predefine future Continuous Integration aspects, like:

- Potential for reuse
- Risks related to the introduction of new tools, frameworks, and processes
- CI steps
- Speed and order in which CI steps need to be created and ready to use
- Main areas on which you will need to focus during further project development
- Type and number of needed equipment

Although assumptions you will make at this stage may not be 100% accurate – they will be a good starting point for designing a phase where requirements towards Continuous Integration will be defined.

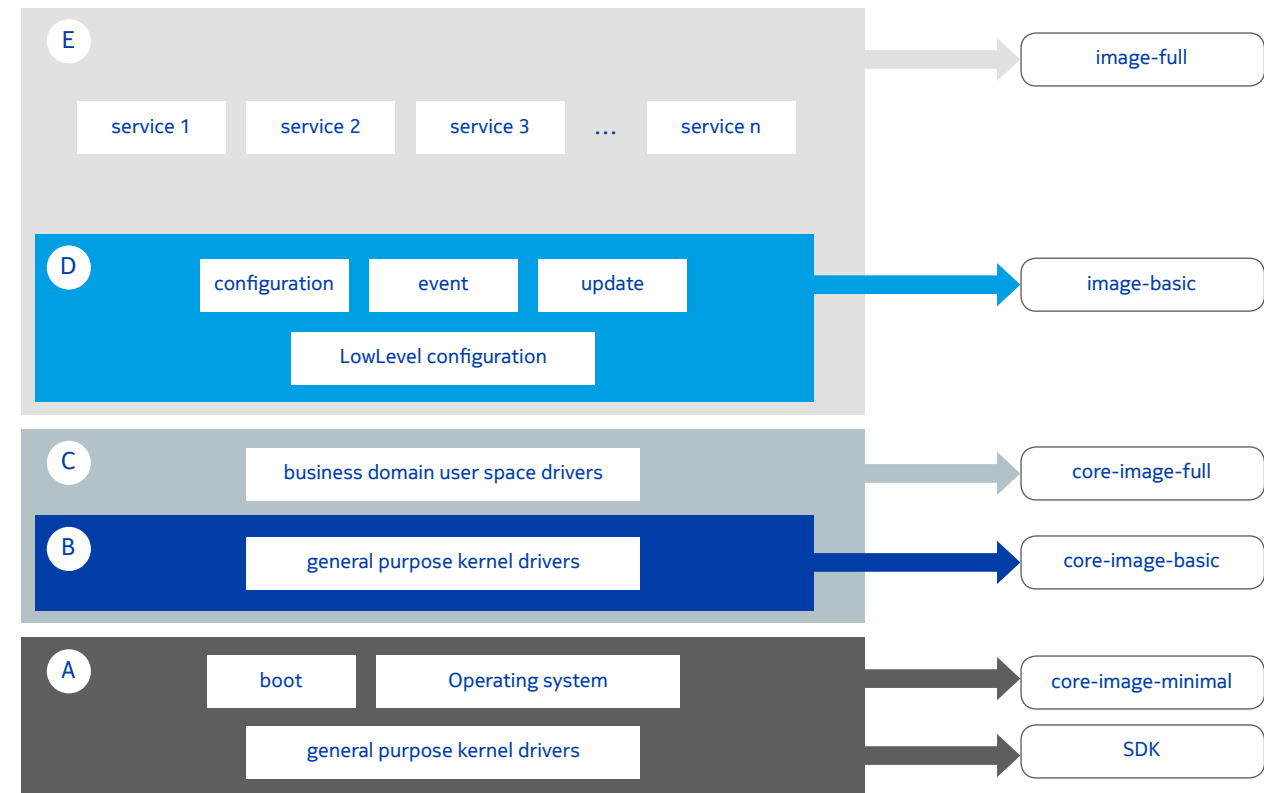
6. Design

The Design stage is dedicated to defining solutions to meet requirements set by the customer. During this stage, software, hardware, and system architects decompose the assigned project into platforms and components. They define communication interfaces, hardware, and software procedures, as well as software and hardware components that are reused. At this point, the project starts to become a product. For engineers responsible for designing and future implementation of the Continuous Integration chain, this is a crucial point. This is due to the fact that based on product architecture they define the complexity of CI, the function of each step, and acceptance criteria which need to be met on each stage – from committing to releasing.

The system presented in **Figure 2** has a layer structure like the ISO/OSI model, where an upper layer is creating services based on resources provided by lower layers. For that kind of architecture, Continuous Integration needs to ensure:

- A build system that allows creating software image with different versions of components on each layer.
- Independent development and verification of a single component within a single layer

Figure 2 An adapted concept of OpenEmbedded systems [7]



- Independent integration of components within a single layer
- Functional and nonfunctional verification of a single layer
- Integration of layers
- End-to-end verification of the whole system

Meeting all of the above criteria allows creating a multilevel CI chain, which thanks to testing performed from a single component level through layer integration to full system verification, ensures early issue detection and short feedback time. Functional, nonfunctional, structural, and regression tests apply to this approach. Implementation of any CI shell requires a set of tools. Those aspects are described in CI as a toolbox.

7. Testing in the design phase

In projects where both software and hardware are developed, it is a common practice to test potential solutions on a simulated target environment before the final implementation. This approach allows early detection of potential issues and choosing the most efficient solution without involvement of expensive equipment. This stage

of design is often called the proof of concept phase. It is essential to remember that all defined tests and acceptance criteria must be applicable for simulator-based test environments as well as for the target product. It saves time during the implementation phase.

8. Design phase exit point

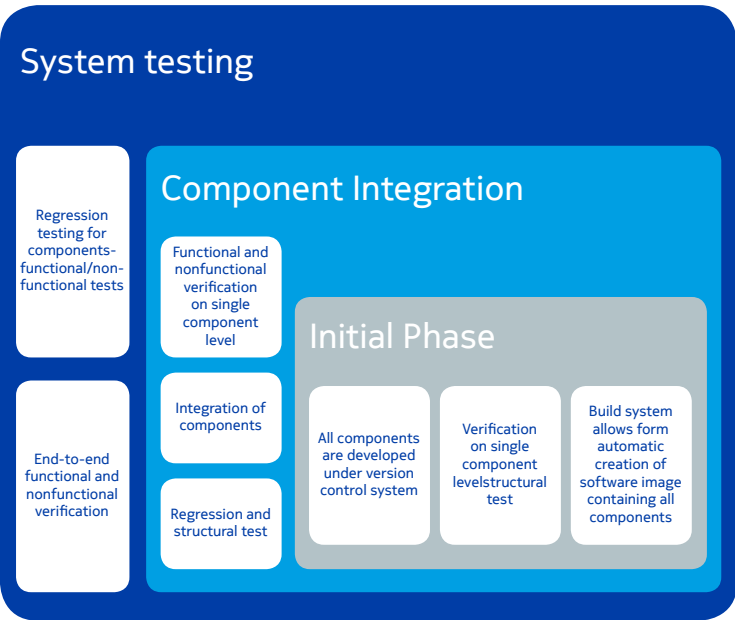
During this stage of project life cycle assumptions from the previous phase have been refined and aligned with product architecture. Based on the specification provided by architects, Continuous Integration steps have been defined along with acceptance criteria and necessary tools. It's time to start the implementation phase.

9. Implementation

During this phase features requested by the customer are implemented into the final product. It starts development of software, and if needed hardware. To ensure proper quality of the final product, continuous practices need to be implemented. Up till this

point, work on the project related CI chain focused on analyzing requirements, tools, available frameworks, and designing a test chain to cover product architecture. Although testing was limited to static methods and manual tests, each issue found in requirement analysis or design phase was cheap to correct. Rules related to testing as early as possible and integrating smallest possible elements have been fulfilled. Transition to the implementation phase of CI should be based on plans and designs prepared in earlier phases. Along with source code development, all aspects of Continuous Integration evolve. It is essential to implement a version control system over source code and a build system at the beginning of product development – at the implementation phase. Having those two elements from the beginning allows more natural control of product development and integration into one image. Testing aspect of CI needs correlating with the speed of product implementation. Tests for product functionalities should be available before implementation of the feature to main software line – trunk. Test-driven development (TDD) allows active issue detection by confronting code implementation with tests written based on the same requirements. Scope of tests executed within CI is changing together with product development. Verification on the component level may be possible only at the beginning. While the product is under development, next levels of testing defined during the project design phase can be enabled. **Figure 3** shows how Continuous Integration may evolve along with project development.

Figure 3 Project CI evolution during implementation phases



10. Implementation exit point

Project implementation stage ends when the product is delivered to the customer. From the technical point of view, it means that all required features have been turned into source code, have been tested, and have met acceptance criteria. Consecutive iterations of CI allow detecting and fixing issues in the developed product.

11. Maintenance

Maintenance – a phase, which is not in the scope of the Project Manager, but is essential for the customer and from the end user point of view. Well planned and implemented CI decreases the risk of corrections on faults revealed after delivering the final product. During maintenance more significant weight is placed on regression and finding issues related to exceptional border cases. From the customer point of view, at this stage of Project Life Cycle Continuous Integration especially the testing part is facing challenges of fast verification (with full regression) and delivering a correction to the customer.

12. CI as a toolbox

Continuous Integration as a process of automatic building and testing of code requires the use of various types of tools like a version control system, build system, automation server, test framework, and more. Remember that there is no such thing as one ultimate kit that always works for all projects, including yours. Consider the CI chain as a toolbox which needs to contain tools allowing you to get the job done.

Before selecting a specific tool, define a set of requirements that must be met by the potential solution – it helps to choose the best option for you. Below are some questions that may help you with selecting right tools for your CI:

- What is the entry level for users and administrators?
- Is it an out of the box solution or does it require an additional configuration and or update?
- Is It Open Source or Commercial?
- Is the tool commonly used or internal?
- What are purchase, deployment, and maintenance costs of the selected tool?
- Is there support available and what does it look like?
- Does the tool require a specific operating system (OS)?
- How much system resources does the tool require?
- Can it be run in a cloud environment?
- Can it be easily equipped with new functionalities?
- How many specialists familiar with the specific tool are available on the market?

It is critical not to get attached to one set of tools connected via case specific interfaces because at some point you will face a situation when the CI toolbox requires reorganization. If this happens at the begging of a project, tool replacement may be done relatively easily because the number of users is rather small. The situation becomes more complicated when you are doing a significant number of commits, the CI chain is working and can't be stopped. In this case, switching to a different tool takes some time, and it requires a transitional period when two solutions need to be supported. Before you swap tools in your CI, ensure that:

- The new solution meets all your requirements,
- The change does not cause any functional regression in CI,
- Potential time break in the CI chain is minimal,
- The new tool has potential to be further developed in the future,
- Added value (for example reduced test duration or ability to perform more complex test cases) is introduced in CI.

Although changing anything in CI is not as easy as replacing a screwdriver with a wrench, if you see that a tool has reached its limit and does not allow doing something more efficiently, you should replace it and move forward [9].

13. Conclusions

Continuous integration and all continuous practices have become an important area of software engineering research and practice, as well as a project management area of interest. There are developed approaches, tools, methods, and yet still CI needs to evolve to meet the need for delivering high-quality products while continually reducing time-to-market.

14. From The Authors

Dear Reader, thank you for reading this article. We hope that we have managed to bring you closer to the subject of Continuous Integration, especially how it has evolved from raw requirements into a system ensuring quality for a whole product. If you are interested in a deeper dive into CI, we strongly encourage you to study all materials from reference section.

References:

[1] Ståhl, D. and Bosch, J. “Modeling continuous integration practice differences in industry software development,” Journal of Systems and Software, vol. 87, pp. 48-59, 1, 2014.

[2] P. Suzie. “The Product Managers’ Guide to Continuous Delivery and DevOps, Available <http://www.mindtheproduct.com/2016/02/what-the-hell-are-ci-cd-and-devops-a-cheat-sheet-for-the-rest-of-us/>

[3] “Test Automation Framework”, [Online]. Available: <https://www.techopedia.com/definition/30670/test-automation-framework>

[4] A Guide to the Project Management Body of Knowledge (PMBOK® Guide ver 6th)

[5] Duncan, W. R. (1993). The process of project management. Project Management Journal, ISSN: 1938-9507, 24(3), 5–10.

[6] Minnesota Office of Continuous Improvement, [Online]. Available: <https://mn.gov/admin/continuous-improvement/resources/projects/toolbox/>

[7] Peregudov, O. (2018) RFSW Archeo RoadMap (adapted view), pp. 51-53, 2018

[8] Shahin, M., Babar, A, Zhu, L, “Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges, and Practices,” IEEE Access, ISSN 2169-3536, pp. 3909 – 3943, 2017.

[9] “Yocto Project Development Manual”, [Online]. Available: <https://www.yoctoproject.org/docs/1.8/dev-manual/dev-manual.html#understanding-and-creating-layers>

[10] “Certified Tester Foundation Level Syllabus”, [Online]. Available: <https://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html4>

[11] “What is “Shift Left”? Shift Left Testing Explained”, [Online]. Available: <http://www.bmc.com/blogs/what-is-shift-left-shift-left-testing-explained/>

About the authors:

I started work in IT in 2010 as an Integration and Verification Engineer. During my career in Nokia, I’ve gained experience as a tester, Integration Leader, Product and Process Owner. Currently, as a Verification Architect, I’m responsible for defining, implementing, tracking processes and practices related to Test Strategy at the department level.

Grzegorz Czaiński
Verification Architect
MN SR

I started my career in Nokia in 2015. Now I work as a Product Owner in MN SR SW R&D WRO Architecture Team, where we are creating Verification Architecture Core. I also actively participate in various international and national conferences about signal processing.

Tatsiana Viarbitskaya
Local Product Owner
MN SR

Automation of Software Testing in LTE Environment

Przemysław Ratajczak
Verification Engineer
MN 4G RAN Pz WRO

NOKIA

Introduction

End-to-end testing of LTE software is by no means a simple task. The testing environment consists of many elements which themselves are quite complicated. To make it all work together we have to plan and execute things carefully. Fortunately, as we'll see, it's a great environment to automate which will make testers' work much easier (well, at least when it comes to mundane and repetitive tasks).

Our testing environment

The most crucial element of our environment is of course the eNodeB – the base station on which the tested LTE software is installed. The eNodeB is connected to our fully functional LTE network with its complete infrastructure. We keep our UEs (short for User Equipment devices, which are mostly prototype or market mobile phones) in shield boxes to separate them from the surroundings and prevent unwanted interferences. Just like in every LTE network, the UEs attach to different cells and perform voice calls or data transfers. Usually we use several base stations configured together to enable handovers between them – it simulates an end user moving between different LTE cells.

The eNodeB is connected to another important part of our testing environment – NetAct. NetAct is a comprehensive management software suite for operators which allows operations such as eNB software upgrade, configuration change, or fault monitoring on many base stations simultaneously.

Apart from the LTE part of the environment, we use separate auxiliary testing infrastructure. We need PCs to connect the eNBs to BTS Site Manager – software used for local management and troubleshooting. This software is usually used by technicians working on-site when dealing with a single base station. We also need a local PC to collect system logs from the eNB, handle connected mobile phones and run additional scripts.

Another part of the testing environment is Dockerized Robot Framework. This is one of the most important parts of an automated testing setup. We could call it the heart of our solution, as it's the central part of all testing. Robot Framework is a Python-based, keyword-driven test automation framework which we'll discuss it in detail later.

Moreover, there are PCs which gather test logs, monitor memory usage, analyze results, generate reports and pass all required data to software tools such as HP Application Lifecycle Management. If any faults are identified, we can report them using our internal Pronto Tool (for reporting errors related to LTE) or Jira (other test environment issues).

Additionally, there are many supporting tools and systems, which provide new software releases, handle script repositories and libraries.

As you can see, this testing environment is quite complex, but it's possible to automate many components. We're aiming at a total automation system, where testers only use their expert knowledge to review failed cases and all repetitive tasks are performed automatically.

The testing process

The end-to-end system testing is one of the last steps of testing – software and its components have been already tested thousands of times in different development phases. We take the entire product (LTE software) and test it in our environment, trying to replicate real-life conditions and our customers' configurations.

Let's take a look at the testing process – bear in mind that it is a simplified view, and many additional elements must be taken into account.

Let's consider a simple manual test. Typical steps taken during the testing process would be as follows:

1. Preparation

Manual procedure

The tester connects to the eNodeB using BTS Site Manager and checks if the base station is working correctly, and there are no initial errors. The tester makes sure that logs are being collected. Next, we check the latest version of software – quick tests should be passed and it should have a 'released' status. We find the download link for the software and related BTS Site Manager download (it changes together with LTE software). We download it to a local disc. In the meantime, we make sure that the current configuration is backed up to enable quick recovery in case of serious errors. Next, we start the upgrade process, making sure that the logs are already being collected (any faults during this phase are critical so it's important to have all the data). We also have to check if the connected UEs are operational and attach correctly. A tester has to repeat these tasks for all the test lines used.

Automation potential

As you can see, the preparation process is pretty straightforward – it contains many repetitive tasks and it can take considerable time. It's a great opportunity for automation, as there are no difficult decisions involved; it's repeated daily for many testers and test lines, so the time gains can be really impressive.

Let's see how it can be done from a technical point of view. At set time a Python script checks for the latest software for tested releases. We use Workflow Tool (WFT) which contains all required information related to software releases – status, release notes, download

links, restrictions, etc. Workflow Tool provides an API, which enables simple integration with scripts. The script downloads tested LTE software and related BTS Site Manager to a local disc where it's easily available to all testers. Next a Python / Robot Framework script checks if the eNodeB is fully operational and no alarms are visible. If there are any problems, a tester intervention is required to manually troubleshoot it. If everything's OK, the software can be upgraded, and new BTS Site Manager can be installed. Moreover, automated batch scripts make sure that the system logs are running and are continuously copied to a local storage.

The preparation phase is the best example of how simple solutions can provide significant time savings. Repetitive manual tasks can add up to several hours each week. Of course, we have to remember that it comes with the price – the environment may (and will) change and we have to maintain our automated tools, but usually the scripts are so simple that fixing them takes only a couple of minutes.

2. Execution

Depending on the planned tests, the tester may for example activate a new feature, change configuration, perform software rollback, etc. During the test the eNodeB should be monitored via BTS Site Manager and NetAct for unexpected and expected alarms. Test results have to be verified and recorded. If there are any errors, they must be thoroughly investigated and reported. Tests usually have to be run independently, therefore after the execution of each test the environment and configuration have to be restored to the original state.

Automation potential

As the tests of LTE software are usually time-consuming and are run repeatedly many times, automation is not an option – it's a must. Our experience shows that most tests can be automated. There are some exceptions where automation isn't feasible, but these are very rare cases.

For automated testing, we use Robot Framework solution. It is very easy to learn and the scripts (if properly written) can be understood even by engineers without programming knowledge. Moreover, it's a powerful solution which can be flexibly expanded with custom Python libraries, so it can be adjusted to local needs. We have our own LTE libraries which enable us to quickly write new tests. We use keywords which are easily understood, self-describing 'building blocks' such as: *Prepare eNB for Test*, *Check If Alarms From ENB and NetAct Are The Same*, *Check Expected Faults*.

There are many tests and libraries used by different teams. To make it all work smoothly, we have to use a coding standard [1] and a version control software [1][2].

- **Coding Standard**

All teams should use the same coding standard – a set of rules describing the test structure, naming conventions and general guidelines. It greatly increases the ability to maintain, modify and re-use existing test cases. Test cases should be as generic as possible – independent of hardware, configuration or team where they're used. Keywords, test cases and test logs should be easy to understand even without additional documentation – it's especially important when they're used by people outside the testing teams.

- **Version control**

All tests should be available in a central repository accessible to all testers – there's a great chance that somebody has already developed similar tests, and it's a great way of sharing knowledge and expertise, and saving valuable time. Using a version control solution enables us to track changes, revert to previous versions, and protects the tests against accidental deletion.

Another important aspect of our testing environment is that we run each of the tests in a separate dockerized container. One of good testing practices is to run tests independently [3] – the results of previous test cases shouldn't have any impact on other tests. In our Dockerized Robot Framework environment, we set up a new container for every test so there's completely fresh and clean system in every instance, which significantly reduces environmental problems and provides exactly the same conditions during every run.

Robot Framework is our main automation solution, but we also use other auxiliary tools to control UEs, simulate end user movement, and gather different logs used in fault investigations and debugging. Most of the elements in our environment provide command-line interfaces (CLIs) which provide easy access to standard operations

3. Result Analysis/Reporting

In a perfectly automated environment this is the step where the tester's knowledge and experience is needed. Of course, if all tests pass without any warnings or errors there's no need to investigate the logs, but every failed test should be thoroughly analyzed. We must stress that failed tests should NEVER be reported automatically [4]. We always have to find the root cause of the failure to avoid false positives.

A tester analyzes test logs. It's handy to have a tool which collects all the results and gives a brief preview of errors and warnings – we can quickly assess if there were some global problems or the system crashed at some point. In case of a failure the tester has to verify if it was an environmental problem or a software failure. Sometimes it's obvious – for example, all tests failed because of a network outage, but sometimes finding an underlying problem might require a long investigation.

If the failure is related to the LTE software, we have to report it using our Pronto Tool. The tester collects all the logs, describes the test and the failure and tries to identify the root cause. If it's an environmental issue, the tester tries to fix the problem or requests support.

Automation at this testing stage isn't viable. We can use scripts and tools to prepare the data, but all results, findings and reports have to be analyzed using expert knowledge. Only investigated faults should be reported, because the false positives are a big burden for investigation and development teams.

Automation isn't recommended during the investigation phase. However, we have a lot of archived test results and plenty of supporting logs – that's precious data we could use. The next natural step is using machine learning solutions to assist the tester in result analysis, fault identification, and quick reporting. If we are aware that this data together with tester's decisions can be used in the future, it will be much easier to implement such tools.

3.1 Testing Tips & Tricks

There are some simple tweaks and best practices which may greatly improve the software testing process:

- **Add randomness to tests [1].** A random element (within the test scope) may increase the test efficiency and find a bug which hasn't been found before.
- **Always use a timeout for test cases.** A test may freeze for different reasons. If there's no timeout other tests can't be executed and it's a huge waste of the resources.
- **Perform an automatic restart of failed tests [1].** After running all planned tests, we can automatically generate a list of failed tests and run them again. That gives us valuable information: Was it an environmental problem? Does the test fail each time? It helps us to save time, which we would otherwise spend trying to replicate the fault.
- **Run tests as often as possible [2].** If our test lines aren't used for reconfiguration or test development, we should always use it for running tests. With automated testing it's really easy – you just chose a test plan and run it. Some faults occur only in very specific conditions, so frequent retests increase the probability of finding them.
- **Keep it simple [4].** When it comes to automated software testing, it's best to simplify as much as possible. The tests should be easy to understand which increases their maintainability. If the test case is too complicated, we should

break it into separate tests (if possible). We shouldn't use complex conditional structures – the test cases should be linear.

- **Do not use pauses [1].** It's one of the important best practices. We shouldn't use set values when the script waits for something. The environment, software and configurations change with time. What's good now can be different in the future. Instead, we should use keywords such as Wait Until Cells Are OnAir, Wait Until The Alarm is Visible etc.

- **Logged errors should be informative [1].** Sometimes test reports don't clarify what went wrong. If we provide a meaningful error comment, it significantly facilitates the log analysis.

- **Don't leave a commented-out code in the final version of a test.** It's a very bad practice to leave old code commented out. It may seem that it may be needed in the future or it's a temporary workaround but it the long run it really clutters up the test – it decreases readability, may be misleading and leads to code degeneration. Instead, we should use source control, which is a much better solution.

- **Use old environment images for regression testing.** In case of regression testing where all test cases are finalized and aren't changed anymore, we can use a 'frozen' image of a system on a given day. It eliminates a lot of environmental problems, e.g. when an external Python library is changed and causes errors that are hard to identify. It provides an additional layer of stability in the system.

4. Summary

Test cases which should be automated are those, which are often repeated, tedious, and time-consuming. The test automation provides better testing quality, reduces the time and costs of software testing, but can also provide additional benefits which are hard to achieve in manual testing [5]:

- Night and weekend testing. The test lines can be used 24/7.
- Increased ability to reproduce software defects by providing exactly the same conditions.
- Ability to perform stress and performance testing.
- Elimination of the so-called *tester's fatigue* – repeated manual execution of mundane test may lead to serious errors.

On the other hand, not all tests can be automated – the effort might be too big compared to the potential benefits. We also have to remember that it's not a maintenance-free solution – our environment is a complicated system, which constantly changes and requires ongoing maintenance. Time and cost savings usually come with time when the system has matured [2][5].

References

[1] Alpaev G., Software Testing Automation Tips: 50 Things Automation Engineers Should Know, Apress 2017

[2] Goucher A., Riley T., Beautiful Testing, O'Reilly Media, Inc. 2009

[3] Posey B., Mosley D., Just Enough Software Test Automation, Prentice Hall 2002

[4] Pettichord B., Bach J., Kaner C., Lessons Learned in Software Testing: A Context-Driven Approach, John Wiley & Sons 2001

[5] Gauf B., Garrett T., Dustin E., Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality, Addison-Wesley Professional 2009

About the author

I'm a graduate of Wroclaw University of Economics. I joined Nokia after taking part in the Nokia Academy program. I work in the Customer Specific Verification team where I develop automatic tests in Robot Framework and scripts in Python.

Przemysław Ratajczak

Verification Engineer
MN 4G RAN Pz WRO

Throughput testing techniques and challenges in modern LTE-Advanced Pro networks

Mariusz Zamłyński
Verification Architect
MN 4G RAN

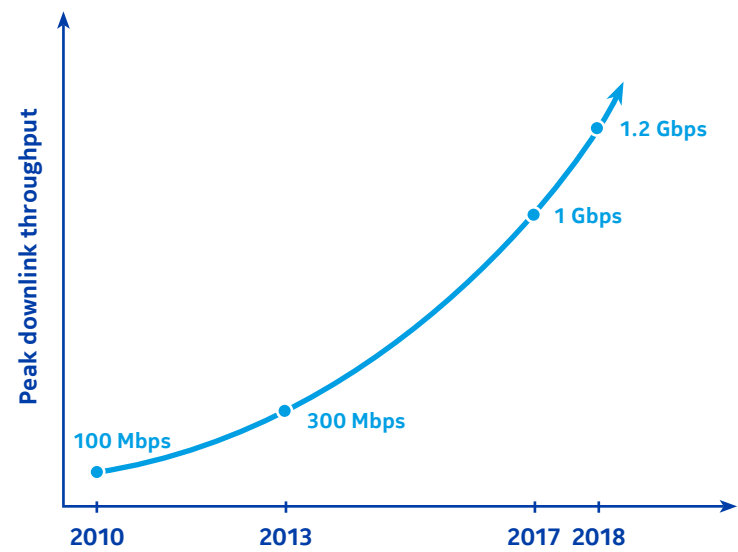


1. Introduction

Throughput is a metric which describes how much information (i.e. data) can be transferred through the network in a particular period of time. In case of telecommunication systems, providers define how many bytes during a second can be downloaded from a server (downlink speed) or uploaded to the network (uplink speed) by an end user. Marketing materials provide information about the maximum download data speed, which is called the peak downlink throughput for a single user. This value is measured in the best possible conditions, i.e. when all available resources are scheduled to the same user which located near to the eNodeB's (eNB) antenna.

The firsts LTE networks introduced in 2010 supported only Category 3 devices [1], which offered download speed up to 100 Mbps (see, [Figure 1](#)). The first Category 6 UE, which is compatible with 3GPP Release 12, was released in Korea at the end of 2013. This device supported Carrier Aggregation (CA) of 2 bands, which translates to 300 Mbps maximum throughput. It started the evolution of LTE networks to LTE-Advanced. The last LTE-Advanced enhancements were commercially introduced with Category 16 devices in Q1 2017. Combination of CA (up to 5 components), 4x4 MIMO and 256 QAM allows operators to provide 1 Gbps class networks. The firsts LTE-Advanced Pro networks were introduced in 2018 together with Category 18 UEs offering up to 1.2 Gbps downlink throughput. This technology will be improved by 2025, and offers up to 3 Gbps speed. Faster transmission will be provided by aggregation of higher amount of carrier components from both licensed and unlicensed bands. LTE-Advanced Pro networks will provide a smooth transition to 5G New Radio technology, so they are often called as 4.9G or pre 5G.

Figure 1 Maximum downlink throughput in LTE networks over years.



Continuous development of mobile networks requires enhancements in throughput measurement techniques. The general methodology has not been changed in recent years. In lab environments, it is still possible to apply methods consistent with RFC 6349 [2]. However, this technology is complicated and time consuming. Therefore, it cannot be applied directly to throughput measurements in mobile networks. Engineers who are working on developing LTE networks use for throughput testing purpose FTP and iPerf techniques, which are simpler and can be adopted to achieve expected results. Additionally, end users of live commercial networks use mobile applications called SpeedTest, which are dedicated to simplified benchmarking of LTE networks.

2. Factors affecting throughput in LTE networks

In case of mobile networks, the maximum theoretical throughput is limited by the radio communication channel capacity. This limit is described by the well-known Shannon's formula [3]:

$$C = B \cdot \log_2 \left(1 + \frac{S}{N} \right) \quad (1)$$

where C is the channel capacity, B is the channel bandwidth and S/N is the signal to noise ratio. This formula can be generalized for multiple antenna systems:

$$C = M \cdot B \cdot \log_2 \left(1 + \frac{S}{N} \right) \quad (2)$$

where M is the number of antennas used during transmission in MIMO spatial multiplexing mode.

The channel bandwidth used by the LTE system depends on the number of resource blocks scheduled for the particular user. It is specified that a resource block consists of 12 subcarriers spaced by 15 kHz each. Therefore, the bandwidth occupied by one RB is 180 kHz. In case of 20 MHz channel, the LTE system uses 100 RBs, so only 18 MHz is used during data transmission. This limitation was overcome by Carrier Aggregation technique. Modern UEs and eNBs allows operators to aggregate up to 5 cells. As a result, 90 MHz effective bandwidth may be utilized.

The signal to noise ratio determines which Modulation and Coding Scheme (MCS) may be used for data transmission with assumed Bit Error Rate (BER). Better channel quality (i.e. higher S/N) allows eNB to select modulation with higher order and more efficient coding rate. 3GPP Release 12 introduced 256 QAM modulation for downlink direction which provide 30% higher maximum throughput than 64 QAM modulation at the same radio conditions. Particular MCS is selected by eNB basing on Channel Quality Indicator (CQI) reported by UE. The CQI value depends on the estimated S/N ratio, but the exact formula is unknown.

3. Throughput measurement techniques

3.1 Test environment setup

The previous section describes radio interface factors which have an impact on throughput. However, the measurement of data transmission speed is done in an end-to-end environment. Therefore, it is extremely important to make sure that there is no single bottleneck in the path between the UE and the application server. Nokia AirScale eNB supports 10 Gbps optical backhaul link, which is required by nowadays LTE Advanced networks. Modern UEs also exceeded 1 Gbps barrier. Therefore, 10 Gbps interface should be used by every device transmitting user data, i.e. eNB, access switch, L3 routers, SAE and application server. Standard 1 Gbps Ethernet connection can be used only by test controlling equipment.

Lab setup used for throughput measurement is presented at Figure 2. The UE can operate in one of two modes. The first option assumes that UE operates as mobile modem and provides an additional network interface for PC. In this case, the link during data transmission is terminated in PC, so good quality USB 3.0 cables must be used during the test. The second option, which is preferable, requires Android device which can be controlled over the ADB Shell interface. In this case, the data transfer is terminated at the UE. This solution better reflects real field situation and allows testers to use USB 2.0

interfaces in a shieldbox, which provides isolation between UE and external unwanted RF signals. UE is connected to Radio Module (RM) of eNB via RF cables, what allows us to provide good radio conditions and uncorrelation between MIMO streams. In order to achieve controlled test conditions, it is recommended to install eNB, SAE and application server in the local network with minimum 10 Gbps links, which emulates operator's EPC. Additional 1 Gbps management network is used for remote access, test control, logs collection and automation.

An application server has two network interfaces. The management interface should be configured as default trace. In this case, every traffic, which is not specified in the routing table will go through management network and will not have a negative impact on the mobile core network used for throughput measurement. However, server administrators must provide proper configuration of the routing table. Packets addressed to subnetworks used by UEs should be routed via 10 Gbps interface.

3.2 FTP technique

One of the oldest throughput measurement techniques involves downloading a file from FTP to a host device connected to test the network. In this case, the peak data transmission speed is determined manually by a tester who observes the throughput estimated

by the FTP client application. The average throughput may also be calculated based on the transferred file size and time needed to download/upload it from/to FTP server.

This simple method has a lot of disadvantages. At first, it is required to prepare and store a series of files (each of a different size) with random content. After a test, the transferred file must be removed. Also, the FTP client does not provide detailed information about transfer history. Additionally, the FTP protocol introduces an overhead which has to be included in throughput calculation. This method is also unpractical from the test automation perspective. The test execution time depends on the file size and end-to-end link capacity, which is generally unknown or variable. If we select a file, which is too small, then we may not achieve the maximum throughput. In case of a file, which is too large, the time duration may be unpractically long for slower links. Therefore, the tester should know what throughput is expected and based on this, select a properly sized file.

3.3 iPerf application technique

Described limitations of the FTP technique are overcome by iPerf tool [4], which is well-known by engineers involved in network benchmarking. This open source application is available on all platforms, including Windows, Linux and Android operating systems. Currently the newest iPerf version is 3.5, however a lot of test solutions still use iPerf 2.0.x version, which is incompatible with newer 3.x branch of application.

Iperf tool works in two modes: client and server. During the test, data stream is transmitted from the client to the server. This terminology is confusing at the beginning, because during the downlink throughput test, it is required to run iPerf application in client mode at a host (i.e. application server) which is often named iperf server. However, this principle of operation has a lot of advantages. For an example, test preparation operations may be executed over the management network which does not have impact on the UE state. Therefore, it is possible to use the iPerf tool to test scenarios, which could not be tested using the FTP method. For an example, a UE may be put in idle mode and the first packet sent from the server to the UE will trigger the mobile terminated call procedure.

This method provides high accuracy results with up to 1 second resolution for two transport protocols: UDP and TCP. It means that iPerf tool calculates how many payload bits (see Fig. x) were received during a 1 second time interval. During the test, it is recommended to use several arguments in order to avoid wrong results and simplify potential troubleshooting. The provided syntax is compatible with iPerf 2.1, which is still the most popular version used in tests:

- **c** – client mode – use this at the transmitting side,
- **s** – server mode – use this at the receiving side,
- **i** – interval – use 1 second in order to get the throughput history log,
- **t** – time duration – default value is 10 seconds, which can be expanded to 30 second in order to demonstrate stability of measured solution,
- **B** – bind to the host – use this on a machine with multiple interfaces in order to guarantee that the test uses 10 Gbps interface,
- **p** – port used by the server to listen – the same port must be used by a client,
- **P** – parallel session – this option is used for multiple TCP connection tests, which are needed to fill the available capacity,
- **u** – UDP mode – by default iPerf uses TCP,
- **b** – bandwidth for UDP – this parameter defines bitrate of UDP stream generated by a client, value 10% higher than the expected peak throughput,
- **l** – data length – datagram size for UDP packets.

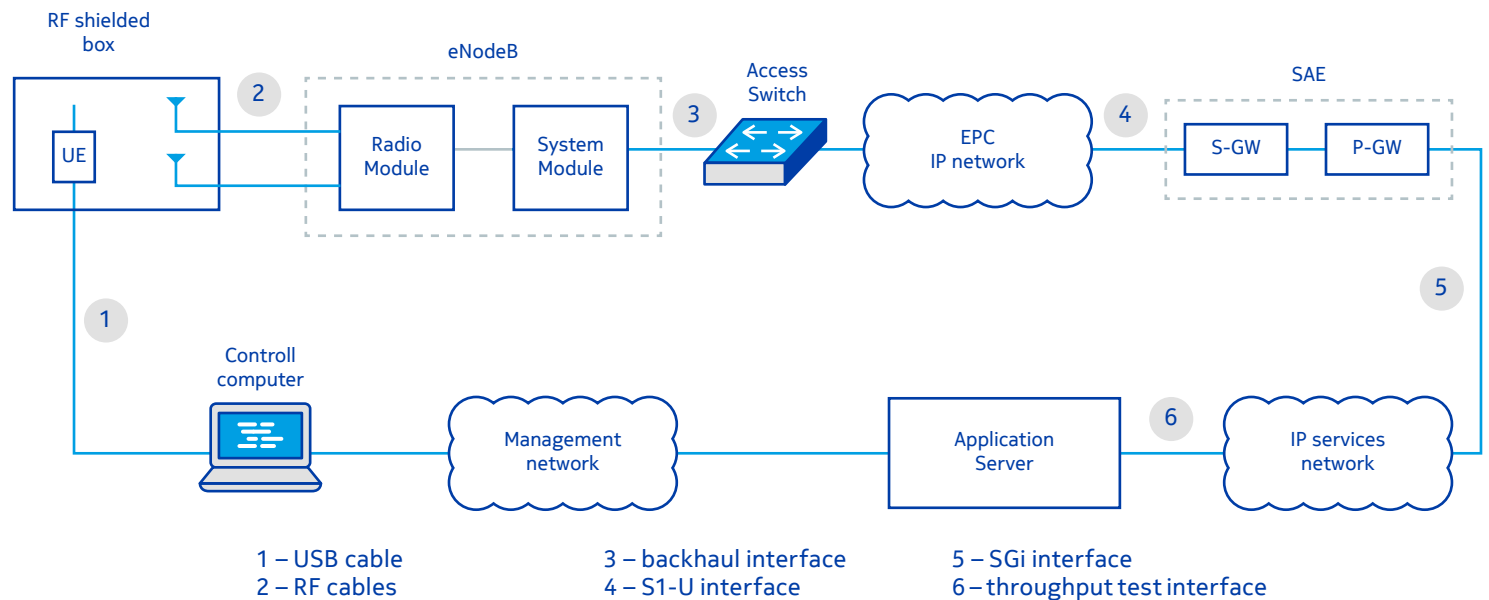
Throughput results are printed by the application run in server mode (i.e. receiving side). The tool provides average throughput and datagrams loss measured in each time interval and for the whole test.

3.4 SpeedTest application technique

SpeedTest applications are available in mobile and web versions. This technique provides good balance between FTP and iPerf methods. The application measure downlink and uplink throughputs initiating connections from the UE side to the application server. In most cases, the traffic is terminated at the server which is located in a different network than the tested eNB. However, from the technical point of view, it is possible to setup a dedicated server in the same lab.

Mobile network benchmarking applications are widely used by end users of commercial networks. These tests are run in uncontrolled conditions, i.e. bad RF conditions, and in the presence of other users connected to the same cell. This methodology makes it difficult to measure the peak throughput, but it is still possible in a lab environment. All mobile applications (e.g. Speed Test by developed by the Polish company V-Speed) measure the HTTP throughput of link between web server and mobile device. During the test, a mobile application establishes a high amount of HTTP connections in order to reach the maximum throughput in a short time. The results are comparable with multiple TCP connection tests done via the iPerf tool and FTP tests for multiple files transfer.

Figure 2 Lab setup dedicated for throughput measurement tests.



4. Summary and challenges for new technologies

All described throughput measurement techniques are commonly used by engineers working with mobile networks. In test automation the most popular are solutions based on the iPerf application, which provides reliable results and may be easily used for nonstandard scenarios. FTP techniques have recently lost popularity because of several limitations. This method is being replaced by SpeedTest type mobile applications, which are commonly used in commercial networks by average users.

Apparently, this test area looks to be well mastered. However, new challenges and technical problems will have to be solved after the introduction of Massive MIMO technology in FDD LTE-Advanced Pro networks. New solutions will require implementation of beam isolation networks in lab environments. Described lab setup will have to be expanded into multiple UEs setup with a centralised controlling system. The main objective of Massive MIMO is not to improve in the peak throughput, but to increase the cell capacity. This means that the test purpose will be measure the aggregated traffic generated several active UEs connected to the same cell.

References

[1] H. Holma, A. Toskala, *LTE for UMTS: Evolution to LTE-Advanced*, 2nd ed., John Wiley & Sons, 2011
[2] Constantine, B., Forget, G., Geib, R., R. Schrage, *Framework for TCP Throughput Testing*, RFC 6349, August 2011, <http://www.rfc-editor.org/info/rfc6349>.
[3] Anu A. Gokhale, *Introduction to Telecommunications*, 2nd ed., Thomson Delmar Learning, 2004
[4] <https://iperf.fr/>

About the author

I am a graduate of the Electronics and Telecommunications at Wroclaw University of Technology. Since 2014 I have been working Nokia’s test department responsible for verification of 4G LTE eNodeB product. My professional activities was focused on the border of Radio Resource Management and Transport areas.

Mariusz Zamłyński
Verification Architect
MN 4G RAN

Primality Tests

Łukasz Grządko
Senior Engineer, Software Development
MN LTE



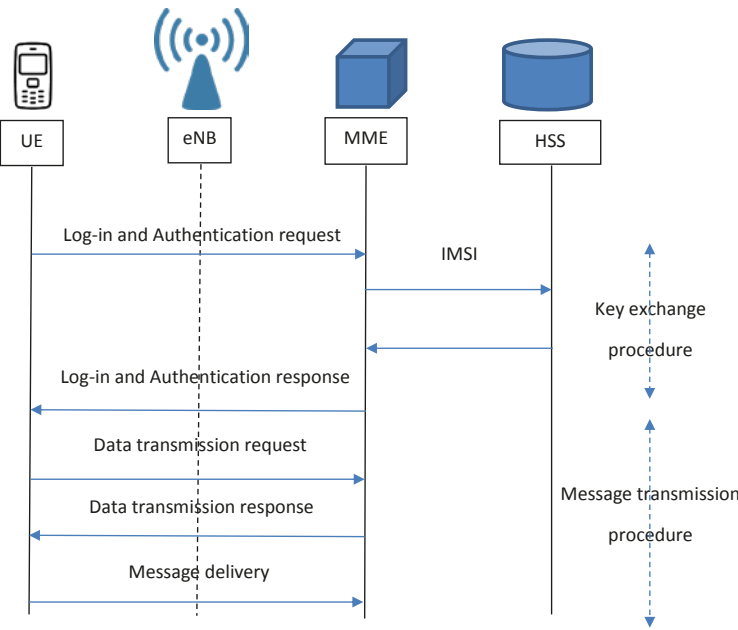
Mobile wireless communication is one of the most common topics in networking. Users can access Internet services anytime and anywhere by using mobile communication. But while this access brings many advantages, it is important to consider the challenges; one of which is data transmission security. An insecure network will endanger network applications as hackers may steal important information, such as credit card numbers, bank transfers, passwords, love letters, instructions for buying stocks, and secret diplomatic information. Therefore, wireless network security protection is essential and it has become desirable to develop a safe method of coding messages. In the past, code had been kept secret and had been known only to the parties sending and receiving messages, but it had been possible to study intercepted messages and crack the code. Enciphering is also applied in war situations. On 8 May 1919, Lt. Stanisłicki established a Polish Army “Cipher Section” which was a precursor to the “Cipher Bureau”. The Cipher Section reported to the Polish General Staff and contributed to Poland’s defense by Piłsudski’s forces during the Polish-Soviet War of 1919–21, thereby helping to preserve Poland’s independence, (major battle in the world history known as “Miracle on the Vistula” prevented the whole Europe from Soviet communism [7]), recently regained in the wake of World War I. During the Polish-Soviet War, hundreds of Russian ciphers were broken by great Polish mathematicians: Kowalewski, Mazurkiewicz, Sierpiński, and Leśniewski [4]. At that time, Germans invented a cipher machine called Enigma and used it during World War II. However, thanks to Polish cryptologists: Ciężki, Rejewski, Zygański, and Różycki [1, 3], who cracked Enigma, World War II finished earlier and took less victims. Rejewski invented the theory of permutations and groups which was a key to deciphering Enigma [2]. Great progress in cryptography came with the advent of public key crypto-systems. The main characteristics of such a system are simplicity, public key, and extreme difficulty to crack. The idea was proposed in 1976 by Diffie and Hellman and the effective implementation was proposed in 1978 by Rivest, Shamir, and Adleman (the authors of the RSA cryptosystem) [32]. The exemplary security system for the 4G environment (called Se4GE), which is an LTE-based system, integrates RSA and Diffie-Hellman methods. They solve some of the security drawbacks of current 4G broadband wireless transmission [22]. The RSA system is a part of key exchange procedure (Figure 1) and works as follows:

$$C \equiv M^e \pmod n$$
$$M \equiv C^d \pmod n$$

- numbers M and C are plaintext and ciphertext respectively,
- $n = pq$ is the modulus with p and q large and distinct primes (randomly chosen and differ in length by a few digits),
- e is the public encryption exponent (key),
- d is the private decryption exponent (key) with $ed \equiv 1 \pmod{\varphi(n)}$.

Efficient generation of public-key parameters is necessary. We need a tool for efficient verification of whether randomly chosen big numbers p and q are primes [36].

Figure 1 The Sequence Chart of the Se4GE Key Exchange and Data Transmission Procedures [22]



1. Primality Tests

A prime number is a natural number $p \geq 2$ which has exactly two distinct natural divisors, i.e. 1 and itself [5]. Note that 1 is not a prime number and not a composite number. An interesting fact is that 1 is one of highly composite numbers (or anti-primes) [12]. Distinguishing primes from composites is one of the most fundamental problems in algorithmic number theory – this is known as the PRIMES problem [47]. For a long time, scientists did not know that testing for primality costs polynomial time. Before this discovery, it was only thought that $\text{PRIMES} \in \text{NP} \cap \text{coNP}$. Therefore, multiple randomized algorithms, also known as Monte-Carlo and Las-Vegas methods, have been discovered to solve the problem with primality testing.

2. Deterministic primality tests

Deterministic algorithms follow the same execution path (sequence of operations) each time they are executed with the same input [26].

2.1. Brute-force primality tests

The brute-force approach is a simple example of a deterministic algorithm which is always correct, but may cost exponential time of computation.

2.1.1. Test all divisors

This is a straightforward translation of primality definition into an algorithm. For each integer number $1 < a < n$ we check if n is divisible by a .

Figure 2 Test All Divisors Algorithm

```
Test all divisors(n)
  if n = 1 return "n is not prime"
  for a from 2 to n - 1
    if n mod a = 0 then return "n is composite"
  return "n is prime"
```

Algorithm from Figure 2 has time complexity $O(n)$ which is exponential on the number of digits in the input. This algorithm can be sped up to complexity $O(\sqrt{n})$ because for each integer a , if a is a divisor of n , then $\frac{n}{a}$ is a divisor of n .

Figure 3 Enhanced Algorithm from Figure 2

```
Test all divisors(n)
  if n = 1 return "n is not prime"
  for a ← 2; a * a ≤ n; a ← a + 1
    if n mod a = 0 then return "n is composite"
  return "n is prime"
```

2.1.2. Sieve of Erathostenes test

The Sieve of Erathostenes is an ancient algorithm, which instead of testing one number, finds all prime numbers up to any given limit [6, 13]. Time complexity $O(n \log \log n)$ is nearly the same as of the algorithm from Figure 2. The disadvantage of this algorithm is space complexity, i.e. $O(n)$. Let $n = 13$ and imagine a table with 12 cells each with an integer from 2 to 13. The first number, i.e. 2, is prime and all its multiples are sieved out. The next not sieved out value is the next prime number, i.e. 3. Like in the previous step, all multiples of 3 are sieved out. It is enough to start sieving out from p^2 each time when a new prime number p is revealed.

Figure 4 Example of Sieve for integers up to 13

2	3	4	5	6	7	8	9	10	11	12	13
---	---	---	---	---	---	---	---	----	----	----	----

2.2. Fast deterministic primality tests for special numbers

There exist several interesting optimal algorithms which only work for specific numbers. They give answers faster than brute-force algorithms. Primality tests for Fermat and Mersenne numbers will be shown.

2.2.1. Pepin primality test

If p is a Fermat number, i.e. $p = 2^{2^k} + 1$, then p is prime if and only if $3^{(p-1)/2} \equiv -1 \pmod{p}$. It can be done in $O(\log(p))$ time instead of $O(p)$.

2.2.2. Lucas-Lehmer primality test for Mersenne primes M_n

A Mersenne number M_n is a number in the form of $2^n - 1$. There is a very efficient and deterministic Lucas test (compare with chapter 4.4.) specifically for Mersenne primes, known as Lucas-Lehmer test. Let $a = -2$ and $b = 2$. Consider the associated Lucas sequences U_k, V_k with discriminant $D = 12$. Then $N = M_n$ is prime if and only if $N | V_{(N+1)/2}$. To check if $N = 2^7 - 1$ is prime, we need to compute V_{64} , which is 28 digits long. For purpose of computation, it is convenient to replace the Lucas sequence with the following Lucas-Lehmer sequence: $L_0 = 4, L_{k+1} = L_k^2 - 2$. Lucas-Lehmer theorem says: let n be an odd prime. Then $2^n - 1$ is prime if and only if M_n divides L_{n-2} [36].

3. Randomized primality tests

Randomized algorithms make random decisions at certain points in their execution. Unfortunately, they cannot guarantee the same result for each run on the same input like deterministic algorithms, but many problems can be solved more efficiently in terms of both time and space.

3.1. Fermat primality test

The set of all congruence classes of the integers for a modulus n is called the ring of integers modulo n , and is denoted $\mathbb{Z}/n\mathbb{Z} = \{\bar{a}_n \mid a \in \mathbb{Z}\} = \{\bar{0}_n, \bar{1}_n, \dots, \overline{n-1}_n\}$. This set is a group under multiplication modulo n and is denoted $(\mathbb{Z}/n\mathbb{Z})^*$. The order of the group is $\varphi(n)$, i.e. the number of all a such that $\gcd(a, n) = 1$. Euler's Theorem states that: if n and a are coprime, then $a^{\varphi(n)} \equiv 1 \pmod{n}$ [5, 6, 8, 14, 24, 27]. According to Fermat's Little Theorem, if p is a prime number and a is a positive integer, then $a^p \equiv a \pmod{p}$ for all integers a . In addition, if $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$. Note that this is also a special case of Euler's Theorem. Suppose p is the number which we want to test for primality. Two steps are applied:

- 1) Choose randomly number $1 < a < p - 1$
- 2) Calculate $a^{p-1} \bmod p$

After the 2nd step, the result is checked against Fermat's Little Theorem. If the result is not 1, then p cannot be prime. Otherwise it is not guaranteed that p is prime. In such situation, algorithm steps are repeated. We stop after several iterations. More iterations should result in higher accuracy. If the result is still 1, then number p is probably prime. However, if p is composite then p is called a pseudoprime to base a . The composite integer $n = 341 = 11 \times 31$ is a pseudoprime to base 2 since $2^{340} \equiv 1 \pmod{341}$. The Fermat base-2 pseudoprimal-ity test is also called Chinese test [36]. Number of arithmetic operations needed for the whole procedure (Figure 5) is $O(k \log n)$. Please note that if the cost of multiplication of two numbers is taken into account, then overall time complexity increases to $O(k \log^3 n)$.

Figure 5 Fermat Primality Test [21]

```
Fermat(n,k)
  for i from 1 to k
    choose a random integer a, 2 ≤ a ≤ n - 2
    if gcd(a,n) ≠ 1 return "n is composite"
    compute r ← a^{(n-1)} mod n
    if r ≠ 1 return "n is composite"
  return "n is probably prime"
```

3.1.1. Carmichael numbers

The difficulty with using the Fermat test is that there is a group of composite numbers named Carmichael numbers, which are pseudo-primes to every base, relatively prime to n . Examples of such numbers: 561, 1105, 1729. Whenever we are not dealing with Carmichael

numbers, the probability of Fermat's test failure is $\frac{1}{2^k}$ at most. There

are 246683 Carmichael numbers up to 10^{16} and 1401644 up to 10^{18} [17, 36]. An example of an algorithm for constructing large Carmichael numbers can be found in [25].

3.1.2. Implementation issues

3.1.2.1. Fast modular exponentiation ($a^b \bmod c$)

If number p is quite big, i.e. $\sim 10^9$, then a brute-force linear time algorithm is too slow. The following recursive formula allows us to compute exponentiation much faster, i.e. in $O(\log(b))$:

$$a^b \bmod c = \begin{cases} (a^2)^{b/2}, & b \text{ is even} \\ a * a^{b-1} = a * (a^2)^{\frac{b-1}{2}}, & b \text{ is odd} \\ 1, & b = 0 \end{cases}$$

3.1.2.2. Multiplying 64-bit numbers

The following recursive formula can be used in case of multiplication overflow, however only a division remainder less than 2^{63} is required:

$$a * b \bmod c = \begin{cases} (2a) * \left(\frac{b}{2}\right), & b \text{ is even} \\ a + (2a) * \frac{b-1}{2}, & b \text{ is odd} \\ 0, & b = 0 \end{cases}$$

Another approach of multiplication of 64-bit numbers modulo some Mersenne prime number (as a part of solution for a string matching problem on a distributed system) can be found in [50]. GCC compiler also supports built-in `__int128` type for 128-bit integers.

3.1.2.3. Multiplying larger numbers

When dealing with numbers wider than machine words, we need to take into account even the complexity of multiplication. For n digit numbers it costs $O(n^2)$ operations. Although many languages support big numbers, it is worth implementing particular arithmetic operations in a more efficient way.

Karatsuba algorithm

Knowing the binary representations of A and B , a divide-and-conquer approach can be applied to multiply the numbers. Each representation of A and B is divided into two halves A_1, A_2 and B_1, B_2

of $n/2$ binary digits each, i.e. $A = (a_n \dots a_{\frac{n}{2}+1}) * 2^{\frac{n}{2}} + (a_{n/2} \dots a_1)$. The

product of A and B is $A_1 B_1 2^n + (A_1 B_2 + A_2 B_1) 2^{\frac{n}{2}} + A_2 B_2$. There are 4 multiplications of $n/2$ digits numbers, therefore time complexity is $T(n) = 4T(n/2) + cn = O(n^2)$. Karatsuba's idea is to make 3 such multiplications using the following formula: $(A_1 - A_2) * (B_2 - B_1) + A_1 B_1 + A_2 B_2 = A_1 B_2 + A_2 B_1$. Therefore, $T(n) = 3T(n/2) + cn = O(n^{\log_2 3}) = O(n^{1.58})$, which is better than a brute-force solution.

Fast Fourier transform (FFT)

Multiplying huge integers can be done in time $O(n \log n)$ using FFT. The main idea is to represent numbers as polynomials. FFT directly multiplies polynomial $p(x)$ by $q(x)$ in time $O(n \log n)$ where $p(x) = \sum_{i=0}^{n-1} a_i x^i, q(x) = \sum_{i=0}^{n-1} b_i x^i$. More details in [11, 16, 41]. Now, if we can multiply polynomials fast, we can use them to multiply integers. Decimal numbers have base 10, therefore $x = 10$. The digits are polynomial coefficients i.e. a_i, b_i . Note, that if we use larger bases like 100, 1000, ..., multiplying becomes faster.

3.1.2.4. Greatest common divisor, calculation $gcd(a,b)$

Trying all $d \leq \min(a,b)$ would be a brute-force approach which would not be sufficient for large numbers. Simple observation that $gcd(a,b) = gcd(b, a \bmod b)$ can be used to challenge the problem. It is also known as the Euclidean algorithm, which runs in $O(\log(\max\{a, b\}))$. If number of divisions of $\log(n)$ -bit numbers is taken into account, then the time complexity becomes $O(\log^3 n)$ [16].

3.2. Solovay-Strassen – Monte Carlo algorithm based – primality test

The Solovay-Strassen probabilistic test [31] was the first test popularized by the advent of public-key cryptography, in particular the RSA cryptosystem [26]. It is also an example of the Monte-Carlo method. Now we need to introduce some definitions and facts before the algorithm will be presented. The Legendre

symbol $\left(\frac{a}{p}\right)$ is a tool for keeping track of whether or not an integer a is

a quadratic residue modulo a prime p . The Jacobi symbol is a generalization of the Legendre symbol to integers n , which are odd, but not necessarily prime. Let $n \geq 3$ be odd with prime factorization

$n = p_1^{\alpha_1} p_2^{\alpha_2} p_3^{\alpha_3} \dots p_s^{\alpha_s}$. Then the Jacobi symbol $\left(\frac{a}{n}\right)$ is defined to be $\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha_1} \left(\frac{a}{p_2}\right)^{\alpha_2} \dots \left(\frac{a}{p_s}\right)^{\alpha_s}$. Observe that if n is prime, then the Jacobi

symbol is the Legendre symbol. Let's now look at some properties of the Jacobi symbol. Let $m, n \geq 3$ be odd integers, $a, b \in \mathbb{Z}$, then the following is true:

- 1) $\left(\frac{m_1 m_2}{n}\right) = \left(\frac{m_1}{n}\right) \left(\frac{m_2}{n}\right)$,
- 2) $\left(\frac{m}{n}\right) = \left(\frac{m \bmod n}{n}\right)$,
- 3) if $n \equiv m \equiv 3 \pmod{4}$ then $\left(\frac{m}{n}\right) = -\left(\frac{n}{m}\right)$, otherwise $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right)$,
- 4) $\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}$, $\left(\frac{1}{n}\right) = 1$, $\left(\frac{0}{n}\right) = 0$.

By properties of the Jacobi symbol, it can be concluded: if n is odd and $a = 2^\alpha r$ where r is odd, then $\left(\frac{a}{n}\right) = \left(\frac{2^\alpha}{n}\right) \left(\frac{r}{n}\right)$.

Figure 6 Jacobi Symbol Computation

JacobiSymbol(a, n)
If $a > n$ return *JacobiSymbol(a mod n, n)*
If $(a = 0 \text{ or } a = 1)$ return a
Find α and odd r such that $a = 2^\alpha r$; if $\alpha > 0$ then return $\left(\frac{2}{n}\right)^{\alpha \bmod 2} * \text{JacobiSymbol}(r, n)$
if $a \equiv n \equiv 3 \pmod{4}$ return $-\text{JacobiSymbol}(n, a)$ else return *JacobiSymbol(n, a)*

For example:

$$\left(\frac{1411}{317}\right) = \left(\frac{1411 \bmod 317}{317}\right) = \left(\frac{143}{317}\right) = \dots = \left(\frac{12}{19}\right) = \left(\frac{2}{19}\right)^2 \left(\frac{3}{19}\right) = -1$$

Running time complexity of Jacobi-Symbol computation is $O((\log n)^2)$ bit operations.

Now let's take a look at the main algorithm. The test is based on

Euler's criterion. Let n be an odd prime. Then $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \pmod{n}$

for all integers a satisfying $gcd(a, n) = 1$. The a which doesn't satisfy this formula is called an Euler witness to compositeness for n . Odd composite n which satisfies the formula for an a is an Euler pseudoprime to base a .

Figure 7 Solovay-Strassen Probabilistic Primality Test

Solovay-Strassen(n, k)
for i from 1 to k
choose a random integer a , $2 \leq a \leq n - 2$
if $gcd(a, n) > 1$ then return "n is composite"
compute $r \leftarrow a^{(n-1)/2} \bmod n$
if $r \neq 1$ and $r \neq n - 1$ return "n is composite"
compute the Jacobi symbol $s \leftarrow \left(\frac{a}{n}\right)$
if $r \not\equiv s \pmod{n}$ return "n is composite"
return "n is probably prime"

k -fold repetition using independent random numbers yields Monte-Carlo test with error probabilities 0 if n is prime, and less than

$\left(\frac{1}{2}\right)^k$ if n is composite. The overall time complexity is $O(k \log^3 n)$.

3.3. Rabin-Miller primality test

It is also known as the strong pseudoprime test and is an another example of the Monte-Carlo method. The test is based on the following fact:

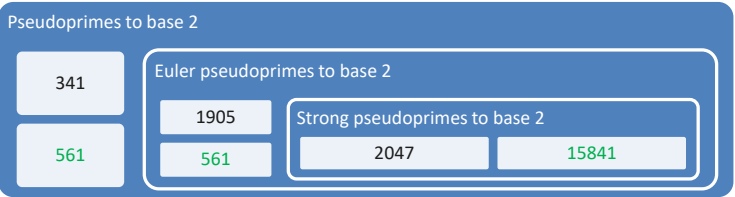
Let n be an odd prime number and $n - 1 = 2^s r$, r is odd. Let a be a number such that $gcd(a, n) = 1$. Then either $a^r \equiv 1 \pmod{n}$ or $a^{2^j r} \equiv -1 \pmod{n}$ for a j , $0 \leq j \leq s - 1$. This fact can be proved with Fermat's Little Theorem and the following lemma: if p is prime, then $x^2 \equiv 1 \pmod{p}$ if and only if $x \equiv 1$ or $x \equiv -1 \pmod{p}$. Thanks to this fact, we may strengthen Fermat's test. Recall Carmichael number 561 as a counter-example for Fermat's test. Let $x = 5$. Then $5^{560} = (5^{280})^2 \equiv 1 \pmod{561}$, but $5^{280} \equiv 67 \pmod{561}$, hence 561 is composite. These facts can be used to check the compositeness of a number. Let $1 < a < n$

- 1) If $a^r \not\equiv 1 \pmod{n}$ and for each $0 \leq j < s$, $a^{2^j r} \not\equiv -1 \pmod{n}$, then a is called a strong witness for the compositeness of n .
- 2) Otherwise n is probably prime. If n is composite, then it is a strong pseudoprime to base a .

Figure 8 Rabin Miller Primality Test [26]

Rabin-Miller(n, k)
Find s and odd r such that $n - 1 = 2^s r$
for i from 1 to k
Let a be randomly chosen such that $1 < a < n - 1$
Calculate $y = a^r \bmod n$
if $y \neq 1$ and $y \neq -1$ then
j ← 1
while $j < s$ and $y \neq -1$
y ← $y^2 \bmod n$
if $y = 1$ return "n is composite"
if $y = -1$ break;
j ← j + 1
if $y \neq -1$ return "n is composite"
return "n is probably prime"

Figure 9 Least Pseudoprimes found by Fermat Test (and Chinese Test), Solovay-Strassen Test, Miller Rabin Test. The Least Carmichael Pseudoprime Numbers found are in Green [42]



For any odd composite number n the probability that Miller-Rabin

test returns "n is probably prime" is less than $\left(\frac{1}{4}\right)^k$. The overall time

complexity is $O(k \log^3 n)$. Using FFT, the running time can be reduced to $\tilde{O}(k \log^2 n)$, where $\tilde{O}(f(n))$ denotes the class $\mathcal{O}(f(n) \log^l f(n))$ [34, 43].

4. Heuristic primality tests

Heuristic test is a test that seems to work well in practice, but is unproven [29].

4.1. Deterministic variant of Miller-Rabin test and Extended Riemann Hypothesis (ERH)

This test is based on deterministic searching for a value a from the set of least consecutive primes. The value a is a witness to the compositeness of n . Work of Ankeny shows that ERH implies a bound of size of the so called "nonresidues" to be $O(\log^2 n)$ [15] which implies a polynomial time algorithm to recognize primes with running time $\tilde{O}(\log^4 n)$ [26, 34]. For example, the following set of values {2, 3, 5, 7} applied to the deterministic Miller-Rabin procedure is enough to check primality for all numbers $n \leq 2 * 10^9$ [14].

4.2. Chinese primality test

This is a deterministic variant of Fermat Primality test with base 2 [36]. There is only one check: if $2^p \equiv 2 \pmod{p}$, then p is probably prime.

4.3. Fibonacci primality test

The sequence F_n of Fibonacci numbers is defined by the recurrence relation: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$. The first elements are: 0, 1, 1, 2, 3, 5, 8, 13, 21 ... It can be proven that if n is prime, then $F_{n - \left(\frac{n}{5}\right)} \equiv 0 \pmod{n}$.

For $n = 341$, which is a counter example for the Fermat base-2 test, the Fibonacci test returns "composite" [30].

Implementation issues of F_n
Straightforward calculation of F_n is very slow. There is an efficient

algorithm which uses the following fact: $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix}$.

By considering the matrix entries \pmod{n} , the matrix exponentiation can be done in $O(\log n)$ time [10].

$\left(\frac{n}{5}\right)$ denotes the Legendre symbol that is:

$$\left(\frac{n}{5}\right) = \begin{cases} 1, & \text{if } n \equiv \pm 1 \pmod{5} \\ -1, & \text{if } n \equiv \pm 2 \pmod{5} \\ 0, & \text{if } n \equiv 0 \pmod{5} \end{cases}$$

4.4. Lucas primality test

Let a, b be non-zero integers and $D = a^2 - 4b$. Consider the equation $x^2 - ax + b = 0$. Its discriminant is D and α and β are two roots:

$$\alpha = \frac{a + \sqrt{D}}{2}, \beta = \frac{a - \sqrt{D}}{2}$$
. Sequences $U_k = \frac{\alpha^k - \beta^k}{\alpha - \beta}$ and $V_k = \alpha^k + \beta^k$ are called

Lucas sequences. They can be also defined in a recursive form: $U_k(a,b)=aU_{k-1}-bU_{k-2}$, $V_k(a,b)=aV_{k-1}-bV_{k-2}$. For example, the sequence $U_k(a,b)$ where $a=1$ and $b=-1$ form Fibonacci sequence. As before, we need a condition to test if a number is composite or probably prime. It is known as the Lucas’ Theorem:

If p is an odd prime, $p \nmid b$ and $\left(\frac{D}{p}\right)=-1$, then $p|U_{p+1}$. Equivalent

presentation is as follows: Let n be an odd positive integer and

$\delta(n)=n-\left(\frac{D}{n}\right)$. If n is prime and $gcd(n,b)=1$, then $U_{\delta(n)}\equiv 0 \pmod n$.

If n is composite but congruence still holds, then n is called a Lucas pseudoprime with parameters a and b .

4.4.1. Strong Lucas primality test

$\delta(n)$ is factored to the form $d*2^s$, where d is odd. A strong Lucas pseudoprime for a given (a,b) pair is an odd composite number n with $gcd(n,D)=1$ satisfying one of the conditions $U_d \equiv 0 \pmod n$ or $V_{d2^r} \equiv 0 \pmod n$ where $0 \leq r < s$ [19]. The fraction of a/b pairs modulo n that declare a composite n to be probably prime is at most $(4/15)$.

Therefore, k applications of the test yield an error of at most $\left(\frac{4}{15}\right)^k$ [19].

Implementation issues of U_m

Please note that the roots are not necessarily integers, not even rational. However, the output is an integer. Straightforward computation using recurrence formula gives a linear algorithm which is insufficient for a large m . To compute U_m in $\log(m)$ time one can use formulas [40]: $U_{2k}=U_kV_k$, $V_{2k}=V_k^2-2b^k$ and $U_{2k+1}=(aU_{2k}+V_{2k})/2$, and $V_{2k+1}=(DU_{2k}+aV_{2k})/2$. U_m can be also obtained from the binary expansion of m [19]. Suppose $m=109=1101101b$. A sequence of required indices of U_i : 108, 54, 27, 26, 13, 12, 6, 3, 2, 1.

Implementation issues of $\left(\frac{D}{n}\right)$

Two methods have been proposed to choose parameter D :

1) Let D be the first element of the sequence 5, -7 , 9, -11 , 13, ...

for which $\left(\frac{D}{n}\right)=-1$. Let $a=1$, $b=(1-D)/4$.

2) Let D be the first element of the arithmetic sequence 5, 9, 13, ...

for which $\left(\frac{D}{n}\right)=-1$. Let a be the least odd number exceeding \sqrt{D} and

$$b=\frac{a^2-D}{4}.$$

There are more Lucas pseudoprimes for the second method [48]. The first method will be later used in Baillie-PSW primality test.

Does a required D always exist? Yes, except D is a perfect square. The average number of steps to find D is about 3.147755149 [48]. We also need an efficient algorithm to check if a number is a perfect square. It can be done, for example by using Newton methods or binary searching for each digit of the potential perfect square [18].

4.5. The power of combining primality tests

Combining primality tests proves to be extremely powerful. For example, there are 29334 Fibonacci pseudoprimes less than 10^{11} and 38975 pseudoprimes to base 2. When combining the Fermat test and the Fibonacci test, there are only 2517 pseudoprimes [30]. Moreover, Selfridge has conjectured that if p is an odd number, and $p \equiv \pm 2 \pmod 5$, then p will be prime if both of the following hold: $2^{p-1} \equiv 1 \pmod p$, $F_{p+1} \equiv 0 \pmod p$ [29].

Another example of a combination of primality tests is the BPSW (Baillie, Pomerance, Selfridge, Wagstaff) test in [Figure 10](#), which is an excellent heuristic. There is a conjecture that if n is a positive integer greater than 1 which can pass the combination of a strong pseudoprimality test and a strong Lucas test, then n is prime [36]. It has been shown to be deterministically correct for all inputs less than $25*10^9$ [39], even for all 64-bit inputs [29]. A similar conjecture, where a strong Lucas test is replaced by a Lucas test, also exists [36].

4.5.1. Baillie-PSW – Lucas test based – primality test

Figure 10 Baillie – PSW Primality test [19, 39]

<i>Baillie-PSW(n)</i> <i>Run Miller-Rabin test to check strong pseudoprimality to base 2 on n. If this test fails, declare n as composite and halt. Otherwise n is probably prime and go into next step.</i> <i>In the sequence 5, -7, 9, -11, 13 ... find the first number D for which $\left(\frac{D}{n}\right)=-1$.</i> <i>Run a strong Lucas pseudoprime test with discriminant D on n. If this test fails, declare n composite. If this test succeeds, n is “very probably” prime.</i>

5. Advanced primality tests

More complex algorithms are based on the Pocklington Theorem and a generalization of the Fermat Theorem.

5.1. Pocklington-Lehmer $N-1$ primality test

The test uses a partial factorization of $N-1$ to prove that an integer N is prime. The test relies on the Pocklington theorem: Let $N>1$ be an integer and suppose there exist numbers a and q such that q is prime, $q|N-1$ and $q>\sqrt{N}-1$, $a^{N-1} \equiv 1 \pmod N$,

$gcd\left(a^{\frac{N-1}{q}}-1,N\right)=1$. Then N is prime [35].

5.2. Elliptic curve – Las Vegas algorithm based – primality test

Elliptic curve primality tests are based on criteria similar to the Pocklington theorem [36], where the group $(\mathbb{Z}/n\mathbb{Z})^*$ is replaced by $E(\mathbb{Z}/n\mathbb{Z})$ and E is a properly chosen elliptic curve [20]. The elliptic curve over $\mathbb{Z}/n\mathbb{Z}$ is given by: $y^2=x^3+ax+b \pmod n$. The test is a Las Vegas algorithm. It means the answer is always correct, however the running time is random. The decision algorithm which checks whether n is a prime number, is known as the Goldwasser-Kilian-Atkin test. The implementation of the algorithm can be found in [38].

5.3. PRIMES $\in P$, AKS primality test

An unconditional deterministic polynomial-time algorithm exists that determines whether an input number is prime or composite [23]. It also proves that $\text{PRIMES} \in P$. The test is based on a generalization of Fermat’s Little Theorem. The asymptotic time complexity

of the algorithm is $O(\log^{\frac{15}{2}}n)$.

5.4. Quantum primality test

In [28] there is considered a probabilistic quantum Shor’s algorithm [9, 33] of a variant of the Pocklington-Lehmer $N-1$ primality test. $O(\log^3N \log \log N \log \log \log N)$ q -bit operations are required to determine the primality of N , making it the fastest known primality test. If a quantum mechanical computer is ever built, the RSA crypto-system will no longer be secure, because there are polynomial time algorithms for prime factorization. Some experts have suggested that quantum cryptography will be the only way to ensure the security of a cryptosystem.

Table 1 Time Complexity of Primality Tests with Error Probabilities

Test name	Time complexity	Kind of test	Probability of error
Test all divisors	$O(n)$	Brute-force/ exponential time	0.0
Enhanced test all divisors	$O(\sqrt{n})$	Brute-force/ exponential time	0.0
Sieve of Erathostenes	$O(n \log \log n)$	Brute-force/ exponential time	0.0
AKS test	$O(\log^{\frac{15}{2}}n)$	Deterministic/ polynomial time	0.0
Lenstra and Pomerance modification	$\tilde{O}(\log^6n)$		
Fermat test	$O(k \log^3n)$	Randomized	0.5^k
Solovay-Strassen test	$O(k \log^3n)$	Randomized	0.5^k
Miller-Rabin test with FFT multiplication heuristic variant	$O(k \log^3n)$	Randomized	0.25^k
	$\tilde{O}(k \log^2n)$ $\tilde{O}(\log^4n)$	Heuristic	
Strong Lucas test	$O(k \log^3n)$	Randomized	$\left(\frac{4}{15}\right)^k$
Baillie-PSW test	$O(\log^3n)$	Heuristic	probably 0.0
Elliptic curve test	random time	Randomized	0.0
Quantum test	$O(\log^3n \log \log n \log \log \log n)$	Randomized	0.0

6. Conclusions

We have started with a great and successful application of Rejewski’s Theory of Permutations in breaking Enigma. Decades later, new cryptography methods involving prime numbers were discovered, for example RSA. It accelerated searching for efficient algorithms to check primality and factorization big numbers. Some probabilistic ideas were discovered, as in that time it was not proved that Primes is in P . These methods are very fast in practice and are proven for larger numbers, including 32- or 64-bit numbers. There is even an another algorithm which discovers 32-bit primes in just one round! Forišek and Jančina described this idea in [43] using a single computation of a simple hash function and a single round of the Miller-Rabin test. For smaller numbers, the sieves can be used and can be more efficient in some circumstances. We have faced here $O(n)$ space complexity, which can be improved with a segmented sieve which requires only $O(\sqrt{n})$ space [37]. The other example is the AKS algorithm which shows not only that Primes is in P , but, because of its big time complexity, gives a large field for further research. Lenstra and Pomerance used Gaussian periods and have come up with a modified version of this algorithm and achieved time complexity $\tilde{O}(\log^6 n)$ [44, 45, 46]. Certainly, there is still a large field for research and optimization of primality test algorithms and for searching for even totally new approaches. Algorithms described in this article have various purposes, e. g. cryptography, number theory, string matching. Example of implementation of fast primality test proposed in [43] (with modification of Miller-Rabin test [14] to decrease number of multiplications) has been used for fast number of divisors computation [49].

References

[1] “How Polish Mathematicians Deciphered the Enigma”, Marian Rejewski
[2] “An Application of the Theory of Permutations in Breaking the Enigma Cipher”, Marian Rejewski
[3] <http://lamaczeszyfrow.pl/index.php?id=n>, where $n \in \{115, \dots, 119\}$
[4] [https://pl\[en\].wikipedia.org/wiki/Biuro_Szyfrów](https://pl[en].wikipedia.org/wiki/Biuro_Szyfrów)
[5] „Teoria Liczb, tom 1 i 2”, Wacław Sierpiński
[6] „Arytmetyka teoretyczna”, Wacław Sierpiński
[7] <https://poland.pl/tourism/traditions-and-holidays/assumption-virgin-mary-and-day-polish-armed-forces/>
[8] „Zadania z Olimpiad Matematycznych, tom 2”, Stefan Straszewicz
[9] „Algorytm faktoryzacji Shora”, In periodical: Delta, Wojciech Czerwiński
[10] „W poszukiwaniu wyzwań. Wybór zadań z konkursów programistycznych Uniwersytetu Warszawskiego. [Looking for a Challenge? The Ultimate Problem Set from the University of Warsaw Programming Competitions]”, red. Krzysztof Diks, Tomasz Idziaszek, Jakub Łącki, Jakub Radoszewski

[11] „W poszukiwaniu wyzwań 2: zadania z Akademickich Mistrzostw Polski w Programowaniu Zespołowym 2011–2014”, red. Szymon Acedański, Krzysztof Diks, Tomasz Idziaszek, Jakub Łącki, Jakub Radoszewski
[12] „Przygody Bajtazara. 25 lat Olimpiady Informatycznej – wybór zadań”, red. Krzysztof Diks, Tomasz Idziaszek, Jakub Łącki, Jakub Radoszewski
[13] „Zaprzyjaźnij się z algorytmami. Przewodnik dla początkujących i średniozaawansowanych”, Jacek Tomaszewicz
[14] „Algorytmika praktyczna. Nie tylko dla mistrzów”, Piotr Stańczyk
[15] “The Least Quadratic Non Residue”, Nesmith Cornett Ankeny
[16] “Introduction to Algorithms”, Thomas H. Cormen, Charles Eric Leiserson, Ronald Linn Rivest, Clifford Stein
[17] “The Carmichael Numbers up to 10^{18} ”, Richard G. E. Pinch
[18] <https://discuss.codechef.com/questions/28947/perfect-square>
[19] https://en.wikipedia.org/wiki/Lucas_pseudoprime
[20] https://en.wikipedia.org/wiki/Elliptic_curve_primality
[21] <https://www.topcoder.com/community/data-science/data-science-tutorials/primality-testing-non-deterministic-algorithms/>
[22] “A Secure Wireless Communication System by Integrating RSA and Diffie-Hellman PKDS in 4G Environments and an Intelligent Protection-key Chain with a Data Connection Core”, Yi-Li Huang, Fang-Yie Leu, Yao-Kuo Sun, Cheng-Chung Chu, Chao-Tung Yang
[23] “Primes is in P”, Manindra Agrawal, Neeraj Kayal, Nitin Saxena
[24] “Lecture Notes on the Complexity of Some Problems in Number Theory”, Dana Angluin
[25] “A New Algorithm for Constructing Large Carmichael Numbers”, Günter Löh, Wolfgang Niebuhr
[26] “Handbook of Applied Cryptography”, Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone
[27] https://en.wikipedia.org/wiki/Modular_arithmetic
[28] “Primality Test Via Quantum Factorization”, Hoi Fung Chau, Hoi-Kwong Lo
[29] https://en.wikipedia.org/wiki/Primality_test
[30] “Fibonacci Pseudoprimes and their Place in Primality Testing”, Carly Allen
[31] “A Fast Monte-Carlo Test for Primality”, Robert Martin Solovay, Volker Strassen
[32] “The Little Book of Bigger Primes”, Paulo Ribenboim
[33] “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”, Peter Williston Shor
[34] “Improving the Speed and Accuracy of the Miller-Rabin Primality Test”, Shyam Narayanan
[35] https://en.wikipedia.org/wiki/Pocklington_primality_test
[36] “Number Theory for Computing”, Song Y. Yan
[37] <https://www.geeksforgeeks.org/segmented-sieve/>
[38] “Implementation of the Atkin-Goldwasser-Kilian Primality Testing Algorithm”, Francois Morain

[39] “Are there counter-examples to Baillie – PSW primality test?”, Carl Pomerance
[40] “New Primality Criteria and Factorizations of $2^m \pm 1$ ”, John Brillhart, Derick Henry Lehmer, John Lewis Selfridge
[41] “Multiplying huge integers using Fourier Transforms”, Ando Emerencia
[42] “The Pseudoprimes to $25 * 10^9$ ”, Carl Pomerance, John Lewis Selfridge, Samuel Standfield Wagstaff, Jr.
[43] “Fast Primality Testing for Integers That Fit into a Machine Word”, Michal Forišek, Jakub Jančina
[44] “Primality testing with Gaussian periods”, Hendrik Willem Lenstra, Jr., Carl Pomerance
[45] “The Analysis and Implementation of the AKS Algorithm and Its Improvement Algorithms”, Hua Li
[46] “Run Time Efficiency and the AKS Primality Test”, Roeland Singer-Hainze
[47] “Computational Number Theory and Algebra, Lecture 9”, Markus Bläser, Chandan Saha
[48] “Lucas Pseudoprimes”, Robert Baillie, Samuel Standfield Wagstaff, Jr.
[49] “Algorithmic Engagements 2018, task PIN solution, <https://sio2.mimuw.edu.pl/c/pa-2018-1/s/293818/source/>”, Łukasz Grządko
[50] “Algorithmic Engagements 2015, task Poszukiwania solution, <https://sio2.mimuw.edu.pl/c/pa-2015-1/s/88254/source/>”, Tomasz Czajka

About the author

My computer adventure began at the age of 8 and just two years later I started programming interactive video games in Basic language. I have participated in many maths and programming competitions and decided to widen and strengthen my math and computer skills by attending and graduating in Computer Science at the University of Wrocław. I have interests that span from the theory and mathematics to algorithms and data structures to application and software. I am also a TopCoder algorithm member since March 2005.

Łukasz Grządko

Senior Engineer, Software Development
MN LTE

Laboratory testing tools



2.1

Bartłomiej Radwan

Network tools – whim or must?
Overview of the network tools used
in the R&D labs

64

2.2

Patryk Furman

Hardware resources monitoring
with Cacti

70

2.3

Tomasz Szandala

One play to rule them all:
write Your first Ansible playbook

76

Network tools – whim or must?

Overview of the network tools used in the R&D labs

Bartłomiej Radwan
Specialist, Lab Network
MN CDS Lab Operations



1. Introduction

The question in the title seems to be quite rhetorical and every-one should esteem after a short reflection that there is only one correct answer “it depends”. Two matters have to be considered. The first one is the size of network under management. If an administrator takes care of two networks with 32 IP addresses each, is there a point to install an advanced tool? Tool installation will consume time, money, hardware resources. Maybe it is better to just keep crucial data in well-prepared static documentation and update them manually? The second matter is the correct definition of “network tools”. A perfect solution should be interactive (it should enable dynamic data collection), automated (task scheduler), easily accessible (online access through a web browser). Optionally, it should store historical data for a defined amount of time, be easily manageable, have a clear and simple graphical user interface, allow customization. This article will focus on the practical aspect. Below you will see three different tools which are used in Nokia laboratory in Wrocław.

2. Network tools

In the beginning, most information was stored in the form of electronic documentation, for example MS Excel, SVN or some other type of database. Common practices were: using PuTTY and gaining access directly to the graphical interface of the end device. Network tools have been evolving and with time static documentation has migrated from the old solution to a new one.

2.1 SolarWinds – IP Address Manager

IPAM is one of the solutions delivered by SolarWinds company. SolarWinds offers many tools and applications to manage modern networks, which are considerably more complex – and their complexity is constantly increasing. It arises mainly from two factors: a large number of devices and diversity of them in a network. Typically, networks may consist of hundreds of thousands of hardware devices like routers, switches, workstations, servers, other IP manageable equipment. Because of the increase of network complexity, there is a need to use automated tools to perform network management – one of these tool is IPAM. It has several major tasks to perform, including checking usability of network IP addresses (how many addresses are available for users) and collecting additional data from hosts, mainly using SNMP (Figure 1).

In a clear and simple graphical interface every user can find all available information about their network and their basic parameters (e.g. gateway, mask, VLAN) or information about a single host (Figure 2) in the matter of seconds. Every IP address, which is used by any type of device, is allocated in a database with additional information:

current status (Figure 2), status icon (Table 1), comment, type of device, person responsible for equipment, physical location, other self-defined custom fields.

Figure 1 Detailed view of single IP address



IP Address Details		EDIT HELP
MANAGEMENT	Edit IP Address	
STATUS		Used
TYPE	Static	
IP ADDRESS	10.42.129.253	
DUAL STACK IPV6 ADDRESS		
MAC ADDRESS	8C-60-4F-BB-2C-FC	
LAST SYNCHRONIZATION	2018-10-04 09:03:24	
COMMENTS	reserved for HSRP	
MACHINE TYPE	Cisco Nexus 5696Q	
VENDOR		CISCO SYSTEMS, INC.
SYSTEM NAME	gta-core-950-1.lte	
DESCRIPTION	Cisco NX-OS n6000, Software (n6000-uk9), Version 7.3(2)N1(1), RELEASE SOFTWARE Copyright (c) 2002-2012, 2016-2017 by Cisco Systems, Inc. Device Manager Version 6.0(2)N1(1),Compiled 5/12/2017 23:00:00	
CONTACT	RMA_LTE_Team	
SYSTEM LOCATION	GTA-CORE-950-1	

Table 1 Description of IP address icons





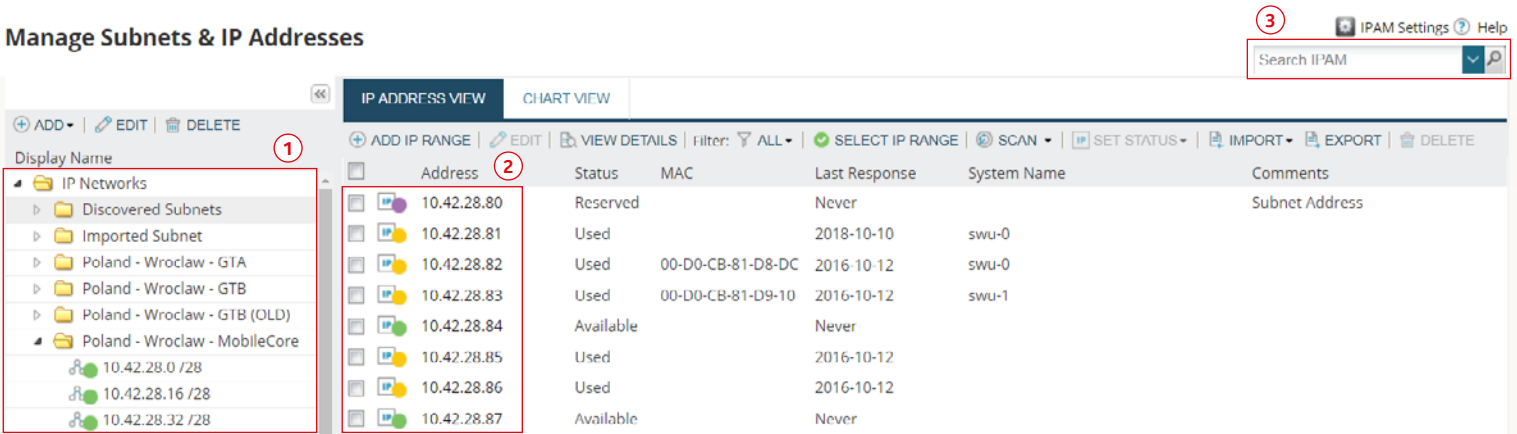
Icon – Color	Status	Status description
 Purple	Reserved	The address is not scanned, typically it is only used for network and broadcast addresses.
 Yellow	Used	The address responded in the last scan or has been manually assigned.
 Cyan	Transient	The address had been used, however during the last scan it did not respond or there is a problem with routing on removed system.
 Green	Available	The address has not been used yet or has been erased from its database manually.

Figure 2 Main graphical interface (1 – basic network list, 2 – IP address entry, 3 – search option)



Everything is presented in web browser windows, which allows users to get all necessary information instantly, even via a mobile device. From network administration point of view, the most important information is the status of usability of subnets, which can be determined by the icon and its color (Figure 2., Table 2.). If the available IP addresses are running out, there are two options. The first one: simply add another network. The second option is to try to find IPs which have “Transient” status and do not respond for a long time, ask the person responsible for them if they are used or not. IPAM has an extremely useful feature – network scan, which detects if IP addresses are occupied or not. There are two types of scans: automatic and manual. Automatic scans are performed once per a defined amount of time (in our case once per day) on each network separately. IPAM schedules all scans in a queue to avoid traffic congestion. Results are written into a database and immediately forwarded to a web interface, where users can easily find them. Manual scans, which can be triggered only by users with specific access rights. It allows acquiring the most current information about every IP address in a single subnet in the fastest possible way. For a common user it is enough to work with the tool, however SolarWinds company has implemented a few additional features, such as integration with Active Directory (using a domain account to log in), DNS and DHCP services. These additional features are not visible by a normal user, but they seem to be very useful for network administrators. Regular users have many limitations (their accounts are read-only) and cannot change anything inside the database. However, there is an option to change default rights to allow users to participate in filling tables with data or even perform a scan. One more additional feature is worth mentioning. IPAM has been fully integrated with Virtual Environment Manager (more information in “Virtual Environment Manager cloud solution. How to create fully automated R&D laboratory” by Bartłomiej Radwan, Łukasz Szewczyk), which allows users to create and manage virtual machines on their own. The feature

uses a Python script to allocate an available IP address and insert necessary details into an SQL database without any action from the administrator.

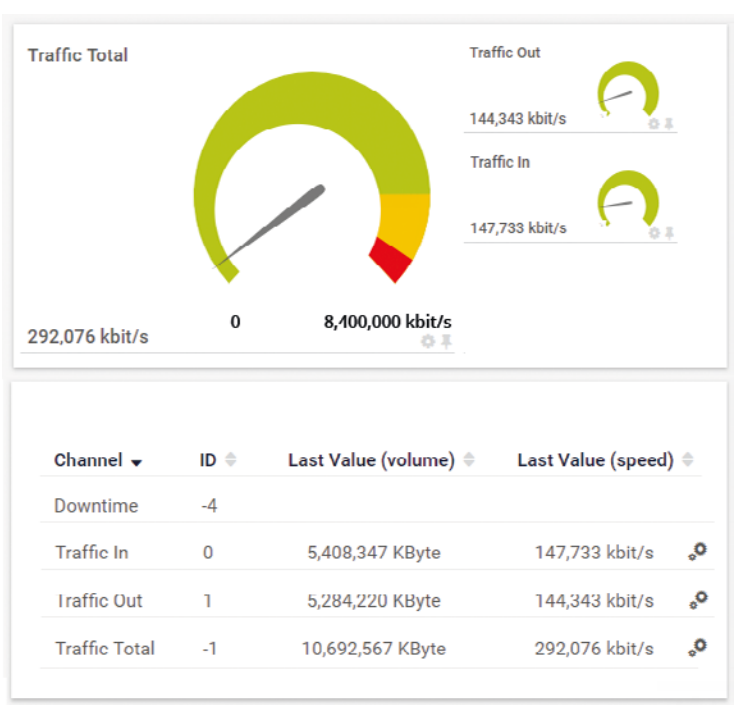
Table 2 Description of subnets icons

Icon	Status	Status description
	Good	Less than 60 percent of all possible subnet addresses are designated as Used.
	Warning	60 to 80 percent of all possible subnet addresses are designated as Used.
	Critical	At least 80 percent of all possible subnet addresses are designated as Used.

2.2 Paessler – PRTG Network Monitor

If there is a place to find direct mapping between specific IP addresses, a device, physical location, it would be very convenient to judge whether the device is working as expected or is it overloaded, is it being used. That is why PRTG Network monitor has been developed. It is an another product which is a commercial solution and it is used inside the Nokia laboratory network in Wrocław. The application uses different methods like GET/POST HTML, SSH, SNMP, WMI to collect necessary data and determine the status of a specific device (e.g. router, switch, servers) or service (e.g. DNS, FTP, VNC) on the monitored system. PRTG is based on specific objects which are called sensors (**Figure 3**).

Figure 3 Example of SNMP Traffic Sensor with defined limits



Every sensor performs a single set of actions: sends a request to the monitored system (e.g. ping, DNS query, SNMP query, etc.), waits for a response, collects data, decides if data is valid or not, waits a pre-define amount of time to perform the same loop again. There is a wide variety of sensors provided by Paessler company (more than 200 types), some of them based on popular protocols such as SNMP. Other sensors are dedicated to specific vendors (e.g. VMware, NetAPP, Microsoft). When the provided sets of queries cannot fulfill requirements, there are several options to extend the capability of sensors, such as importing vendor’s MIB (Management Information Base) libraries (only for SNMP) or creating a script (e.g. Bash, Python, etc.). After confirmation that sensors are capable of collecting significant data, it is time to start adding objects to the monitoring system. The procedure is straight forward, just open web browser window, log in, add the device of your choice and choose one of the two ways of creation of sensors: manual (add one by one) or auto-discovery (application will scan automatically added device and add as many sensors as possible). Auto-discovery takes more time. Each sensor works independently and can collect data in different time periods, from once per 10 seconds up to once per a few days. The user has to determine which devices should be monitored more or less often to avoid a situation in when the monitoring system will run out of

resources. When all necessary devices are added to PRTG and data starts filling the database, it is time to set values to the correct state. These states are used to determine if there is a problem with monitored devices (**Figure 3**). When a threshold is exceeded, PRTG generates a notification to inform the user or group of users about detected changes in the monitored device. Notifications can be send via e-mail or text message to a mobile phone. Such notification system allows users to stop staring at the monitoring system and act when an incident appears. Collected data is not deleted immediately after receiving, it is stored in the database and can be used to create statistics or reports about status or stability of the network or devices. Just as in case of the SolarWinds IPAM product, it is possible to create local or domain users accounts and assign dedicated access rights to the application. This way, users can define their own “mini” monitoring system with notifications. Such monitoring system has been recently implemented for the Unified Test Environment project.

2.3 Switch Config Tool

The last but not least tool, Switch Config Tool, has a significant impact on testers’ daily life, is an in-house product, developed by the CDS Lab Ops team. It has been created because of a few reasons: to expand users’ network awareness, to decrease the number of questions asked about switch configuration, and to lower the number of incorrectly created JIRA tickets. SCT (Switch Config Tools) connects http service with a Python script, which together allows checking the state of switch configuration in laboratory racks. The tool offers two options: port status and monitor configuration (**Figure 4**).

The first option presents detailed information about every port on the selected switch, its status, mode (access, trunk, or monitoring), errors, description, and other parameters. The second option informs about the number and type of SPAN sessions configured on the specified device. SPAN sessions are used to monitor traffic on other ports or vlans. Using SCT is simple, user selects localization and switch from a (drop-down) list) through web browser window, then decides which information is needed and clicks on the correct button. Results are displayed after a few seconds (**Figure 5**).

At the beginning, the tool was dedicated only to Cisco devices, however due to the expansion of Nokia’s portfolio, the need to add other manufacturers has come up. The tool is currently capable to work with hardware from Cisco, Quanta, and Dell, but the list is not closed and it is possible to expand it in the future. The same applies to port parameters and supported options. The next step forward to meet user expectations is unification of data presentation. Ports configuration is read directly from the device and presented in a common format, independently from the chosen vendor.

Figure 4 Switch Config Tool Page

NOKIA

Cloud Development Service - Switch Config Tool

GTA Laboratory

(Select switch) ▼

Port

Monitor

GTB Laboratory

(Select switch) ▼

Port

Monitor

WL Laboratory

(Select switch) ▼

Port

Monitor

Figure 5 Example of using Port configuration button

```
#####
# DEVICE TYPE: cisco      SWITCH:  10.42.160.50  (gtb-acc-001)      #
# Software version: 15.2(2a)E1                                     #
#####

Port      Description      Link  Status  Mode  Speed  MTU  P  IN  Vlan
Gi1/0/1   TL350_ENB_LMP         Up    enabled 3150  100M   9198 -  --
Gi1/0/2   TL349_ENB_LMP         Up    enabled 3149  100M   9198 -  --
Gi1/0/3   TL1906_ENB_LMP        Up    enabled 214    1G     9198 -  --
Gi1/0/4   TL325_ENB_LMP         Up    enabled 3125  1G     9198 -  --
Gi1/0/5   TL1901_ENB_LMP        Up    enabled 200    1G     9198 -  --
Gi1/0/6   TL1902_ENB_LMP        Up    enabled 201    1G     9198 -  --
Gi1/0/7   TL1904_ENB_LMP        Up    enabled 204    1G     9198 -  --
Gi1/0/8               Down  enabled 1      n/a    9198 -  --
Gi1/0/9               Down  enabled 1      n/a    9198 -  --
Gi1/0/10              Down  enabled 1      n/a    9198 -  --
```

3. Summary

True network tools are an integral part of modern networks, without them it would be almost impossible to control or monitor services in an organized way. Returning to the question in the title, in reference to mentioned parameters and awareness of the size of managed infrastructure, the presented tools are not a whim, they are a real must. Without them network management would become chaos which couldn't be controlled in any way.

References

[1] <https://www.solarwinds.com/ip-address-manager>
[2] <https://www.paessler.com/>

About the author

Bartłomiej Radwan is currently an R&D Laboratory Engineer in Cloud Development Services in Wroclaw. He joined Nokia 7 year ago as a test automation developer and moved to CDS in 2015 to expand his passion, which is networking. He is currently focused on development of tools and management of laboratory network.

Bartłomiej Radwan
R&D Laboratory Engineer
MN CDS

Hardware resources monitoring with Cacti

Patryk Furman
Technical Leader, Test Automation
MN SRAN Pz



1. Introduction

Resources monitoring is a part of DevOps methodology cycle. It can provide much useful information about hardware average and peak utilization values. It can also prevent system downfalls, which may occur due to lack of computing power. While working in continuous delivery or DevOps, a failure in production chain can cause high losses and hold up workflow on dependent components. With properly configured resources monitoring, recovery actions can be triggered if an unusual situation would be detected (i.e. too high network traffic on a certain interface, lack of storage space on the server). An administrator can be immediately informed via SMS or email, or an external script can be launched. Furthermore, all gathered information could help estimate what additional equipment is needed to serve continuous delivery without any blunders. This article describes Cacti monitoring utility and discusses some of its use-cases.

2. Cacti overview

Cacti is an open-source monitoring tool. Its main features include:

- **data gathering** – Cacti uses the SNMP protocol for querying network hosts about their resources. Additionally, users can specify their own scripts for some custom queries.
- **data visualization** – Cacti uses RRDTool for data visualization in the form of graphs. Graphs can be grouped into hierarchical trees and the view can be customized along with time span which users want to access.
- **data storage** – all data which are presented to users are also stored inside a MySQL database and RRD files.

- **front-end access** – the whole application can be accessed and controlled via a web browser.
- **user management** – users can log in to the application with previously given credentials or via their Intranet accounts (if the Cacti administrator configured the LDAP protocol properly). Each user or user group can get certain permissions (i.e. only view graphs or edit data sources).
- **templates** – Cacti lets the user save a device, a data source, as a template, which can be later used for making repetitive tasks easier (i.e. creating the same graph type for many network hosts).
- **functionality expansion with plugins** – Cacti owners and some open-source contributors provide users with additional plugins.

The installation process is clearly described on the Cacti main page [1]. Its main dependencies are Apache, MySQL database (which can be also replaced by MariaDB), PHP, SNMP utilities, and RRDTool for graphs creation. Source code of the project is stored on GitHub [2] and it's contributed to by many people. The general logic, which Cacti follows, is maintaining given data sources and visualizing them in the form of charts. Under the hood, the software is using a pooler process which gathers information in fixed time intervals. Everything caught by this process is then saved to a database.

There are many data sources defined as built-ins, which consist of:

- SNMP data inputs (CPU, RAM, HDD, network interface traffic, load average)
- Linux/Unix scripts

Figure 1 shows sample graphs generated with the use of SNMP data sources.

Figure 1 Sample built-in templates graphs



Cacti supports additional options regarding finding and detecting of data sources available on a host in the network. Users can specify the IP address of the device which they want to monitor. The software will try to communicate with device via a SNMP query and automatically detect all information which can be gathered and graphed. SNMP daemon needs to be launched and properly configured, so that the device would be able to answer correctly. **Figure 2** shows sample data sources detected after fetching from virtual machine.

Figure 2 Device data sources

Graph Templates	
Graph Template Name	
Host MIB - Logged in Users	
Host MIB - Processes	
Net-SNMP - Combined SCSI Disk Bytes	
Net-SNMP - Combined SCSI Disk I/O	
Net-SNMP - Context Switches	
Net-SNMP - CPU Utilization	
Net-SNMP - Interrupts	
Net-SNMP - Load Average	
Net-SNMP - Memory Usage	
Create (Select a graph type to create)	
Data Query [Net-SNMP - Get Monitored Partitions]	
Index	Mount Point
1	/var
2	/
3	/run
4	/dev/shm
5	/run/lock
6	/sys/fs/cgroup
7	/boot
8	/opt
9	/tmp
10	/home
11	/run/user/116
12	/media/burak
13	/run/user/1000

Users can choose which data they want to get and they can create corresponding graphs. All devices can be also saved and placed in hierarchical view for better access. Thanks to that, all previously chosen sources are visible within the host label so they can be easily found. **Figure 3** shows sample devices grouping for three categories – virtual machines, servers, and network switches.

Figure 3 Devices trees

TA Virtual Machines	
TA_servers	
Tatools_server	
PMS_server	
Hran_master_Jenkins	
TA_switches	
Core_switch_rack_220	
Core_switch_rack_145	

Cacti also provides a bunch of templates for different objects:

- device templates** – users can choose one of the given templates when querying the network host. Cacti will then map all graphs associated with the template to the device’s data sources.
- graph templates** – users can choose from many graph templates which provide a variety of information formats (i.e. traffic in bit/s or bytes/s format, load average in different time intervals).
- data source templates** – templates of data gathering methods. Thanks to them, users are not forced to define their own methods of data gathering for simple queries as they are already included with built-ins (SNMP queries, Linux/Unix scripts for information fetching).

Using all these templates, users can easily create a basic data monitoring system for many device types (routers, switches, Linux and Windows machines) without much effort.

Cacti also supports multi user settings. Users can be placed inside a group with specific permissions. Permissions are connected to graphs, templates, and device management and can be configured either for a group or a single user.

Administrators can follow all events which are happening inside Cacti and its plugins via logs listings also available on the main server page.

3. Use cases

Depending on what devices are used, use cases of the monitoring system can vary. Below are examples connected to system test automation environments, which consist of real BTSes managed via virtual machines. Other network devices available in the test

stands are network switches and power distribution units (PDUs). All mentioned hosts are connected and monitored by Cacti via SNMP protocol.

3.1 Comparing gathered data for hardware expansion and issues debugging

All data is saved in a database and can be accessed via Cacti web GUI. Administrators can easily check if there weren’t any deviations among all the devices in a time frame. It is very handy information for when hardware resources inspection is underway. The most frequent issue in this area is lack of RAM or hard drive space which can lead to inoperable environment. Such statistics could be an unquestionable argument for a laboratory maintainer that devices should be expanded with additional equipment.

Another situation when accessing historical resources monitoring would be useful is a memory leak or high network traffic issues debugging. Properly configured monitoring will show how much RAM is being used, cached, and free and it will visualize how much flow is going through the network interface in both directions.

Figure 4 Thold events log

Device	Time	Type	Event Description	Alert Value	Measured Value
Tatools_server	2018-07-02 16:25:27	High / Low	NORMAL: Tatools_server - Used Space - / [hdd_used] Restored to Normal Threshold with Value 150000996352	N/A	139.70 G
Tatools_server	2018-07-02 08:00:27	High / Low	ALERT: Tatools_server - Used Space - / [hdd_used] went above threshold of 1600000000000 with 165553336320	149.01 G	154.18 G
Tatools_server	2018-07-02 07:45:27	High / Low	NORMAL: Tatools_server - Used Space - / [hdd_used] Restored to Normal Threshold with Value	N/A	N/A
Tatools_server	2018-06-30 14:15:28	High / Low	ALERT: Tatools_server - Used Space - / [hdd_used] went above threshold of 1600000000000 with 171455283200	149.01 G	159.68 G
Tatools_server	2018-06-30 14:00:28	High / Low	NORMAL: Tatools_server - Used Space - / [hdd_used] Restored to Normal Threshold with Value	N/A	N/A
Tatools_server	2018-06-29 15:05:27	High / Low	ALERT: Tatools_server - Used Space - / [hdd_used] went above threshold of 1600000000000 with 166983876608	149.01 G	155.52 G

Figure 5 Detected devices list

Device Name	IP	SNMP Name	Location	Contact	Description
iv5-66-lnx	10.42.35.253	iv5-66-lnx	Sitting on the Dock of the Bay	Me me@example.org	Linux iv5-66-lnx 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u1 (2016-09-03) x86_64
iv5-65-lnx	10.42.35.252	iv5-65-lnx	Sitting on the Dock of the Bay	Me me@example.org	Linux iv5-65-lnx 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u1 (2016-09-03) x86_64
iv5-69-lnx	10.42.35.244	iv5-69-lnx	Sitting on the Dock of the Bay	Me me@example.org	Linux iv5-69-lnx 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u1 (2016-09-03) x86_64
iv5-73-lnx	10.42.35.254	iv5-73-lnx	Sitting on the Dock of the Bay	Me me@example.org	Linux iv5-73-lnx 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u1 (2016-09-03) x86_64
iv5-74-lnx	10.42.35.247	iv5-74-lnx	Sitting on the Dock of the Bay	Me me@example.org	Linux iv5-74-lnx 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u1 (2016-09-03) x86_64
iv5-29-lnx	10.42.35.249	iv5-29-lnx	Sitting on the Dock of the Day	Me me@example.org	Linux iv5-29-lnx 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u1 (2016-09-03) x86_64
iv5-78-lnx	10.42.35.245	iv5-78-lnx	Sitting on the Dock of the Bay	Me me@example.org	Linux iv5-78-lnx 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u1 (2016-09-03) x86_64
iv5-24-lnx	10.42.35.248	iv5-24-lnx	Sitting on the Dock of the Bay	Me me@example.org	Linux iv5-24-lnx 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u1 (2016-09-03) x86_64
iv5-68-lnx	10.42.35.243	iv5-68-lnx	Sitting on the Dock of the Bay	Me me@example.org	Linux iv5-68-lnx 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u1 (2016-09-03) x86_64
iv5-08-lnx	10.42.35.250	iv5-08-lnx	Sitting on the Dock of the Bay	Me me@example.org	Linux iv5-08-lnx 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u1 (2016-09-03) x86_64
iv5-19-lnx	10.42.35.242	iv5-19-lnx	Sitting on the Dock of the Bay	Me me@example.org	Linux iv5-19-lnx 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u1 (2016-09-03) x86_64

Using device templates, users can map specific threshold to a certain device type. When such option is selected, thresholds generate automatically for the given resource, after device graphs creation.

3.3 Discovering hosts from a specific network

Cacti supports automatization of the process of adding devices. If a specific subnet is reserved for your teams’ devices, you can simply scan it and add the found hosts to the monitored ones. For example, a separate subnet can be reserved for virtual machines connection and a separate one for network switches access. If all hosts support SNMP querying, hosts will be detected by Cacti during a network scan.

3.4 Logging via Intranet accounts and grouping users accounts

Cacti supports LDAP authentication so it is possible to use LDAP server. It simplifies user logging and permissions passing. When a user logs in for the first time using his Intranet credentials, Cacti automatically uses a chosen template for him with specified permissions and creates a new account. Guest account is set as the default one, which has permissions only for graph viewing.

Users can also be grouped. The administrator can for example create separate groups for developers (with higher privileges) and for testers (with lower privileges).

3.5 Periodical reports of device resources consumption

Reporting functionality gives an opportunity to send an e-mail message with the chosen devices’ graphs in constant periods. The administrator can inform device owners about the level of device utilization per exact interval. This is useful information for deciding if additional computing power or even new hardware is necessary.

4. Conclusion

Cacti is a simple solution to set up a monitoring system focused on information visualization. RRDTool, also open-source, provides readable graphs which are dynamically created independently from the chosen time slot. The main advantage of Cacti is the usage of the SNMP system service in client-server communication. Cacti doesn’t require any additional client utility to be installed on the destination host. After monitoring server setup, all desired devices can be found automatically via a network scan which speeds up the whole “ready to serve” process. Lack of more advanced alarming system is a small disadvantage. Using the *thold* plugin, an administrator can set up only basic rules and alarm users via e-mail (no SMS notification is available). Nevertheless, with small effort: users are able to set up an elegant monitoring and reporting system.

References

- [1] Cacti main website, <https://www.cacti.net/> (access 04.05.2018),
- [2] Cacti source code on GitHub platform, <https://github.com/Cacti/cacti> (access 04.05.2018),
- [3] Main page of Cacti *thold* plugin, <https://docs.cacti.net/plugin:thold> (access 04.05.2018)

About the author

I am a graduate of the Electronics faculty at Wroclaw University of Technology. My journey in Nokia began in 2014 when I was hired as a manual tester. I currently work on the test automation team as a technical leader. We’re maintaining a framework for manual testers to speed up their work in system testing. I’m interested in finding sufficient solutions for repetitive tasks which can be done by a computer program with minor human interaction. Privately, my interests include network security and Linux based operating systems.

Patryk Furman

Technical Leader, Test Automation
MN SRAN Pz

Server Provisioning with Ansible

Tomasz Szandala
Software Configuration Engineer
MN CDS



The proliferation of cloud computing solutions has led to a significant uptick in the number of servers that need to be configured and managed within organizations. Not so long ago administrators had to deal with racks of physical servers that could be accessed in the local data center, but nowadays we have to manage thousands of servers that could be spread all around the globe. Here comes Ansible - one of the most popular provisioning software. In this paper I will present how to write our first playbook and use it to configure a sample host for deploying a simple Python-Django application.

1. What is Ansible?

Ansible is an open-source IT automation engine which can remove drudgery from your daily work, and will also dramatically improve the scalability, consistency, and reliability of your IT environment. Ansible is a software that allows you to write unified configuration files that can set up a server, a container, or even a specific application on demand.

With Ansible it is possible to automate three types of tasks:

- Provisioning: Set up the various servers you need in your infrastructure.
- Configuration management: Change the configuration of an application, OS, or a device; start and stop services; install or update applications; implement a security policy; or perform a wide variety of other configuration tasks.
- Application deployment: Make DevOps easier by automating the deployment of internally developed applications to your production systems.

Ansible can automate IT environments whether they are hosted on traditional bare metal servers, virtualization platforms, or in the cloud. It can also automate the configuration of a wide range of systems and devices such as databases, storage devices, networks, firewalls, and many others.

A huge advantage is that you do not even need to know the commands used to accomplish a particular task. You just need to specify what state you want the system to be in and Ansible will take care of it.

1.1 Why Ansible?

Apart from Ansible, there are at least three other provisioning solutions. Ansible has been chosen by my Team for two reasons: first, it does not require to have any additional software on client hosts, only Python – which we already had. Secondly, we have considered

Ansible to have the shallowest learning curve among other similar solutions like Puppet or Chef.

Table 1

Feature	Ansible	Puppet	Chef
Release	2012	2005	2009
Language	Python	Ruby	Ruby
Configuration scripts language	YAML	Unique DSL	Unique DSL/Ruby for advanced
File templates format	Jinja2	ERB	ERB
Software on clients	Only Python needed	Needs a specific client on slave	Needs a specific client on slave
Communication protocol	SSH	Based on SSL	Based on SSL
Compiled configuration	No	Yes	Yes

After a few months of using Ansible, the only disadvantages we have discovered are performance slowdowns on simultaneous configurations of dozens of hosts (but this has a negligible impact on our environment) and a lack of consistency check before the configuration. The last one might cause problems one day, for example Puppet first compiles the configuration, therefore it warns if there are conflicting requested packages (Python 3.4 and 3.5). Ansible will just proceed and we will end up with the last requested one during play (“a play” is called the execution of playbook(s) for a given host).

“Keep it simple, stupid” (KISS) is the underlying principle of Ansible, a configuration management and orchestration tool. In that sense, here is a simple introduction.

2. Deploying Django applications

This paper will describe how to use Ansible to deploy an application written in Python with Django framework and PostgreSQL. What are the basic steps to deploy the Django application?

1. Get a server with a chosen system. For this example choose a host with a Centos 7.3 operating system.
2. Create a user, for example “django”, who will host the application.
3. Fetch the Git repository with our application to the server.

- 4. We need to install and configure python-virtualenv (preferably using requirements.txt).
- 5. Install and launch PostgreSQL database.
- 6. Install and configure Gunicorn service.

2.1. Creating a “django” user

Performing a configuration using Ansible is all about Yaml playbooks.

Listing 1 Starting the playbook – creation of a new user

```
---
- hosts: all
  vars:
    app_user: "django"

  tasks:
    - name: Creating a user {{ app_user }}
      user:
        name: "{{ app_user }}"
        home: "/home/{{ app_user }}"
      register: app_user_creation
```

First, we have to specify which servers this playbook should be run against, then we list the tasks. In this case we are allowing any host to be the target. Here we can specify the DNS name of the host(s), the IP, or a reference to a script that can dynamically generate a list of hosts. If we keep assigning playbooks to only strictly defined hosts (instead just “all”), it will prevent us from an accidental triggering configuration of the playbook against the restricted host.

Next comes the “vars” section that specifies variables used throughout the entire script. Now if we want to change the username, we can do it in one place, instead of searching through all tasks. It is also worth to mention the variables precedence: the most important variables come from calling the script from command line, next comes the ones from the main playbook, then separate vars files, other playbooks (a playbook can call another playbook inside), and lastly – default values.

Finally appears the “tasks” section – a list of tasks, one by one to be performed on a given host(s). In the abovementioned simple playbook, there is only one task: creating a new user with a name and home directory. Ansible comes with a very well written documentation, which is available online or by using the ansible-doc command tool, for example “ansible-doc user”. Each task can be “registered”, which means that we can refer to it later.

Now we can call this play using a command:

Listing 2 Running our first playbook

```
$ ansible-playbook -i "10.157.53.18, "
--private-key ~/.ssh/magic_key -u root django_app.yml
```

2.2. Fetch Git repository with sources of our application

It is the time to make our code available on the server. This step consists of two parts: preparing authorization for Git and cloning Git repository. Since the preferred method is SSH cloning, it is reasonable to add the private key to the user’s home that gives the read access to application’s repository. Also, it is reasonable to keep all ansible-associated files in one repository but keeping a private key in repository is not safe, unless we encrypt it. Ansible comes with batteries included – there is an ansible-vault program that allows encrypting passwords, private keys, or any other data we want to keep in secret. It uses 256-bit AES encryption, which should be sufficient for most use cases. To encrypt the file, we have to type:

Listing 3 Encryption of a id_rsa_key file (a chosen private key)

```
$ ansible-vault encrypt id_rsa_key
```

First you will be asked for a password. The password used with a vault must currently be the same for all files you wish to use together at the same time.

After preparation we can go back to our playbook and add:

Listing 4 Playbook steps to configure authorization to Git and finally cloning the repository

```
- name: Set the authorized key of a starting user
  authorized_key:
    user: "{{ app_user }}"
    state: present
    key: "{{ lookup('file', '~/.ssh/id_rsa.pub') }}"

- name: Copy the encrypted SSH key
  copy:
    src: django_ssh_key
    dest: /home/{{ app_user }}/.ssh/id_rsa
    mode: 0600
  become: yes
  become_user: "{{ app_user }}"

- name: Cloning app’s repository
```

```
git:
  repo: "ssh://gerrit.nokia.com:29418/NOKIA/B00K/django"
  dest: "~/django-app"
  accept_hostkey: yes
  become: yes
  become_user: "{{ app_user }}"
```

On a side note, let’s take a closer look at the authorized_key/key entry – “lookup”. Lookups allow to access outside data sources. Like all templating, these plugins are evaluated on the Ansible control machine (the one running the playbook), and can include reading the filesystem, as well as contacting external datastores and services. This data is then made available using the standard templating system in Ansible.

2.3. Python and virtualenv

A clean environment in the host always helps to maintain the system. To keep Python pure, we should use python-virtualenv and install its packages. This step consists of two substeps: installing python34-virtualenv and installing the list of modules from requirements.txt.

Listing 5 Steps for virtualenv installation and its population with modules

```
- name: Install python34-virtualenv
  yum:
    name: python34-virtualenv
  environment:
    http_proxy: "http://1.2.3.4:8080"
- name: Install requirements
  pip:
    requirements: "~/django-app/requirements.txt"
  virtualenv: "~/venv"
  virtualenv_python: python3.4
  virtualenv_command: virtualenv-3
  become: yes
  become_user: "{{ app_user }}"
```

The most notable element is the environment section – the list of variables that should be set in an environment when executing a given task. Whenever I want to install something new in this playbook, I need to ensure that my office’s proxy server is set and I do it using the environment section. Since this element will be repeated, I prefer to put it at a “global” level of this play, below the “vars” section.

2.4. PostgreSQL installation and configuration

While my application will configure the tables in database, it must have a ready database in the system. We can of course have one in Docker’s container, but we will do it in an Ansible-way. Last of the most useful Ansible modules is the “ansible-galaxy”. Ansible is written in Python and adopts a similar solution: the community can freely prepare and share their own configuration. In Ansible those shared configurations take form of “roles” – a package with playbooks and resources files that only takes some variables and can be used to set up our hosts. Similar to an “ansible-doc”, we can go browse roles’ definition on the website or use an ‘ansible-galaxy’ command:

Listing 6 Searching and downloading a role from Ansible-galaxy

```
$ ansible-galaxy search postgresql
(...) # we choose one of the printed options
$ ansible-galaxy install -p ./ ANXS.postgresql
```

The role consists of structured directories and files. I use roles for my personal use but consider most of them as an overabundance. Roles are often written to support multiple systems and I have a rather unified environment in my work. In this case I cut out some of the role’s components or write my own, basing on the one from Ansible-galaxy.

Listing 7 Contents of the role’s directory

```
$ tree ANXS.postgresql
ANXS.postgresql/ # role’s root directory and name of the role
├─ ansible.cfg
├─ defaults      # default values for the role – common among systems
│   └─ main.yml
├─ handlers
│   └─ main.yml # special tasks to be executed at the end of the play, for example service restarts
├─ LICENSE
├─ meta
│   └─ main.yml # file with galaxy metadata, for example dependencies
├─ README.md
├─ tasks        # main part of the role – playbooks with tasks
│   └─ configure.yml
│   └─ databases.yml
│   └─ extensions
│       └─ contrib.yml
│       └─ postgis.yml
└─ install_yum.yml
```

```
|   ├── main.yml # main.yml is started by default
|                   when including the role
|   └── templates  # files in jinja2 that are populated with
|                   variables
|   ├── etc_apt_preferences.d_apt_postgresql_org_pub_repos_
|   |   apt.pref.j2
|   ├── postgresql.conf-10.j2
|   ├── postgresql.conf-10.orig
|   └── ... more
└── vars           # directory with variables – split per system
    ├── Debian.yml
    ├── empty.yml
    ├── RedHat.yml
    └── xenial.yml

11 directories, 57 files
```

When the role is downloaded, we should edit variables to meet our needs. We can pass variables to the role by setting them in our play-book or overwrite them in the role’s vars directory. Now we can add the role call to our playbook.

Listing 8 Task to call the role in the playbook

```
- name: Run the role to install PostgreSQL
  include_role:
    name: "ANXS.postgresql"
```

2.5. Install and configure Unicorn service

This step consists of two substeps: installing Unicorn software and running it by specifying our application to be launched.

Listing 9 Final steps to install and run Unicorn

```
- name: Install Unicorn
  yum:
    name: python-unicorn

- name: Run Unicorn
  unicorn:
    app: wsgi
    chdir: "/home/{{ app_user }}/django-app"
    venv: "/home/{{ app_user }}/venv"
    user: "{{ app_user }}"
```

As we may see, the yum task does not have the environment part, since I have moved it to the top level, as a global value. Moreover, it might be better to merge iterative steps like yum, pip, or copy to one call, instead of calling them multiple times, since it improves a bit of the performance of the play, but for this tutorial I will keep it simple.

3. Play summary

Listing 10 Output from the play of the fully prepared playbook

```
ansible-playbook -i "10.157.53.63,
" --private-key ~/.ssh/magic_key -u root django.yml

PLAY [all] *****
*****

TASK [Gathering facts] *****
*****
ok: [10.157.53.63]

TASK [Creating a django user] *****
*****
changed: [10.157.53.63]

TASK [Set an authorized key by starting the user] *****
*****
changed: [10.157.53.63]

TASK [Copy the encrypted SSH key] *****
*****
changed: [10.157.53.63]

TASK [Cloning app’s repository] *****
*****
changed: [10.157.53.63]

TASK [Install python34-virtualenv] *****
*****
changed: [10.157.53.63]

TASK [Install python34-virtualenv] *****
*****
changed: [10.157.53.63]

TASK [Run the role to install PostgreSQL] *****
*****

TASK [ANXS.postgresql : include_vars] *****
*****
ok: [10.157.53.63] => (item=/home/szandala/nokia/ci/ansible/
playbooks/ANXS.postgresql/vars/./vars/RedHat.yml)
```

```
TASK [ANXS.postgresql : PostgreSQL | Make sure the CA certificates
are available] *****
*****
skipping: [10.157.53.63]

(... very long output from the role, I will omit it ...)
```

```
TASK [Install Unicorn] *****
*****
changed: [10.157.53.63]

TASK [Run Unicorn] *****
*****
changed: [10.157.53.63]

PLAY RECAP *****
10.157.53.63      : ok=8   changed=25  unreachable=0    failed=0
```

The summary output shows that 8 steps have the “ok” status, so nothing had to be done, and 25 have the “changed” status, so they were executed. There are also a few marked as skipped – coming from the role execution, they were omitted under certain condi-tions, for example they do not apply to this system. In a perfect configuration, we would have only “ok” tasks and in the worst case – “skipped”, which means that the configuration did not differ from the one we set at the beginning.

4. Summary

Ansible is a simple automation language that can perfectly describe an IT application infrastructure. It is easy to learn, it is self-docu-menting, and does not require a grad-level computer science de-gree to read. Automation should not be more complex than the tasks its replacing.

Do not forget about a very well-prepared Ansible Tower solution. It is the RedHat’s answer for GUI dashboards provided by Chef or Puppet. There is of course also an open-source version – AWX, that is without the company’s support. Apart from the easier running playbooks, it also provides a logging mechanism, an access control, and can mimic agent-architecture. We can just set a configured host to periodically ask the Ansible Tower if there are any configuration updates. But this and many more features of Tower/AWX is a topic for a separate article.

References

- [1] <http://docs.ansible.com/>
- [2] <https://www.edureka.co/blog/chef-vs-puppet-vs-ansible-vs-saltstack/>
- [3] <https://docs.ansible.com/ansible/2.4/vault.html>
- [4] http://docs.ansible.com/ansible/latest/reference_appendices/galaxy.html
- [5] <https://galaxy.ansible.com/>
- [6] Lorin Hochstein, Rene Moser, Ansible: Up and Running. Automating Configuration Management and Deployment the Easy Way
- [7] Udemy course: Automation with Ansible based on Red Hat Training
- [8] <https://www.ansible.com/products/awx-project/faq>

About the author

I am a PhD student at Wroclaw University of Science and Technology in the field of Artificial Intelligence. I am also a member of the Creation & Development Group KREDEK, which is a university science club. In NOKIA I work as a Software Configuration Engineer in Wroclaw’s DevOps Team. My daily work consists of deploying and maintaining hundreds of hosts for multiple purposes: Jenkins, Web Applications, compilation servers, and so on. My favorite task in life is solving problems since with computers impossible is nothing, it just takes a bit more time.

Tomasz Szandala

Software Configuration Engineer
MN CDS

Tests and Laboratory development directions



3.1

Grzegorz Juszcak
Application of OpenStack Technology
for Managing R&D Virtual Machines
in 5G Infrastructure Laboratory

84

3.2

Wojciech Zmyślony
Passive Intermodulation Cancellation

90

3.3

Łukasz Szewczyk and Bartłomiej Radwan
Virtual Environment Manager Cloud solution.
How to create fully automated
R&D laboratory

96

3.4

**Łukasz Michna, Grzegorz Dygoń
and Stanisław Pospiech**
Migration towards future Data Center
Networking

100

3.5

Adam Chamera
Using Artificial Intelligence for radio
performance test result analysis

110

Application of OpenStack Technology for Managing R&D Virtual Machines in 5G Infrastructure Laboratory

Grzegorz Juszczak
Technical Leader, Lab System
MN CDS Lab Operations

NOKIA

1. Adopting OpenStack Technology in Wroclaw Infrastructure Laboratory

Nokia has been constantly evolving to meet its customers' requirements, expectations and stay competitive on the global telecom market. Our business units need a reliable and flexible IT infrastructure that meets their expectations in order to let them work on the continuous Nokia product development, and integration with the third party software. On the other hand, we are focusing on cost-effective solutions to reduce the expenses on our laboratory maintenance and operation.

All these aspects, that is dynamic Nokia business units development and focus on cutting operational costs, enforce the need of searching for a less expensive and responsive virtualization platform to run business units critical servers, and store their important project data.

Nokia 5G laboratory operation team has found a solution for business demands mentioned above, and that is cloud computing software. Cloud is a computer technology, which enables common access to shared system resources and IT services over Internet. All these resources and services can be deployed and provisioned rapidly with the minimum end-user effort. The real usefulness of cloud infrastructure is proven, when the corporation can focus on its main businesses and forget about scaling hardware resources and datacenter architecture planning, because in cloud environment, expanding infrastructure is as simple as adding new server to the existing cloud.

We chose OpenStack software as a default cloud computing platform to deliver laboratory infrastructure for main business lines. OpenStack is free, open-source, IaaS (Infrastructure as a Service) based cloud computing platform, developed as a joint venture of Rackspace Inc. and NASA. The software platform consists of coexisting services that control multi-vendor hardware based pools of virtual servers, storage, and networking resources, managed using centralized web-based dashboard or RESTful web services.

Main components of OpenStack are:

- **Horizon:** web browser user interface (dashboard) based on Python Django for creating and managing instances (OpenStack virtual machines)
- **Keystone:** authentication and authorization framework based on MariaDB database
- **Neutron:** OpenStack project providing "networking as a service" between interface devices (vNICs)
- **Cinder:** persistent block storage for OpenStack instances with multiple backend support
- **Nova:** instances management system based on Linux KVM (Kernel-based Virtual Machine) Hypervisor
- **Glance:** registry for instance images

- **Swift:** file storage for cloud
- **Telemetry/Ceilometer:** metering engine for collecting billable data and analysis
- **Heat:** orchestration service for template-based instance deployment

OpenStack is rapidly taking over global IT market, as well as end-users and supporters of this cloud technology, mainly due to its flexibility, integration capabilities with the third party technologies and open source, clear and easy to understand source code. Although still considered as a new technology, OpenStack in fact consists of common and mature background technologies. Compute nodes are based on well known and solid KVM – Kernel-based Virtual Machine which runs all the instances in cloud. Keystone authentication and authorization service runs on MariaDB database daemon, while messaging mechanism between nodes is based on AMQP (Advanced Message Queuing Protocol) and by default RabbitMQ message broker. Cinder service utilizes reliable and popular Linux LVM – Logical Volume Manager as a basic backend for persistent block storage. Of course Cinder service supports other third party technologies, i.e. GlusterFS, Ceph, Netapp, HPE 3PAR, Dell EMC.

2. Deployment Method for Production Environment

There are few methods of OpenStack deployment: Packstack, TripleO or Ansible. Packstack was the first official automated method of OpenStack deployment, introduced and supported by OpenStack RDO community. Nowadays, Packstack based deployments are usually performed in order to proof of concept installations. If you are a developer working on OpenStack integrated software and you want to test your application in OpenStack environment, it's best to choose OpenStack-Ansible (OSA) deployment, which quickly installs OpenStack components on isolated Linux Containers (LXC). However, if you are deploying a large production cloud environment, TripleO is the most accurate and reliable solution, and that's the way we deploy OpenStack in Nokia 5G infrastructure laboratory.

TripleO has an advantage over other mentioned installation scripts (Packstack, Ansible), mainly due to its focus on high availability and redundancy aspects of the deployment, and last but not least, unlike others – TripleO gives us the possibility of continuous OpenStack software upgrades of already deployed cloud. TripleO stands for „OpenStack On OpenStack" method of cloud deployment, which means it utilizes so called Undercloud node with installed OpenStack software, which acts as an installation server for the destination cloud environment, called Overcloud.

OpenStack version installed on Undercloud includes Glance image service to store the images of Baremetal nodes used in the destination environment, which include: Compute, Controller, Block Storage

and Ceph Storage nodes. Neutron service in cooperation with Open vSwitch (OVS) software provides control plane network, which is used by Ironic service to deploy Baremetal nodes using PXE boot and upload OS images via TFTP service. Before Opencloud deployment, the Ironic service checks the connectivity and gathers information from all nodes about their hardware capabilities by means of so called Introspection procedure, to check if all nodes are compatible with OpenStack requirements.

3. Nokia 5G Laboratory Redundant Cloud Architecture Overview

During design of OpenStack environment topology, we had to put an emphasis on high availability of our services, redundancy of the hardware and security aspects of hosted business infrastructure. That’s why we tried to design our cloud environment as reliable as possible to avoid any potential failures in future during cloud operations. The OpenStack architecture used in 5G infrastructure laboratory team has been presented in the **Figure 1**. Some elements of the architecture have been removed from the picture to make it clear.

To meet the High Availability services concept requirements, we decided to use the Quorum of three redundant Controller nodes, which control overall behavior of the cloud including network, storage and other resource management utilizing the Pacemaker/Corosync cluster software. This solution is meant to minimize system downtime, when the services in cloud are unavailable to the end-user, and protect the system against accidental data loss or deletion. Typical cumulative downtime of High Availability systems equals approximately one hour per year! In case of a Controller failure, a second redundant node is brought online immediately to take over running services. Switching requests between Controller nodes are handled using Virtual IP (VIP) address and HA Proxy service, and the whole switching procedure is managed of course by Pacemaker/Corosync cluster software.

In order to improve network reliability and potential troubleshooting of our OpenStack architecture, we have implemented the concept of network isolation. Default TripleO based OpenStack installation sends all the OpenStack related traffic using control plane / provisioning network, making Undercloud node a kind of proxy server, and a single point of access for Overcloud nodes. It’s a pretty safe solution, since the whole Overcloud infrastructure is hidden behind Undercloud node, but from the other hand, single point of access becomes a single point of failure. In case of Undercloud failure, the Overcloud nodes will be cut off from external network connectivity, and our business lines might lose access to the infrastructure services temporarily, or even permanently.

First of all, the network isolation concept used in our deployment let the Overcloud nodes become independent from Undercloud. Moreover, all Controller nodes are given IP addresses from external 5G

infrastructure laboratory network. This lets us control and manage our cloud using external API calls or create instances using automation tools, like VEM (Virtual Environment Manager), developed in our Nokia 5G infrastructure laboratory team. To protect the Controller nodes against unauthorized access attempts from our business departments, we devoted a separate network, called Floating IP, for them, to let them access their virtual machines (OpenStack Instances). In this way our service and maintenance OpenStack networks are completely isolated from the business traffic and inaccessible to people outside 5G laboratory infrastructure team. Moreover, the network isolation idea let us distinguish OpenStack core service networks which are as follows:

- Control Plane / Provisioning network
- Tenant network
- Internal API network
- Storage network
- Storage Management network
- BTS LMP (Base Transceiver Station Local Management Port) networks

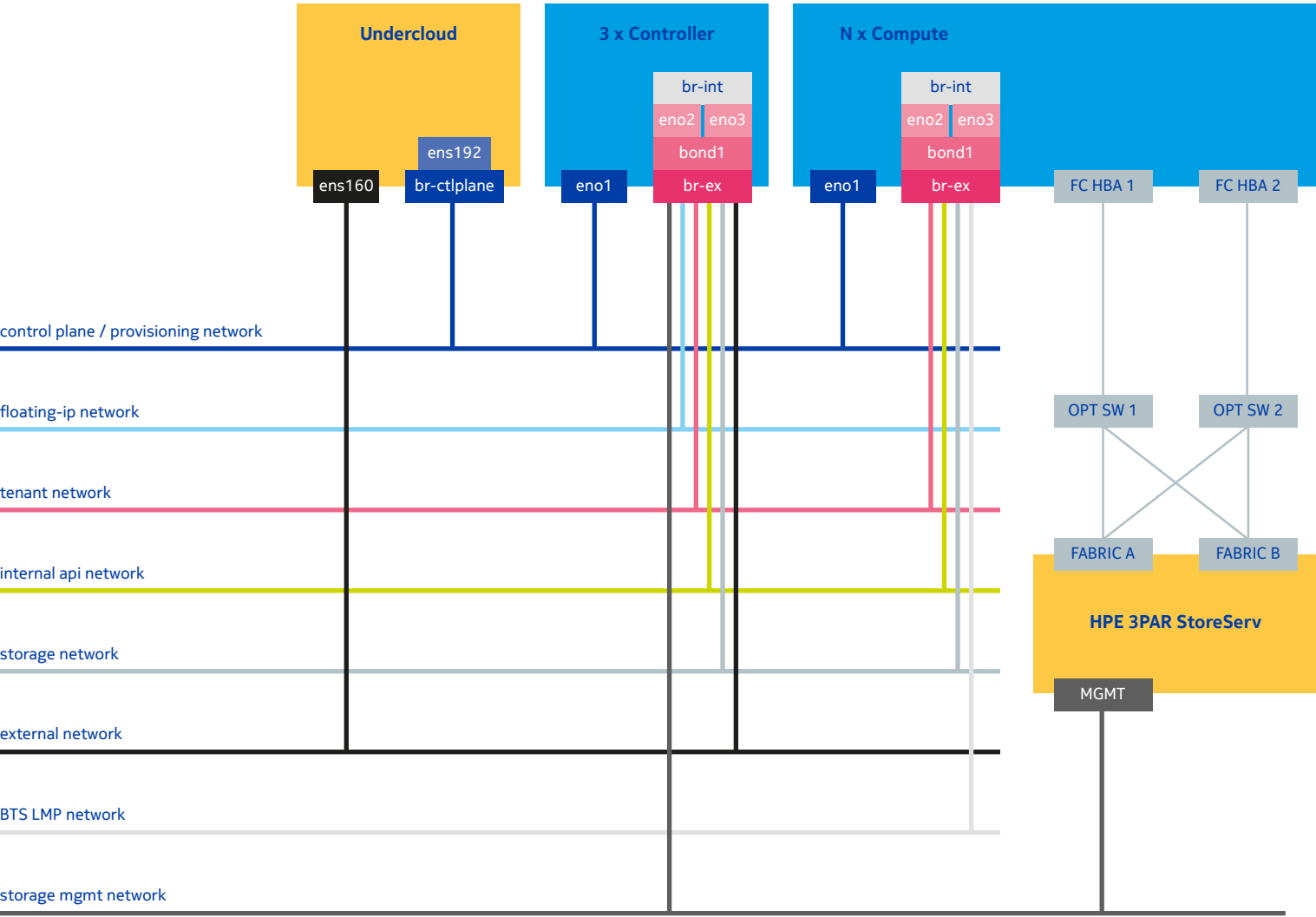
OpenStack core service networks are separated from each other using VLAN isolation to make the network diagnosis easier and faster in case of any network functionality or performance issues (i.e.: network bottlenecks).

4. Our Business is Our Data

Cinder is another OpenStack service, if not the most important one, that requires redundancy and backup policy, since it’s responsible for block storage – a place where all our businesses data is kept. Our business is our data, stored on our hard disks, that’s why mass storage has become a critical component of Nokia infrastructure defining our existence on a Telecommunication and IT global markets. OpenStack Cinder volume service supports lots of backend technologies starting from Ceph, GlusterFS or NFS, ending up on commercial third party solutions like HPE 3PAR, Dell EMC VNX or Netapp storage.

In 5G laboratory we have implemented OpenStack along with HPE 3PAR StoreServ storage as a backend for Cinder service. 3PAR is a redundant, effortless and affordable enterprise storage matrix, easy to deploy, manage and maintain. Its multi-node-capable storage architecture provides a great scalability, meeting most demanding application requirements of dynamically changing modern data centers. And what is the most important – HPE 3PAR StoreServ is OpenStack ready. It comes out-of-the-box with HPE 3PAR FC Driver based on python-3parclient, which is a part of the Python standard library. Using the driver, HPE 3PAR integration with OpenStack is quick, painless and requires a minimum effort in its implementation. Each OpenStack Compute node is connected to 3PAR

Figure 1 OpenStack architecture used in 5G infrastructure laboratory team



StoreServ using two FC (Fiber Channel) Host Bus Adapter cards creating two redundant FC paths, routed by independent optical switches to 3PAR independent Fabric ports. This solution provides „light speed” – secure and eavesdropping resistant communication between nodes and storage, which is required by Instances running in the cloud, in order to work quickly and efficiently. Due to a third party storage integration with the cloud, OpenStack gives us live migration possibility of our Instances between the OpenStack nodes during potential infrastructure reorganization, without downtime of the migrated Instances. This wouldn’t be possible without external storage attached to the cloud. To sum up, cloud integration with HPE 3PAR StoreServ provides data safety on the redundant storage level, as well as the communication level between those two end-points.

5. Affordable Scalability and Flexibility

Working with OpenStack cloud, we have learned that this technology, although still considered as a relatively new one, consists of reliable and mature components like KVM or MariaDB, and gives us enormous flexibility and scalability every company can afford. OpenStack deserves a special attention and its implementation should be taken into consideration, every time the cost-effective, easily scalable virtualized hardware is about to supersede obsolete and expensive server-based concept of infrastructure in laboratory or datacenter. OpenStack has become a serious player on the global IT market, and it makes it attractive and competitive against other cloud technologies and solutions.

References

- [1] OpenStack Official Website: www.openstack.org
- [2] RDO Project: <https://www.rdoproject.org/>

About the author

I work as a system engineer at the Nokia 5G Infrastructure Laboratory in Wroclaw. I am a Linux and OpenStack enthusiast, founder of the technical blog www.tuxfixer.com about Linux, virtualization, and cloud computing.

Grzegorz Juszcak
Technical Leader, Lab System
MN CDS Lab Operations

Passive Intermodulation Cancellation

Wojciech Zmyślony
Software Architect
MN SR



1. Passive Intermodulation

Passive Intermodulation (PIM) is a phenomenon present in passive elements of RF systems. It results generation of new frequencies at the system output due to non-linearities in the signal path when two or more carriers are transmitted.

Non-linearity means that the gain of the RF component is not constant over amplitude, and, for instance, negative amplitudes are amplified or attenuated more than positive ones. This results in change of the output signal shape, as well as change of the signal spectrum.

New frequency components present in the spectrum are called intermodulation products (IM). Depending on the order of intermodulation, its frequency can be calculated based on the frequencies of the transmitted carriers. There is a rule of thumb for calculating frequencies of IM products for two continuous wave (CW) carriers. When frequencies of these carriers are f_1 and f_2 ($f_2 > f_1$, $\Delta f = f_2 - f_1$), third order intermodulation products (IM3) will be located at $f_1 - \Delta f$ and $f_2 + \Delta f$; fifth order intermodulation products (IM5) will be at $f_1 - 2\Delta f$ and $f_2 + 2\Delta f$ (see [Figure 1](#)).

Formula for calculating intermodulation product frequencies for more than two transmitted carriers is more complex.

Since every carrier of bandwidth > 0 MHz can be treated as a composition of multiple CW carriers, intermodulation is present also for single carrier wideband transmissions.

Passive intermodulation can be caused, among others, by the presence of ferri- or ferromagnetic elements or alloys in RF path, such as (see [1], [2]):

- Iron
- Nickel
- Cobalt

Moreover, quality of metal-to-metal contacts between RF connectors is critical. Presence of the following conditions can cause PIM:

- Rust
- Corrosion
- Dust
- Metal flakes
- Loose connections due to too small torque

Examples of linear materials that do not produce PIM are:

- Copper
- Aluminum
- Silver
- Air

PIM is especially unwanted in Frequency Division Duplex (FDD) mobile network systems (like GSM, WCDMA or LTE) featuring multicarrier and/or wideband transmissions when a single antenna is used both by the receiver (uplink) and the transmitter (downlink). This is because the transmitter can, depending on frequency allocation, generate PIM which will overlap receiver carriers (see [Figure 2](#)). Depending of the PIM source strength, presence of PIM might cause uplink receiver desensitization by many decibels (dB).

The mechanism of deteriorating uplink transmission by PIM is as follows:

Figure 1 Third and fifth order intermodulation products (IM3, IM5) for two CW carriers.

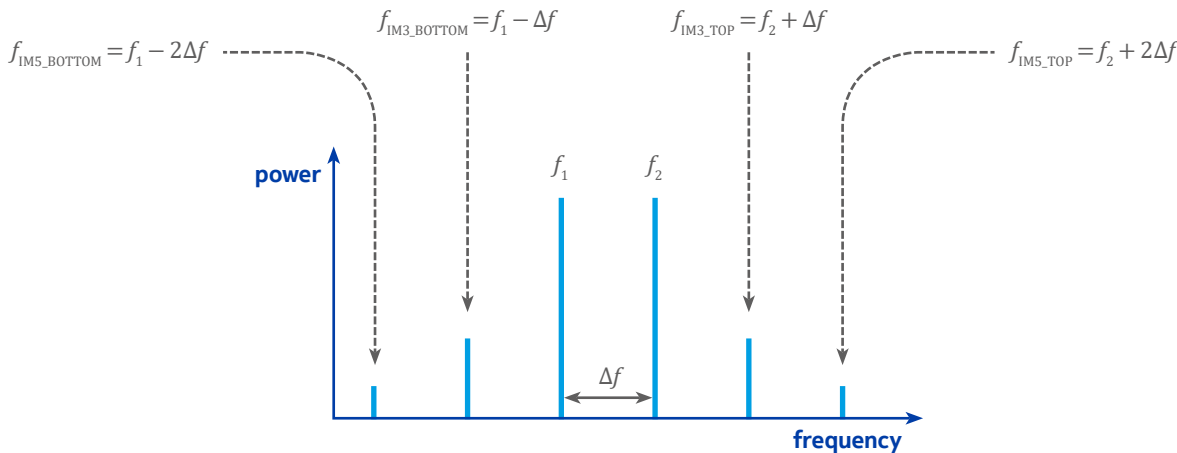
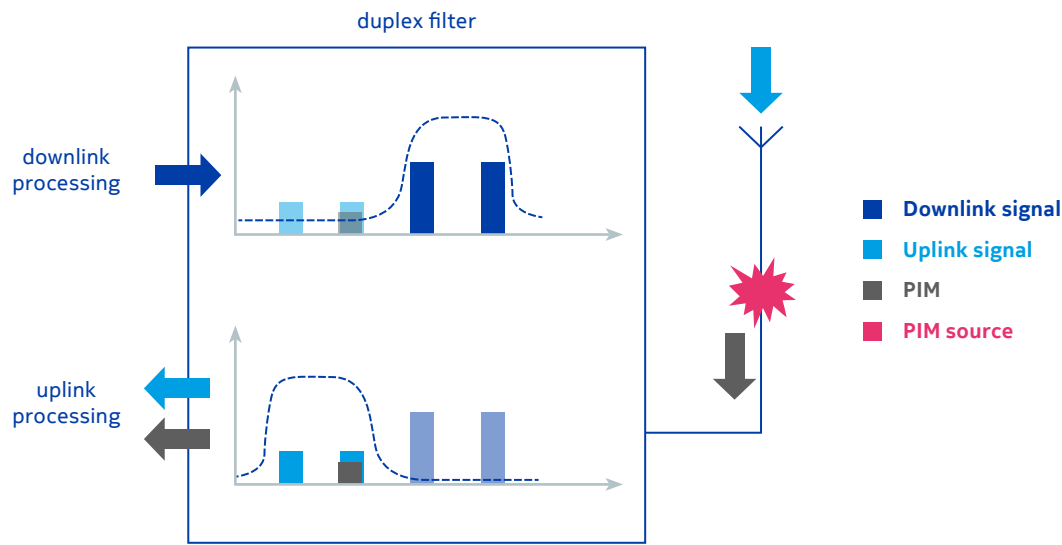


Figure 2 Block diagram combined with frequency plot demonstrating how PIM generated from downlink carriers in the antenna line can overlap uplink carrier.



- Rise of noise in receiver due to PIM
- Drop of receiver sensitivity
- Errors in decoding of LTE symbols by base transceiver station (BTS)
- Drop of uplink throughput

Intermodulation products problematic in the context of mobile networks are IM3, IM5 and partly IM7. Even order intermodulation products fall far away both from the transmit and receive bands. On the other hand, odd intermodulation products greater than the seventh order are negligible as their power is usually below the noise floor of the receiver.

In order to mitigate negative impact of PIM on mobile networks only specially manufactured and rated low-PIM passive RF devices should be installed on antenna lines of BTS. The level of PIM in continuously monitored and, if needed, antenna line is repaired and passive devices (connectors, adapters, splitters, etc.) are replaced. This generates additional operational costs for mobile network operators.

Another approach is to use carrier configurations which do not generate PIM overlapping uplink carriers. This, however, limits the possible use cases of flexible equipment delivered to operator.

2. PIM Cancellation

To mitigate the negative impact of PIM and allow mobile network operators to keep on using antenna lines which normally should be

repaired, a special solution was designed. It is called Passive Inter-modulation Cancellation (PIM Cancellation; PIMC).

PIMC is a mixed hardware and software solution deployed in Nokia radio modules. It allows to clean the uplink signal by removing the PIM in digital domain without any modifications of useful uplink signal.

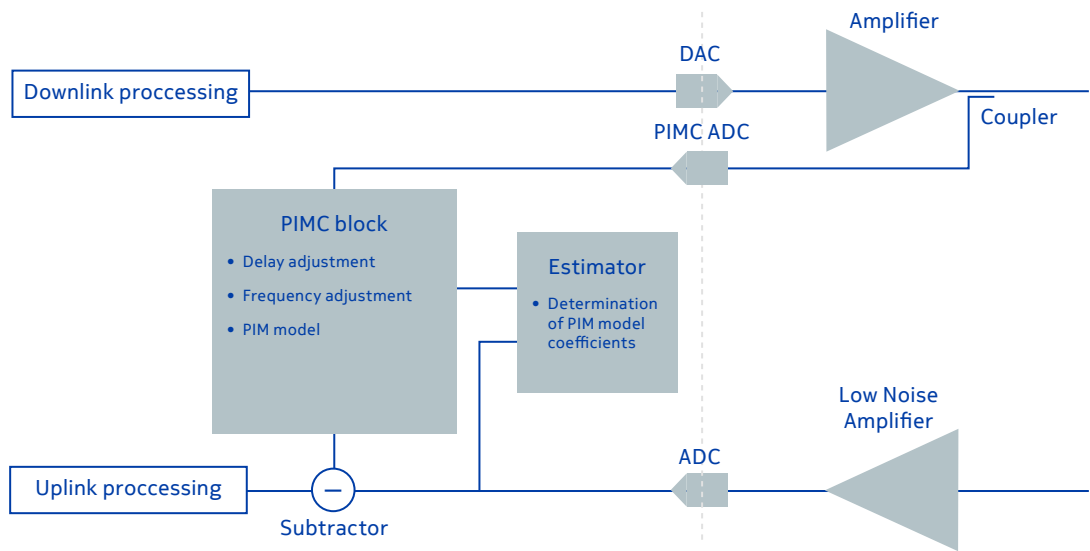
The idea behind PIMC is straightforward:

- To generate PIM model signal based on the downlink signal
- To subtract the PIM model signal from the uplink signal

Simplified block diagram of the PIMC gives overview of its position in the radio module (see **Figure 3**). Downlink signal is obtained by coupler at power amplifier output and sampled to the digital domain by PIMC Analog-Digital Converter (ADC). The signal is then processed by delaying it in the time domain and by shifting it in frequency.

Delaying is needed to compensate the time difference between the PIM present in the uplink signal and the original downlink signal which generates the PIM in antenna line. These signals enter the PIMC block nonsimultaneously. Downlink signal is sampled directly after power amplifier, within radio module, with very small latency. Real PIM present in the uplink signal has latency equal roughly to the double propagation time of the downlink signal from power amplifier to the strongest PIM source. This latency must be determined dynamically for the strongest PIM source and can be done by delay

Figure 3 Simplified block diagram of PIMC architecture.



correlation approach (see [3]). Some hardware specific offsets (signal processing time, propagation time on analog path between coupler and ADC, etc.) must be taken into account as well.

Shifting in frequency is needed to adjust the downlink and the uplink frequency plans, which must differ for FDD systems.

Downlink signal, after abovementioned conditioning, is transformed according to the given equation:

$$pim_n = c_1 f_1(tx_n) + c_2 f_2(tx_n) + \dots + c_K f_K(tx_n)$$

where:

pim_n is n^{th} sample of the PIM model
 tx_n is n^{th} sample of the downlink signal (delayed and shifted in frequency)
 f_1, f_2, \dots, f_K is set of base functions used for PIM modelling
 c_1, c_2, \dots, c_K is set of coefficients responsible for weighting impact of particular base functions to entire model

Values of coefficients need to be calculated before subtraction starts. Estimator block is responsible for this task.

The estimator calculates a set of coefficients which minimizes the difference between the PIM model signal and the uplink signal. The difference, marked as ϱ , is a function of coefficients and can be written as:

$$\varrho_n = (c_1, c_2, \dots, c_K) = rx_n - (c_1 f_1(tx_n) + c_2 f_2(tx_n) + \dots + c_K f_K(tx_n))$$

We are interested in the cumulative absolute difference over a larger batch of signal rather than in the difference of a specific (n^{th}) sample. Thus, this equation can be rephrased further. Additionally, let's minimize the function using the well-known least squares approach (see [4]):

$$\begin{aligned} \varrho_n(c_1, c_2, \dots, c_K) \\ = \sum_{n=1}^N \left(rx_n - (c_1 f_1(tx_n) + c_2 f_2(tx_n) + \dots + c_K f_K(tx_n)) \right)^2 \end{aligned}$$

Since the above equation is in fact a sum of square functions, the solution of:

$$\min(\varrho_n(c_1, c_2, \dots, c_K))$$

corresponds to the system of equations solution:

$$\begin{cases} \frac{\partial \varrho}{\partial c_1} = 0 \\ \frac{\partial \varrho}{\partial c_2} = 0 \\ \dots \\ \frac{\partial \varrho}{\partial c_K} = 0 \end{cases}$$

After further transformations, we obtain the following system of equations written as a matrix equation:

$$\begin{bmatrix} \sum_{n=1}^N f_1^2(tx_n) & \sum_{n=1}^N f_1(tx_n)f_2(tx_n) & \dots & \sum_{n=1}^N f_1(tx_n)f_K(tx_n) \\ \sum_{n=1}^N f_1(tx_n)f_2(tx_n) & \sum_{n=1}^N f_2^2(tx_n) & \dots & \sum_{n=1}^N f_2(tx_n)f_K(tx_n) \\ \dots & \dots & \dots & \dots \\ \sum_{n=1}^N f_1(tx_n)f_K(tx_n) & \sum_{n=1}^N f_2(tx_n)f_K(tx_n) & \dots & \sum_{n=1}^N f_K^2(tx_n) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_K \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^N rx_nf_1(tx_n) \\ \sum_{n=1}^N rx_nf_2(tx_n) \\ \dots \\ \sum_{n=1}^N rx_nf_K(tx_n) \end{bmatrix}$$

Its solution gives us a set (vector) of coefficients which can be used in PIMC block. PIM model generated based on the downlink signal will match real PIM present in the uplink signal and can be subtracted from it to remove the PIM.

Since downlink and uplink signals are not correlated (except for the presence of PIM), it is extremely unlikely to remove any useful components of the uplink signal (user equipment transmission).

3. PIMC performance

Verification of PIMC performance can be done by power measurements of uplink carriers affected by PIM. It is expected that the power of an uplink carrier will drop after enabling PIMC, if PIM is present. This drop, measured in dB, can be understood as PIM performance.

If PIM is very low, the power level in the uplink carrier shall not change.

Example shows such verification for the uplink LTE5 carrier (see **Figure 4**). It can be seen that the power measurement is lowest when downlink carriers are disabled.

When downlink carriers are enabled, the power rises significantly, by approximately 20 dB.

After enabling PIMC, the power drops back almost to the initial level, mitigating negative effects of PIM in the antenna line on the uplink transmission.

PIM performance depends strongly on:

- Strength of PIM source
- Number of PIM sources in antenna line

Technical solutions of PIMC handle the abovementioned aspects by:

- Complexity of PIM model (number of functions)
- Number of PIMC engines (to handle multiple PIM sources)

Experiments show that PIMC architecture of reasonable complexity allows to remove up to 20–25 dB of PIM. Antenna lines with greater PIM should be considered as damaged with a need of repair.

4. Summary

PIM Cancellation is an adaptive algorithm which can be applied in radio modules of BTS. Its purpose is to mitigate negative effects of antenna lines which do not meet low-PIM requirements. It requires dedicated hardware and software support within radio module.

Depending on the implementation, expected reduction of PIM can reach up to more than 20 dB.

Figure 4 Spectrum of the uplink LTE5 carrier demonstrating impact of PIMC on carrier power.



References

[1] S. Hienonen, Studies on microwave antennas: passive intermodulation distortion in antenna structures and design of microstrip antenna elements, Espoo: Helsinki University of Technology, 2012.

[2] A. Kretz, Passive intermodulation (PIM). White paper, Huber & Suhner, 2012.

[3] W. Zmyślony, Optical Interface Baseband Signal to RF Signal Delay Measurements for Radio Modules. The Future of Telecommunication is Happening Now. Check how the experts do it., Wrocław: Nokia, 2016.

[4] P. Ch. Hansen, V. Pereyra, G. Scherer, Least Squares Data Fitting with Applications, Baltimore: The Johns Hopkins University Press, 2013.

About the author

I joined the company – then Nokia Siemens Networks – in July 2010, just after graduating from Wrocław University of Technology. Initially I worked at the Hardware Baseband department as an integration engineer. In 2013 I moved to the RF Software department (now Smart Radio Software), as a member of DAPD team, which is responsible for implementation of various DSP algorithms for radio modules. Since 2017 I am a Software Architect of PIMC team.

Wojciech Zmyślony
Software Architect
MN SR

Virtual Environment Manager Cloud solution. How to create fully automated R&D laboratory

Łukasz Szewczyk
Specialist, Lab System
MN CDS Lab Operations

Bartłomiej Radwan
Specialist, Lab Network
MN CDS Lab Operations



1. Requirements for a modern R&D laboratory

The requirements for a modern R&D laboratory are evolving with time. Nowadays, the trend and the challenge is to create a fully automated, “self-service” laboratory that only requires manual work in the process of building and setting-up the laboratory and ideally minimal physical and manual intervention afterwards. Flexibility, cost-reduction (Engineers’ time), and ease of use are becoming more and more important. Thanks to the development of technologies like virtualization and Cloud computing that are being widely used at Nokia, we can meet the requirements that are being put on modern R&D laboratories. There are different Cloud platform solutions like OpenStack or VMware vCloud Director with NSX controller but in the laboratories where real hardware (with embedded systems) is the subject of measurement and testing scenarios, we face additional requirements. One of the most important requirements is that software on Virtual Machines has to connect directly to the hardware under testing. It means we need a Cloud solution with virtual-to-physical connection set up that will work in an multi-vendor environment and will be compatible with all kinds of hardware in the network. Because none of the available Cloud platforms has met our requirements, we have developed our own solution – Nokia VEM (Virtual Environment Manager).

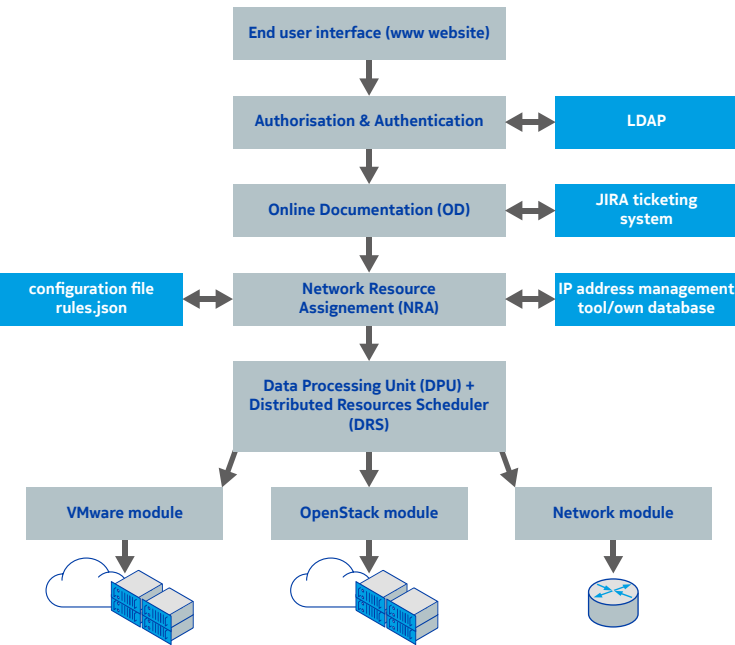
2. The scheme and logic of a self-service R&D laboratory system

The basic principle of the design is that the product has to have a modular architecture. When a new kind of hardware (for example, from a different vendor) will be added to the existing environment, there will be no need for a change in the main part of the program, only a need to create a small service module for the new piece of hardware.

In **Figure 1**, we can see a block schema of the solution. The first step is to provide an interface (webpage – GUI) to the customer where, after authorization, the user can choose: the type of a test environment, the template for VMs and hardware location, and then request to build a Test Line with just one click. The webpage is partially written in PHP, which provides features for authentication and authorization, like sessions, cookies, and integration with company LDAP. The next step is to document the actions in a ticketing tool like JIRA.

For the past few years, users have been using JIRA as the main tool to place a request to the lab team, for example, to assign a new IP address, to create a virtual machine, debug a network problem, etc. Tickets are the basic source of knowledge for the person who is resolving the issue and also for the person who has created the ticket. Each task should contain a detailed description of the issue and its solution, which can result in shortening the time for solving similar cases in the future. Furthermore, a ticketing system is a huge database, it

Figure 1 The scheme of Virtual Environment Management



collects information about all modifications which have been made in the Data Center infrastructure. For instance, if a user has requested a reconfiguration of a network device, which caused outage of a small part of the network, it is possible to revert to the previous state, because changes should be included in the ticket. An online documentation module has been created to track all changes and requests in the JIRA database, to inform users about the states of their tickets, and to insert more detailed information in comments. The module works in two phases, in the first one it creates a ticket with data which have been entered by the user via the web portal and assigns the proper laboratory team member who will debug the issue in case of failure. VEM, through an API, sends the appropriate structure, which contains necessary fields and their values (**Listing 1**).

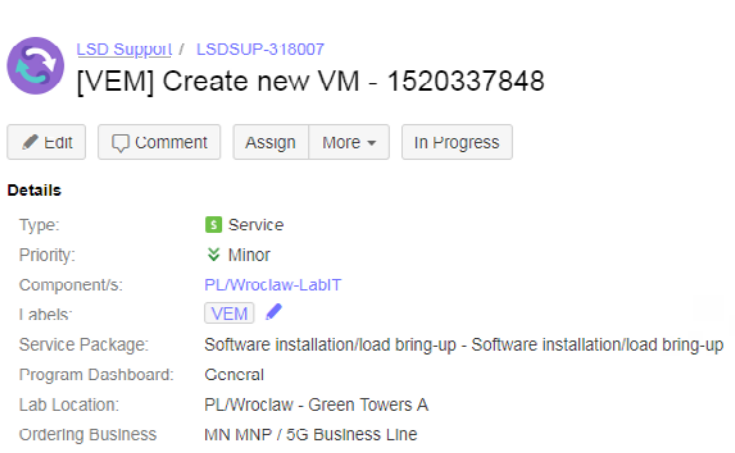
Values which are inserted into a request depend on the type of action which is sent to the server. The server will respond with a unique Ticket ID, which can be used later (**Figure 2**). The second phase is activated when the demanded action has been completed (for example, a VM has been created). If the task is successful, the module will request transition from the “in progress” to the “resolved” state in JIRA using the unique Ticket ID received in phase one. Additionally, if additional information is needed, it will be added in comments (for example, an IP address of a management interface of a new VM). On the other hand, if the task has failed, Online Documentation (OD) will insert an annotation about the problem. In such case, the user should directly contact the person who has been assigned to this specific task in phase one (the Field Assignee in JIRA).

Listing 1 Example of a structure send via API

```
issue_dict = {
  //Project name
  'project': {'key': 'LSDSUP'},
  //Issue type
  'issuetype': {'name': 'Service'},
  //Issue summary description
  'summary': '[VEM] Create new VM - 1520337848',
  //Full description of the issue
  'description': 'Please create VM with parameters...',
  //Issue priority
  'priority': {'name': 'Minor'},
  //Service Package
  'customfield_15700': {'id': '26314', 'child': {'id': '26315'}},
  //Laboratory location
  'customfield_15131': {'value': 'PL/Wroclaw', 'child': {'value': 'Green Towers A'}},
  //Ordering business
  'customfield_15132': {'id': '32937'},
  // Components
  'components': [{'id': '32937'}],
  //Label for statistics
  'labels': ["VEM"]
}
```

Next important module which has been created especially for VEM is Network Resource Assignment (NRA). It is used to directly access the database with IP addresses (IP Address Manager from SolarWinds, for more information, see “Network tools – a whim or a must? Overview of network tools used in the R&D labs.”). The NRA module’s main functions are: searching for available resources and reserving and releasing network addresses. By using Python library pymssql, the module logs into the database and performs dedicated SQL queries. NRA is started right after OD, it loads information about the mapping team (previously chosen in a web browser) and its resources. Else, if there is no direct mapping, a shared pool will be used. Such data is stored in a JSON file, which contains several sections that are divided into pools, which include resources (**Listing 2**). Each section means a physical location (e.g. “gta” means the Green Tower A building in Wrocław), a pool represents specific teams (e.g. “PL-WRO-SiSo”) or shared values. It is possible that a single team is located in several buildings, so it has to be defined in multiple sections. At the bottom, there are two types of resources: “ip” – dedicated networks or a range of IP addresses, “range” – the dedicated range of unique vlan ids per location (crucial connection to eNB in the R&D environment). Moreover, it is possible to limit “ip” ranges by using dedicated flags: “I”, which means “include” (take addresses only from the given range) and “E”, which means “exclude” (don’t take addresses from the given range) (**Listing 2**).

Figure 2 Example of an issue created in JIRA via API using the structure from **Listing 1**.



Listing 2 Example of a single section from a mapping file

```
//Section
“gta”: [
  {
    //Pool
    “common”: [
      {
        //Resource
        “ip”: “10.42.134.0/23, 10.42.82.0/23:E:10.42.83.1-10.42.83.99”,
        “range”: “700-759, 761-800, 1400-1599, 1701-1799, 1900-2000”
      }
    ],
    //Pool
    “PL-WRO-SiSo”: [
      {
        //Resource
        “ip”: “10.42.82.0/23:I:10.42.83.1-10.42.83.99”,
        “range”: “1300-1399”
      }
    ]
  }
]
```

The distributed resources scheduler is the brain of the application. This unit has full overview of physical resources in a laboratory infrastructure, like servers, switches, routers, information about routing and where VLANs are terminated, information about hardware utilization, like current CPU and memory load on available servers, and available free storage space. Based on this information, VEM can decide where to create a new virtual machine to optimize the CPU, memory, and storage usage, but also decide how to connect it to the hardware under test so that the connection will have minimal latency.

The final step is to create a VM and configure the network according to the requirements above. Here comes the great benefit of VEM – modular architecture. Nokia laboratory is multi-vendor environment where devices can be delivered by Cisco, Dell or our own company. Moreover servers can run different hypervisors, like VMware ESX or KVM in one datacenter, and OpenStack in another - but in VEM, logic (all steps above) and deployment are completely independent. There is one module that drives deployment of a new VM in the VMware environment, one module for OpenStack, one module for Cisco, and one for Nokia. Each module works similarly to a driver in a computer – it translates logic from VEM into CLI commands and

API calls that each hardware vendor and hypervisor can understand. Thanks to this approach, customers can expect the same way the test environment works regardless of the underlying hardware.

3. Benefits of using VEM

Since its deployment in September 2017, VEM has performed over 3700 tasks, including building new Test Lines, creating new Virtual Machines, and re-installing existing Virtual Machines with the option to preserve all VM configuration and customer data. Everything was done fully automatically the with average time of accomplishment under 45 minutes. This means huge savings in laboratory maintenance time. Laboratory Engineers do not need to spend time on non-value added repetitive tasks anymore. However, the most important improvement was achieved on the service level – now end users can just log in to the portal and with a few clicks request a whole new Test Line that will be delivered to them in less than 45 minutes. And due to VEM architecture; independent logic and deployment modules, customers can expect the same, unified Test Line configuration regardless of the hardware used in their Laboratory.

About the author

Łukasz Szewczyk is a graduate of Electronics and Telecommunication at Wrocław University of Technology. Later worked as a Virtualization Engineer for VMware Inc. He is currently working in Common Development Services division at Nokia Wrocław.

Łukasz Szewczyk

Specialist, Lab System
MN CDS Lab Operations

About the author

Bartłomiej Radwan is currently an R&D Laboratory Engineer in Common Development Services in Wrocław. He joined Nokia 7 years ago as a test automation developer and moved to CDS in 2015 to expand his passion, which is networking. He is currently focused on development of tools and management of laboratory network.

Bartłomiej Radwan

Specialist, Lab Network
MN CDS Lab Operations

Datacenter network architecture and migration towards modern design

Łukasz Michna
Technical Leader, Lab Network
MN CDS Lab Operations

Grzegorz Dygoń
Specialist, Lab Network
MN CDS Lab Operations

Stanisław Pospiech
Solution Architect
MN 5G&SC



1. Introduction

Is your data center ready for the upcoming trends in the modern IT world? What are the trends actually and what makes them evolve? For about a decade, two major challenges are observed: The time needed and the cost associated with the delivery of user-facing services seem to be the most significant factors, which directly influence the customer’s perceived quality of service. What is the solution to provide a platform which assures instant service delivery, on-demand scalability at a reasonable price in a pay-per-use model? How data center is perceived needs to be considered from a new angle, where infrastructure is not limited by the space and is extend above and beyond the wall. This is how Cloud infrastructure becomes fundamental for an organization to achieve flexibility with a scale-out architectural design to deploy services rapidly, with a low cost of entry. But the question is whether the DC network is prepared for the mentioned changes towards cloud-enabled platforms that may affect the network patterns which rise another question - Is a DC network prepared for that? We will focus on this question in the upcoming chapters.

2. Design techniques and multi-tier approach

2.1. Hierarchical network design

Data Center Network blueprint should take two major principles in mind: modularity and scalability. Following those design targets allows a network architect to overcome challenges like rapid traffic growth or changes in traffic profile, balancing application workload or deploying new network gear in case of an HW scale-up. It also brings profits on management and troubleshooting layer and opens an opportunity to improve robustness by implementing redundancy mechanisms.

The most common way to keep modularity and scalability in place is building a network based on hierarchical layers, where each set of network devices have a particular role to serve.

Starting from the bottom, the access layer, where switches are placed closest to the server socket, provides them sufficient density of access ports. Then through the distribution layer placed in the middle, where access switches are aggregated and grouped into blocks and represented, for example as server rows, server rooms, whole data center floors or buildings. Blocks with aggregation switches on top are easily to scale-up in case of growth demands, and may be grouped and interconnected together through the core layer switches.

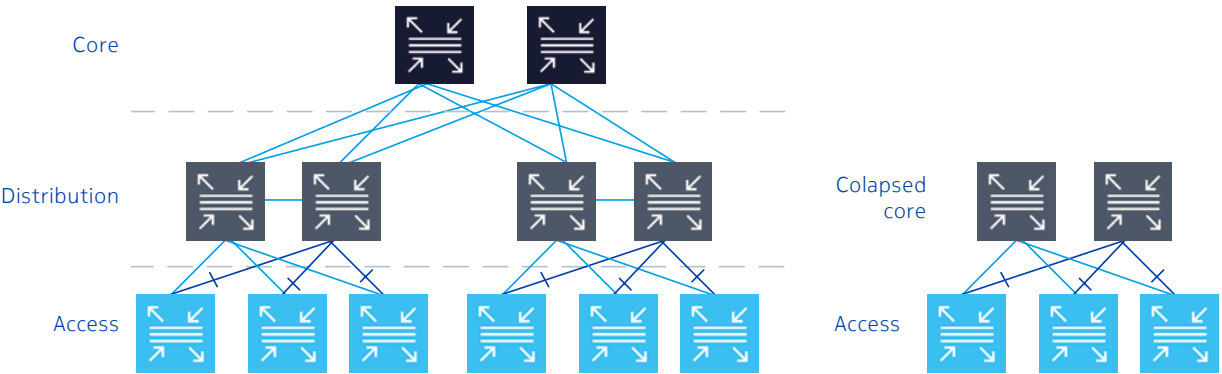
Each layer has its own unique role and capabilities.

Role of **access layer** devices is to provide high port density together with keeping the lowest cost per switch port. Moreover, keeping user traffic in right network domains (VLANs), taking care about size of broadcast domain and providing separation. Sometimes these basic functions may be extended to security policy for network end hosts like MAC address filtering or traffic and protocol filtering.

Distribution layer switch provides high layer 3 throughput for packet handling and aggregates multiple access layer devices. Also, security and policy-based functions are implemented there like access list and packet filters.

Network devices used in data center **core layer** should provide very high throughput at layer 3 and should be capable of switching the traffic as efficiently as possible, and without a cost. All packet manipulations like filtering and access list solving should be done in access or distribution layer and let the core provide excellent performance on network edge.

Figure 1 Hierarchical network design (2-tier and 3-tier approach)



When it comes to the network redundancy, it may be implemented on every level through multiple uplinks, or adding second, redundant unit.

2.2. 2-tier vs 3-Tier design

In some cases, not all layers are needed. This situation occurs quite often in small data centers or projects where the surface area is well known and limited in advance. A 3-tier architecture in this case is an unnecessary over-design. In that event a core and distribution functions are served by the same device(s). In such scenario, a duplicated distribution-core providing redundancy failover is a good resolution. This approach lets an administrator to secure access layer switches via two redundant uplinks access-to-core where each of them is terminated to a separated neighbor in an upper layer. Although the mentioned technique perfectly fits a small environment and appears to bring savings, it causes growth limits and makes many difficulties while integrating with an another collapsed block and a need for layer 3 links in-between rise rapidly.

2.3. Top of Rack and End of Row

Data centers characterize themselves by increased density of server racks. Typically, they are in multiple rows, split by hot and cold aisle, and fully occupied by servers. Connectivity between servers and network access layer can be organized two ways.

End of Row (EoR) design is a physical topology in which all servers in a row are connected to the one central access switch installed in the last rack. This requires using a switch with sufficient number of ports to provide connectivity to every single server port. EoR switches exist in a modular form, with many line cards or port extenders.

Top of The Rack (ToR) is a design where access switches reside independently in every server rack, quite often on the top. Access ports and uplinks are provided with proper oversubscription ratio, which is a ratio between a sum of all access ports throughput capacity to all active uplink ports throughput.

It is common to set an inter-VLAN routing on an EoR or ToR switch or in an aggregation point, and splitting data center physical topology together with a logical network into corresponding blocks or points of delivery (PoD). This makes the topology more modular and activities like upgrades or changes do not affect the infrastructure but only the single rack or row. This is also a good moment to step into Spine-and-Leaf design when 2-tier / 3-tier may be transformed in full-mesh topology, and ToR switch becomes multihomed L2 or L3 Leaf (which is explained in detail in the following chapters).

Lastly, for a redundant design two switches should be placed, for the EoR on the opposite ends of a row to avoid mess in cabling and for the ToR in the middle or in the bottom of the rack for the same reason.

2.4. L2 / L3 boundary and Inter-VLAN routing

In date center network architecture, as in the traditional networks, crucial is the decision about the network layer nominated to perform the inter-VLAN routing. The choice concerns the distribution or the access layer.

The first option requires L3 connection to the core. This improves keeping all VLANS inside one distribution block providing STP topology always converged.

A design with L2-L3 boundary on access level is another option. VLANs are limited to the access switches, and on the distribution level all links are layer 3 type. This means no dependences on Spanning Tree Protocol (STP) and keeps broadcast domains far away from extending across the distribution. Fast convergence is offered between an access and an aggregation by the routing protocol rather than by the STP or Rapid-STP (RSTP) mechanism. What is crucial is that implementation of the boundary on the corresponding hardware, it needs to support layer 3 features. This is particularly important for the case with access layer example, where most of the switches from the vendors portfolio are pure layer 2 capable.

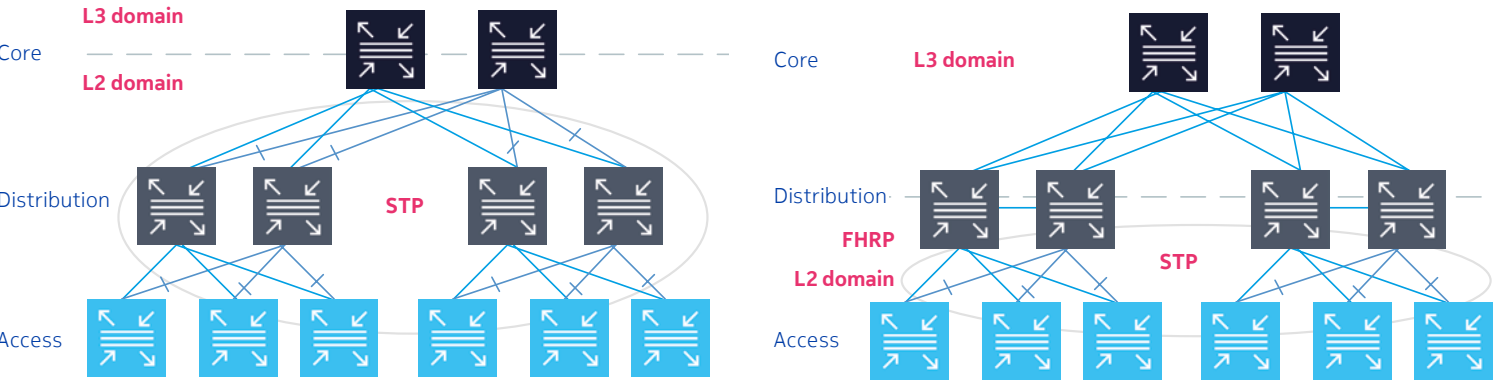
Decision which manner is more suitable depends strongly on the sort of services running in a data center. For instance, in a data center with many virtual machines running on different hosts and live migration feature enabled at the same time, large layer 2 domains are needed. This points to the inter-VLAN routing implemented on the distribution or even done in the core layer. However, design like this introduces also drawbacks, mostly because of the increased layer 2 flooding domain, and may affect the network performance.

3. Modern Data Centers trends

3.1. Limitations of traditional DC network layout.

Traditional data center network architecture based on the hierarchical concept that supports classic 3-tier design or 2-tier design have significant limitations that must be overcome to meet modern IT requirements. One of these limitations is a traffic pattern in a traditional flow that is based on oversubscription and blocking architecture, focused mainly on north-south communication, from internal users to the applications servers, residing in different subnets or point of delivery.

Figure 2 L2 and L3 Domain Boundary for DC Aggregation and Core.



Today's workloads and application requirements have been changed and traffic flow from server virtualization and cloud-based services became a combination of east-west and north-south communication. Mobility of virtual machines requires communication across one or multiple data center hosts. That is why modern data center

networks require more flexibility, high-speed nonblocking any-to-any connectivity, and low-latency architecture to support scalability for a new bandwidth requirement.

How can this be achieved? Eliminating or limiting dependences on traditional protocols such as Spanning-tree (STP) and First Hop Redundancy Protocol (FHRP) is a key to build a fast forwarding underlay network. STP protect network but built layer 2 loop free topology by blocking all redundant links. In combination with the FHRP converge time increases when the node or link fails.

Modern data center design is moving from classic architecture to a flatter fabric based approach to overcome all limitations described above. Network designer needs to select a proper data center architecture which will meet business demands and technology trends to offer data center networks a higher level of scalability, flexibility, and performance.

3.2. Modern design - what is a Spine-Leaf?

Clos architecture was created by Charles Clos from Bell Labs who proposed a mathematical model that offers a nonblocking performance in a switching array used in telephone networks. Switches were organized in multistage topology to achieve 1:1 (zero) oversubscription and any-to-any connectivity, which was the model needed for modern cloud-enabled data center.

IP Clos in Data Center Network provides a flatter architecture offered a 2-tier design. This model consists of spine and leaf switches,

Figure 3

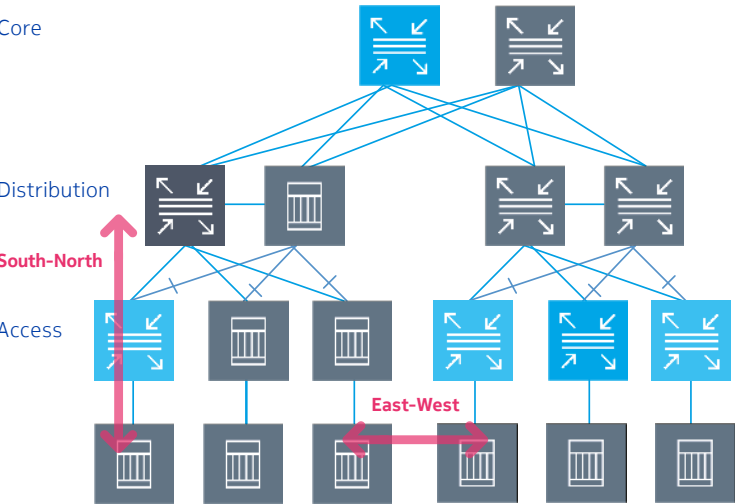
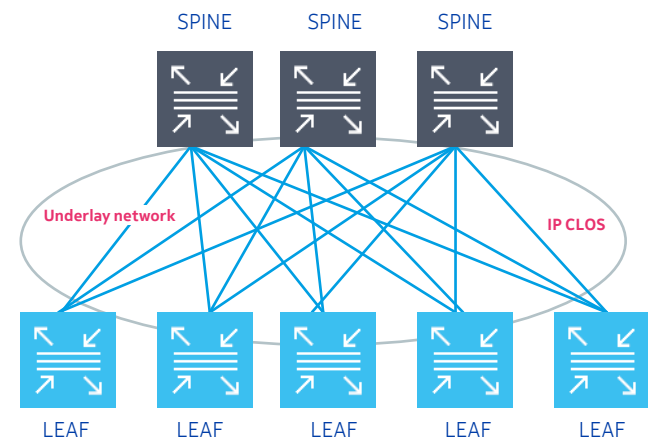


Figure 4 Spine-leaf network architecture



where each leaf switch is connected to every spine switch, based on an IP that provides Equal Cost Multipathing load balancing (ECMP) over all available layer 3 links. Network redundancy and a simplified design are the benefits of this approach, also performance can be scaled as needed by adding additional spine switch. Role of a spine switch is to forward the traffic between leaf switches, which provides direct connectivity to bare metal servers and hypervisor based servers. This approach keeps latency at a predictable level because there is only one hop (spine switch) on a traffic path between servers connected to different leaf switches.

Comparing to a traditional data center architecture where layer 3 gateway and other services, such as firewall and load balancers, reside on an aggregation layer in IP Clos boundary point are moved down to the leaf switches named border leaf. This allow IP Clos offer scalability and reliability for today's multitenant virtualized and cloud environment because is not limited to pair of aggregation switches.

IP Clos eliminates reliance on a Spanning Tree Protocol because it is based on layer 3 routing protocols to forward packets across fabric nodes. Top-of-rack switch in this model is a L2/L3 boundary point which is limiting layer 2 domain flooding to the one rack. Which routing protocol will be suitable for Layer-3 (underlay) network? If the environment does not require to extend layer 2 connectivity across a fabric (TOR leaf switches), then the external BGP can be chosen because it provides a more controlled and scalable IP Clos. Otherwise Interior Gateway Protocols (IGP) such as OSPF or IS-IS can be used to control plane routing and forwarding. In such case it is best to enable link state protocols to optimize its performance.

One of the requirement for a modern data center network and a cloud-based environment is to support virtual machines' mobility across different leaf switches. This can be achieved by an overlay transport network.

3.3. Network overlays to overcome STP issues

Network overlays are the representation of virtual networks to create additional layers of abstraction on underlying physical network offered by a IP CLOS fabric design. The basic idea is that all changes made on the virtual overlay networks do not require to changes the physical underlay network. What should be understood about overlay is building a layer 2 networks between the hypervisor hosts by means of creating layer 3 tunnels to allow virtual machines to communicate one with another in the same broadcast domain. Traffic is encapsulated into an overlay tunnel and transmitted across the underlying IP network until it reaches the destination hypervisor which firstly decapsulates traffic from a tunnel and then forwards to a proper VM. What about bare metal servers that do not handle virtual traffic? The TOR leaf switch can handle the overlay tunnel encapsulation and forward it on behalf of the non-virtualized servers by creating a direct overlay tunnel to the hypervisor.

Figure 5 Overlay tunnel terminated on virtual and physical switches

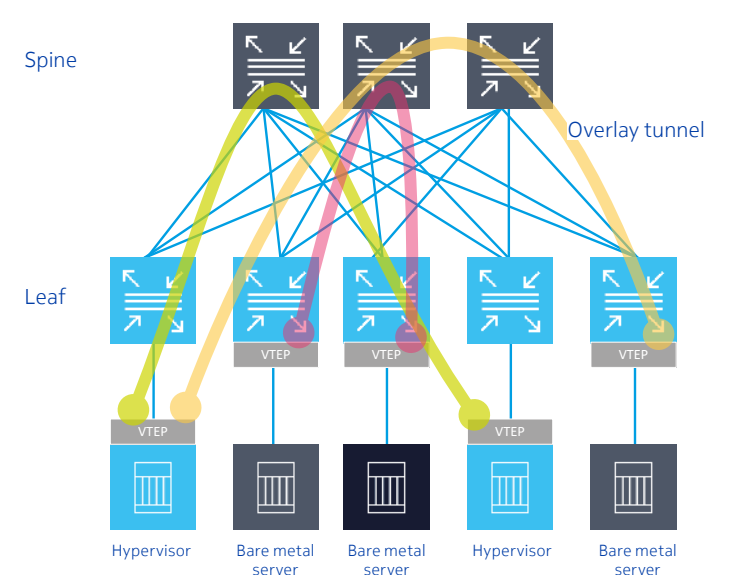
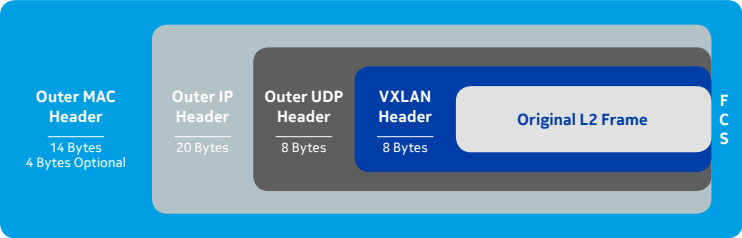


Figure 6 VxLAN packet encapsulation

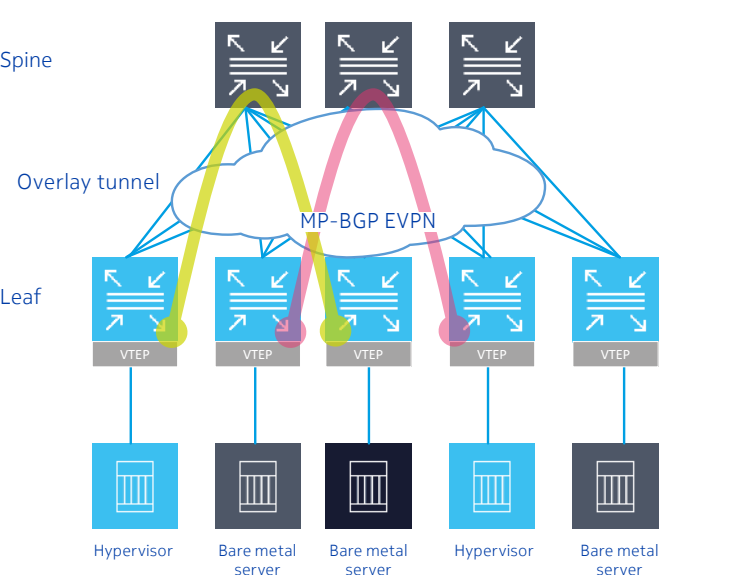


Transport protocol used to build an overlay network needs to offer a scalable and flexible Equal Cost Multipathing routing (ECMP) and listed below can be used as primary protocol supporting MAC-in-IP encapsulation:

- Network Virtualization using Generic Routing Encapsulation (NVGRE)
- Stateless Transport Tunneling (STT)
- Virtual Extensible LAN (VxLan)

VxLAN is the most common overlay protocol used in IP Clos and cloud-based environment, defined in RFC 7348, designed to provide the same Ethernet layer 2 network services as VLAN, but overcoming

Figure 7 VTEP termination on an HW level



the limitation of a traditional pool of 4094 VLANs. VxLAN uses a 24-bit segment ID known as the VxLAN network identifier (VNID), which enables up to 16 million VXLAN segments to coexist in the same administrative domain. VxLAN defines a MAC-in-UDP encapsulation scheme where the original layer 2 frame has a VxLAN header added and is then placed in a UDP-IP packet. Due to the encapsulation, VxLAN adding 50-byte overhead to the original frames and the maximum transmission unit in the underlay network needs to be increased by this value.

VxLAN uses VTEP (VxLAN tunnel endpoint) devices to map a tenant end device to a VxLAN segment, and to perform encapsulation and de-encapsulation of the traffic originating from a VM located on the hypervisor server. VTEPs could also be terminated on a physical switch or a physical server and could be implemented on a software or hardware level.

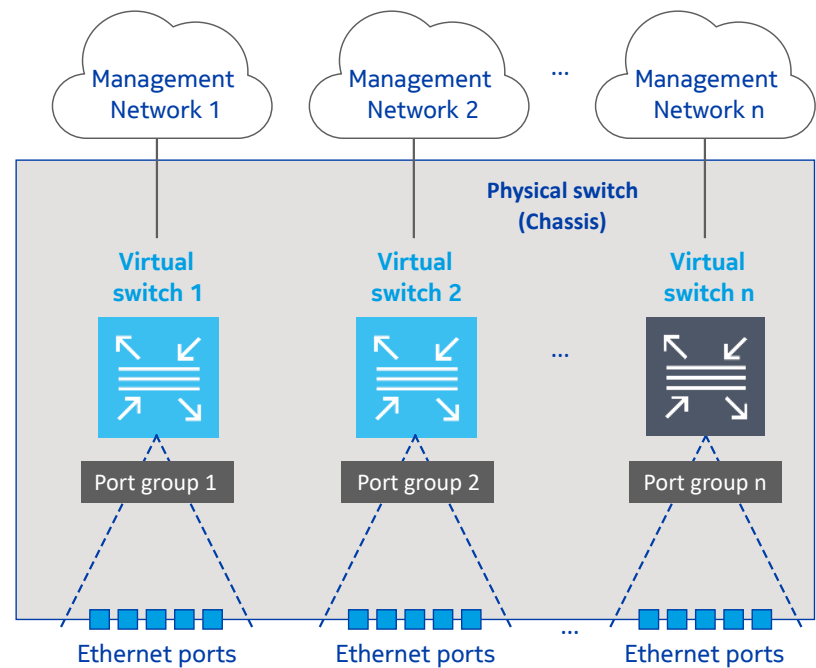
The question is how VTEPs are discovered and how traffic flow is passing packets on underlay network? In general, VxLAN can be configured in a flood-and-learn mode using a multicast group to discover other VTEPs, while also enabling MAC address learning. Broadcast, unknown unicast and multicast (BUM) traffic is forwarded to every VTEP in the network and the end result is to flood MAC address to all IP fabric port on particular segment and MAC address learning on data plane in the traditional way.

As an alternative to the flood-and-learn design is to use a control plane such as Multiprotocol Border Gateway Protocol Ethernet Virtual Private Network (MP-BGP EVPN). The MP-BGP EVPN introduces a set of features that reduce or eliminate traffic flooding in the overlay network and enable optimal forwarding for both west-east and south-north traffic by the use of MP-BGP extensions, such as rout target (RT) and route distinguishers (RD), to identify VTEP and VxLAN memberships. VTEPs are defined on leaf switches and the servers don't need to have information about VTEPs and MAC address learning. The benefit of the MP-BGP EVPN technology is that any VTEP located on a leaf-spine switch can be the distributed in an anycast gateway for end hosts in its IP subnet by supporting the same virtual gateway IP address and the virtual gateway MAC address.

3.4. Multitenant Data Center

Modern data centers share the pool of computing, storage and network resources among different business units. Common, shared infrastructure allows the administrator to optimize utilization and reduce costs. However, security and overutilization became risk register items. The solution may be a logical isolation of available resources, data flows and user traffic from each other and closing them inside separated tenants, where each tenant's data remains invisible to other tenants providing end-to-end isolation.

Figure 8 Network Device Virtualization



Multitenant data center ensures the network separation on multiple layers which presented below.

• **Network Device Virtualization**

In this example, implementation takes place on layer 1, on a physical network device, where a single equipment may be presented as many unique ones. Logical network units are virtualized and working on the top of common hardware (switch), expanding tenant separation on data and control plane but, also separating the management plane. Logical entity within the switch is working with its own unique configurations, network packet processing and may be managed by separated administrators.

• **Network layer 2 isolation**

Layer 2 isolation is a technique related with access or aggregation layer. Tenant isolation is fulfilled by means of VLANs, which close tenant traffics and network domains, and separates access ports.

• **Network layer 3 isolation**

Virtual routing and forwarding (VRF) behaves towards layer 3 similar to how VLAN behaves for layer 2. As a technology, VRF allows a router to hold many independent routing tables, best paths calculated by routing protocols and deal with overlapping IP addresses from multiple tenants. Tenant networks are linked with corresponding layer 3 interfaces which are examined by VRFs.

• **System-Level isolation**

Isolation is delivered on virtualization level, and provides separation between virtual machines (VM). Hypervisor assures user traffic separation, and in practice looks like layer 2 and layer 3 isolation, but is implemented inside the host on internal virtual switches. Tenant traffic may be tagged by VLAN, or VxLAN ID and becomes unseen to others.

4. Network automation and Step towards SDN

Automation as a process is usually considered as asking computers to execute repeatable actions in expected way, offloading humans. But, it is not only about speed, and agility. Automation provides to a network administrator predictable outcome which can be gathered and tailored in a database. Furthermore, engineer can reduce time spent on manual configuration and troubleshooting. Tools that helps gather information such as CPU usage, ports utilization, reachability, can create alarms, or even deploy new software or configuration on device looks to be mandatory in modern infrastructure. Ready-made tool sets available in a vendor's portfolio are mostly expensive, and vendor-specific. As an alternative there are also open source tools which often do not provide as much information as the proprietary ones but they can also be very helpful. Tools are constantly evolving because every day more and

more data are needed. A constantly expanding network infrastructure generates more complicated, harder to maintain, and manage environment.

Network automation cannot be treated as an “add-on” or just another project, but as a new parallel architecture to avoid as much one-off network changes. A well-designed API, and rich documentation can be a big step forward in the automation world. Avoiding vendor-proprietary features and extensions leads to a more simplified, repeatable, and easier to automate architecture.

4.1. User and network automation types. Zero Touch Provisioning

Device provisioning is one of the most important tasks that can be automated. The idea is to push previously prepared templates of initial configuration to a new device. But what if there are more vendors in our network? This means we'll end up with 2,3 or even more templates. To avoid that situation Ansible and Salt can be used to rapidly generate hundreds of configuration files.

Aside from device provisioning something that is also very important is **data collection**. Most tools use Simple Network Management Protocol (SNMP), but that is not always the best solution because information is provided by every polling cycle and not streamed when collected in the device. Specified command outputs could be more accurate than large data package sent by SNMP. Nowadays, a scripting language(for example Python) with a netmiko library delivers all the needed data.

Configuration Management like for example device provisioning involves deploying, pushing, and managing the configuration state of a device, but in a daily basis.

Compliance - making manual changes can be risky, which is why it is better to start with data collection, monitoring, and configuration building, which are all read-only and low-risk actions. One low-risk use case that uses the data being gathered is called configuration compliance checks. With configuration compliance checks, it is easier to verify whether something is true or false. Also, it is easy to start with one compliance check and then add more as needed.

Reporting, gathering data, and creating periodical or on demand reports is very useful to see how our network grows and think about the next step to improve our network.

Troubleshooting - running more commands on each device can be very time-consuming. Automation can speed troubleshooting especially if there are more devices to check. It can also be used to run more complicated sets of commands so the user does not have to remember all of them.

Migration from one platform to another requires good knowledge of both vendors, but can be easily automated by commands translation, and reconstruction of the syntax.

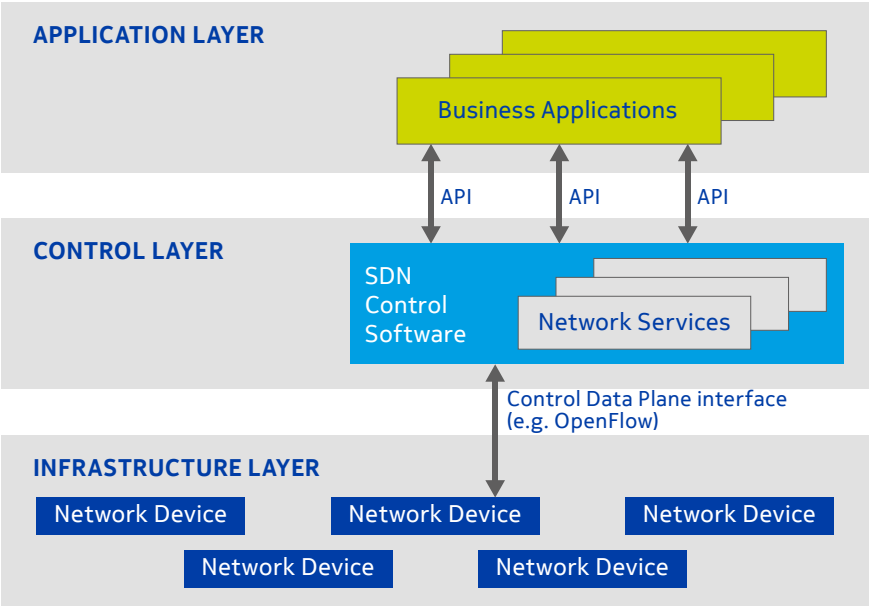
4.2. Open networking and a big step towards SDN

The word “open” suddenly appeared on the list of tools like Open APIs, OpenFlow, Open Compute, Open vSwitch, OpenDaylight, and OpenConfig. The idea was to improve automation as much as it is possible. This movement shows that there is a big need for automation in a much more programmatic manner, better suited for modern network architecture. Network devices and controllers are opening themselves to be programmed. The fact that more and more devices appear with Python on-box, confirms how devices and operating systems evolve. Another great thing used for automation is Representational State Transfer (REST), which develops, and designs network applications. RESTful APIs is a device that exposes REST, and general HTTP-based API, which requires a web server accessible via URL, which then has to be sent to that URL (that is, a SDN controller). New approach of programming network by REST API is implemented in many new devices.

Fortunately, automation technologies like REST, Python on-box, and other mentioned above became more popular, resulting in easier automation. This leads to Software Defined Networking (SDN) being a solution for the modern data center. Having SDN applied to the infrastructure can extend and expand effectiveness, scale, and flexibility. Looking at the overall state of the network is very complex when considering individual elements and trying to build up an overall view. SDN allows to view the entire network as a system, which helps to make better routing and forwarding decisions. SDN architecture defines how a networking and computing system can be built using a combination of open, software-based technologies and commodity networking hardware that separate the SDN control plane and the SDN data plane of the networking stack.

Traditionally, both the SDN control plane and data plane elements of a networking architecture were packaged in proprietary, integrated code distributed by one or a combination of proprietary vendors. The OpenFlow standard, created in 2008, was recognized as the first SDN architecture that defined how the control and data plane elements would be separated and communicate with each other using the OpenFlow protocol. The Open Network Foundation (ONF) is the body in charge of managing OpenFlow standards, which are an open source. However, there are other standards and open-source organizations with SDN resources, so OpenFlow is not the only protocol that makes up an SDN. This gives the possibility to run the control software on modern, multi-core processors, update the control algorithms more frequently, and implement simpler data plane.

Figure 9 SDN architecture



4.3. SDN Architecture

SDN in practice is split into three groups of functionalities.

SDN Applications are programs request behaviors and needed resources from SDN Controller via application programming interface (API). Applications can build an abstracted view of the network by gathering information from the controller for decision-making purposes. These applications could include networking management, analytics, or business applications used to run large data centers. For example, an analytics application might be built to recognize suspicious network activity for security purposes.

SDN Controller is a logical unit that receives commands or requirements from the SDN Application layer and passes them to the networking components. The controller also extracts information about the network from the hardware devices and communicates back to the SDN Applications with an abstract view of the network, including statistics and events that are happening

SDN Networking Devices control the forwarding and data processing capabilities for the network. This includes forwarding and processing of the data path.

Multi-vendor automation was always hard to achieve because most of the vendors focused on their devices, not on creating the common language in networking. SDN based solutions are clearly an

everyday tool used to manage and automate the networks. This is the only way to achieve high system responsiveness, thanks to an automation capabilities. Truly Cloud/Application driven data-centers require rapidly application driven networking, and this is where SDN gives a big promise. Current implementations that are already known are still missing on unification (different vendors have some specifics), which reduces the potential to build truly modular networks. What can be done is to use the existing automation tools for everyday business tasks. Still, the true programmable network comes with SDN solutions, and this is what the world requires today, not tomorrow.

References

[1] Marwan Al-shawi, Andre Laurent, Designing for Cisco Network Service Architectures. Cisco Press 2017.
[2] Marwan Al-shawi, CCDE Study Guide. Cisco Press 2015.
[3] Duglas Richar Hanks, A comprehensive guide to building next-generation networks. Douglas Richard Hanks, Jr. All rights reserved. 2015.

About the authors

Grzegorz Dygon, R&D Engineer in Common Development Service.
Łukasz Michna, R&D Laboratory Specialist, over 10-years in IP network area field, Cisco certified engineer.
Stanisław Pospiech, Verification Architect in MN 5G&SC. Certified enthusiast of Networking and Cloud. In Nokia since 2010.

Using Artificial Intelligence for radio performance test result analysis

Adam Chamera
Tester / Integrator
MN SRAN



1. Radio Network Performance

Our customers want to have the maximum network performance achieved with the existing radio network resources. As the traffic within the network increases, and because of its high dynamic, maintaining network performance is becoming more and more important. Network Performance has a great impact on a mobile operator's revenues. It is becoming increasingly important to have proper measurements sources and solutions. All to have a clear understanding of network behavior and traffic distribution, and to keep the real network environment under control. Network optimization is an important task in Mobile Network Management, in order for network performance to satisfy certain thresholds or targets for Key Performance Indicators (KPI) to ensure network quality [1].

2. Key Performance Indicators

KPIs are a set of selected indicators, appropriate for a given radio technology, used as an input for measuring network performance and trends. KPIs can give warnings of potential problems and can be used to find badly behaving areas, RNCs, or sites [1].

Exemplary KPIs are as follows:

- Cell availability
- Call Setup Success Rate (CSSR)
- Packet Session setup/success rate (NRT, HSDPA, HSUPA)
- Call Drop rate (Call Retainability)
- SHO/ISHO/HSPA SCC success rate
- Call & Session setup
- MultiRAB
- Mobility
- Throughput & Efficiency
- Quality
- Number of users
- HARQ Failed
- Connected Ues
- Intra eNB Handover
- Drops per Volume

...and many more. There is plenty of KPIs varying with different RATs or even between SW releases. KPIs are usually calculated based on values of counters (numeric or of the enumerator type). Network optimization starts with KPI identification and agreement with the customer on those KPIs that will reflect network performance and the targets that need to be achieved once optimization has been concluded. A KPI could be based upon network statistics, UE drive testing, or a combination of both. Based on this, there are two different sets of KPIs:

- Field KPIs: reflecting network performance from the user (User Equipment) perspective. They are related to coverage and call performance. Their main assignment is to adjust antenna azimuths and tilts, find installation errors, and show the quality of the network in terms of coverage.
- Network KPIs: show network performance on different levels (PLMN, site, cell). They are based on counters which reflect network performance with a cell level granularity or cell-pair level (e.g. for HO performance). The scope covered by these KPIs is not necessarily limited to a certain area, but to the whole network [1].

3. My name is Machine, Learned Machine

Cognition, or learning, is a process of acquiring knowledge, where knowledge could be understood as a human mental state [2]. Nowadays, we want to bestow knowledge and intelligence on machines - and make them work for the human benefit. We call it Artificial intelligence (AI). The idea of a machine that is able to learn something is not new. It is at least 70 years old, just look at Claude Shannon and his mouse learning to cross a maze [3]. Nowadays we are facing a significant increase in the available computing power of modern computers and huge increase in the possible collection of data and logs documenting the operation of devices. It is causing the growth of popularity of programming solutions based on the methods of AI, such as Machine Learning techniques. Self-driving cars are being created, as well as mechanisms for recognizing complex images or the human voice. Tools for diagnosing cancer and genetic tests have been developed. Mechanisms of online stores, mailboxes, and social networks are matching the presented content to the end user's preferences based on his or hers previously observed behavior [4].

It is possible that Machine Learning will even solve simple problems! Since according to *Moravec's Paradox*, difficult problems can already be solved by means of the systems available today! [5]

The fields of Machine Learning (ML), Deep Learning (DL) in particular, have seen a very rapid growth in recent years; their applications now extend towards almost every industry and research domain. The main reason for this, is that extremely accurate and versatile systems and channel models have been developed that enable algorithm design grounded in information theory, statistics, and signal processing, with reliable performance guarantees. Another reason is that, only since very recently, powerful DL software libraries (e.g., python's Theano, TensorFlow, Keras, MXNet, Torch7) and specialized hardware, such as graphic processing units (GPUs, e.g. available in **NOKIA GARAGE** in Wroclaw) and field programmable gate arrays (FPGAs and increasingly other forms of specialized chips) are cheaply and widely available. These are key for training and inference of complex DL models needed for real-time signal processing applications [6].

Perhaps the most known AI applications are image recognition and self-driving cars.

Furthermore, computational intelligence paradigms (e.g. neural networks and neuro-fuzzy methods), swarm intelligence algorithms (e.g. ant colony optimization), and evolutionary algorithms (e.g. the competitive imperialist algorithm), may also be applied to improve the performance of mobile networks. Among these compelling techniques, neural networks and Deep Learning have recently become particularly popular [7].

Generally, a neural network consists of a number of neurons and weighted connections among them, where the neurons can be regarded as variables and the weights can be viewed as parameters. The network should be appropriately configured with the aid of learning techniques to ensure that the application of a set of inputs produces the desired set of outputs. Neural networks have been widely utilized for spectral white state estimation, prediction, and handover decisions in cognitive radio networks [7].

ML algorithms can be simply categorized as supervised and unsupervised learning, where the adjectives “supervised/unsupervised” indicate whether there are labeled samples in the database. Another kind, reinforcement learning, is concerned with reward/utility of an agent, which is connected to its environment via perception and action.

The family of ML algorithms can also be categorized based on their similarity in terms of their functionality and structure in the following way:

- Yielding regression algorithms
- Instance-based algorithms
- Regularization algorithms
- Decision tree algorithms
- Bayesian algorithms
- Clustering algorithms
- Association rule-based learning algorithms
- Artificial neural networks
- Deep Learning algorithms
- Dimension reduction algorithms
- Ensemble algorithms

and more. ML can be widely used in modeling various technical problems of next-generation systems, such as large-scale MIMO, device-to-device (D2D) networks, heterogeneous networks constituted by femtocells and small cells, and more. **Figure 1** portrays the family tree of ML techniques and their potential applications in 5G [7].

ML can also be applied in examination of network functionality and location of the the areas suitable for possible increase in radio network performance. ML techniques feed themselves with big amounts of ordered data. The hudge amount of historical records of

KPIs, counters, and results of measurements of radio signal parameters could be the base for machine-run analysis. Some parameters of radio signal, like Output Power, ACLR (Adjacent Channel Leak Ratio), CCDF (Complementary Cumulative Distribution Function), EVM (Error Vector Magnitude), Frequency Error, and BLER (Block Error Rate), were deeply described in the first edition of the Nokia Book [8][9] and firstly within the 3GPP specification [10].

4. Performance Analysis

The scope of Performance Analysis is to check network performance and to provide recommendations in the form of corrective actions to be carried out in order to improve performance. The aim of this process is to collect and report KPI’s, to have a failure cause breakdown per category, and to list the worst contributing BTSs or cells.

The activities of Performance Analysis are:

- Analyzing and proposing means for performance improvement in the following areas:
 - Accessibility (including capacity checks)
 - Retainability
 - Mobility
 - Usage (including operational availability)
 - Integrity (including QoS)
- Analyzing any other KPIs defined in specific customer contracts
- OSS Performance Benchmarking

The inputs for Performance Analysis are:

- Performance data (PM - Performance Management)
- Alarms/faults (FM - Fault Management)
- Log data collection and preparation (Megamon/Megaplexer data)
- Several groups of KPI’s, such as RRC setup-, (E) RAB setup-, Voice-, VoLTE-, Packet Session-, HSPA- and Mobility Performance.
- A commonly agreed set of KPIs based on network statistics

In practise, statistic KPIs are used as input for detecting performance problems and monitoring. At least two-week data of daily average value collected from NetAct Reporting Suites Reporting Suites for KPI evaluation is needed. The agreed set of KPIs should be created for an agreed aggregation level (Network, BSC) and cell level (busy hour data) is used for detailed analysis and optimization.

Examples of recommendations are as follows:

- Checks of BTS configurations, HW faults, power settings
- Coverage or RF quality checks
- Checks of capacity issues
- Parameter checks

- Check of the activation of features
- Implementation of latest technical notes
- Checks of Admission control resources
- Checks of layering strategies, neighbouring definitions, and parameterizations

The results report is usually focused on the list of top N worst performing cells or BTSs.

Performance Analysis, besides other types of analysis such as parameter, configuration, and RF analysis, is an introduction to a more specific action - Root Cause Analysis (RCA) as a part of RAN Assessment process [1].

5. RAN Assessment

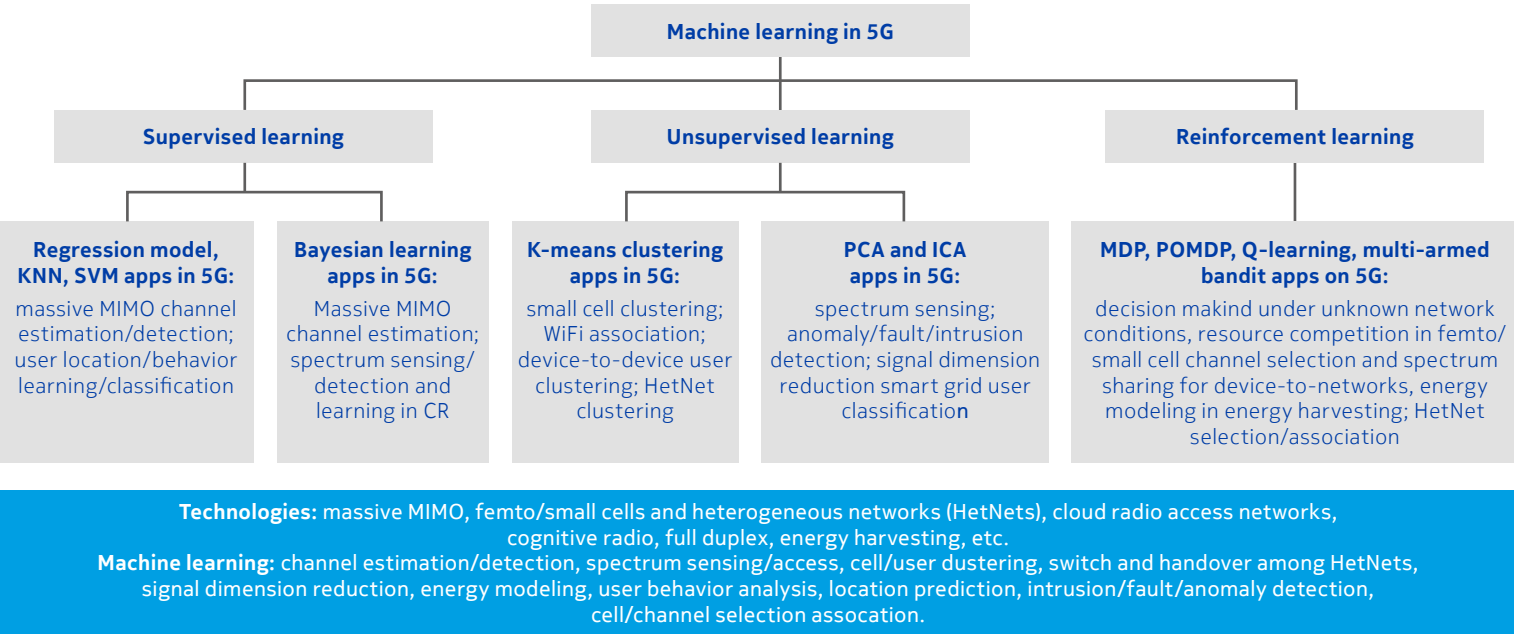
RAN Assessment provides an efficient way to find the prevailing status of the examined network. A snapshot of the network is taken and configuration and performance are analyzed. The service enables to:

- Understand the optimization potential of the network
- Locate, identify, and analyze performance problems, including layering aspects
- Suggest possible improvement actions
- Assess network performance trends
- Find out root cause(s) - Root Cause Analysis

The RAN Assessment service consists of the following Modular Working Items (MWI) [1]:

- 1. Performance Analysis.**
- 2. Capacity Analysis** - comparison of the utilized network capacity vs. the offered capacity. Current capacity usage level and already existing capacity shortages are reported.
- 3. Parameter Analysis** - consists of the following main activities: Radio parameter plan verification, Radio parameter defaults verification, Radio parameter consistency, Neighbor Assessment (CM based), Feature usage assessment.
- 4. Configuration Analysis** - evaluation of network configuration and topology
- 5. RF Analysis** - focuses on cell coverage and interference analysis and is based on Performance Management (PM) and Configuration Management (CM) information, including site and antenna data (Interference matrix); network logs and radio network performance data collected directly at network controllers (e.g. RNC); Drive tests.

Figure 1 Radio learning architecture [7].



6. Radio Capacity Analysis - aims to find problems related to capacity licensing and BTS configuration problems affecting capacity (Missing rx diversity, HSDPA scheduler configurations etc.).

7. Mobile Access Parameter and Capacity Analysis - identifies transmission path OSS performance, parameter and configuration settings in the BTS and in the controller (BSC/RNC) which potentially degrade network performance. It also analyzes the logical interface performance (for example, in WCDMA, the lub and lu-CS and lu-PS interfaces) based on RAN counter data. Selected KPIs is observed and shortages are reported [1].

8. 3G and 4G MBH (Mobile backhaul) Performance Analysis, as a part of RAN Assessment within the overall mobile network domain, consisting of Measurement, Light Analysis (Detailed Analysis as optional, follows a Light Analysis), and Reporting. After examining internal counters, transport equipment, TWAMP (Two-Way Active Measurement Protocol) measurement per QoS/CoS, analyzing statistic data and KPI variation, we can deliver recognition

of Worst Performers among network elements and Site Classification, Indication of problem pattern, Identification of under-performing sites, and Root cause analysis with a summary list of recommendations. This service can usually take more than two weeks [11].

6. Conclusions

It is becoming increasingly important to have the proper measurements sources and solutions in order to have a clear understanding of network behavior, traffic distribution, and to keep the real network environment under control. We are using ML algorithms analyzing high quality data coming from critical points in the network for failure pattern identification and custom KPI calculation. In the coming years, continuing focus on operational efficiency will propel the demand for ML and analytics-based assurance automation. To solve these challenges, Nokia has already developed several products and is still working on more. Just to mention some of them:

Nokia Autonomous Customer Care

The Nokia Autonomous Customer Care solution uses emerging technologies like AI, ML, speech recognition, natural language processing (NLP), natural language understanding (NLU), and bots to enhance the delivery of omni-channel customer care. This is a solution for resolving customer issues in the shortest time, with the least number of steps and human interventions and that means reducing OPEX (operating expenditures), decreasing customer and operator effort [13].

EdenNet SON solution

EdenNet is a Self-Organizing-Networks solution that is using Machine Learning techniques. It enables mobile operators to automate and efficiently realize the full potential of their existing networks, as well as drive transformation to 5G. It offers the industry's widest range of SON modules, helping operators achieve self-configuration, self-healing, and self-optimization [14].

Nokia evolved Service Operations Center (eSOC) solution

eSOC enables communication service providers (CSPs)/digital service providers (DSPs) to proactively detect network and subscriber performance degradation, including anomalies and congestion over physical and virtual network and services resources, including applications, IoT, and devices. It can detect locations exhibiting large amounts of handover or performance trends impacting 2G, 3G, 4G, Wi-Fi, and 5G networks. It can, for instance, monitor:

- Quality and performance of VoLTE services
- The impact of a new device or application prior to launch
- Roaming volume generated by each roaming partner

eSOC combines service quality management, customer/subscriber-centric analytics, and cultivated and machine intelligence, including automated problem detection and ticketing, as well as automated recovery through intelligent agents and machine learning. There are also editors for creating or modifying models for services, subscribers, networks, data centers, devices, locations, and segments. It enables drilling down to root-cause management systems, such as OSS, to take and recommend actions [15].

References

- [1] [Ed.] Viero J., NPO Modular Working Item (MWI) Wiki, NOKIA last review 2017-07-05, <https://nokia.sharepoint.com/sites/NPOServices/Wiki/NPO%20MWI/Home.aspx>
- [2] Bocheński J., Współczesne Metody Myślenia, „W drodze” Poznań 1992.
- [3] Gleick J., Informacja. Bit – Wszechświat – Rewolucja, „Znak” Kraków 2012.
- [4] Hearty J., Zaawansowane uczenie maszynowe z językiem Python, Helion 2017.
- [5] Hans Moravec H., Mind Children, Harvard University Press. 1988.

- [6] Dörner S., Cammerer S., Hoydis J., ten Brink S., Deep Learning-Based Communication Over the Air
- [7] Chunxiao Jiang, Haijun Zhang, Yong Ren, Zhu Han, Kwang-Cheng Chen, Lajos Hanzo, Machine Learning Paradigms for next generation wireless networks, IEEE Wireless Communications 2016. <https://ieeexplore.ieee.org/document/7792374/?part=1>
- [8] Miernik M. Orzechowski A., Parametry sygnałów radiowych, [in:] Praktyczny wstęp do wytwarzania systemów oprogramowania, które zmieniają bieg historii, NOKIA 2015.
- [9] Domański M., Pomiar parametrów radiowych stacji bazowych LTE (eNodeB) przy użyciu wektorowego generatora sygnałowego i analizatora widma, [in:] Praktyczny wstęp do wytwarzania systemów oprogramowania, które zmieniają bieg historii, NOKIA 2015.
- [10] 3GPP TS 36.141 „Base Station (BS) conformance testing” <http://www.3gpp.org/dynareport/36141.htm>
- [11] Ciriaco E., RAN Assessment with MBH Performance Analysis - Overview, Nokia 03-08-2017
- [12] Itkonen J., MWI: Root-cause analysis, NOKIA 2018. <https://confluence.int.net.nokia.com/pages/viewpage.action?pageId=475242650>
- [13] Nokia Autonomous Customer Care, NOKIA 2017, SR1705010801EN (May) Solution sheet
- [14] <https://networks.nokia.com/solutions/edennet>, 12-06-2018
- [15] Becoming a digital service provider: Profiting from 5G, IoT and virtualization with service-oriented operations, NOKIA White Paper 2017, SR1703009227EN (September)

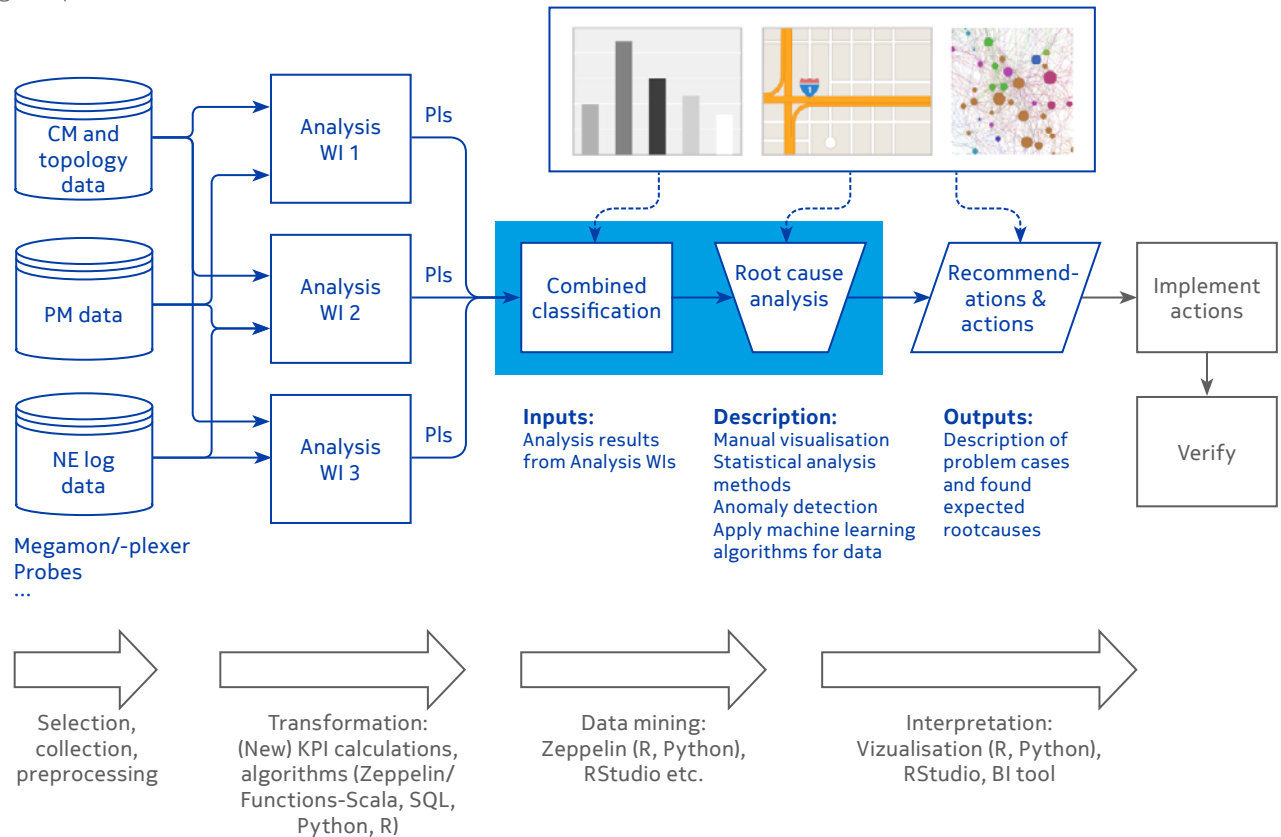
About the author

I work as a Tester and Integrator in the MN SRAN team testing Nokia system modules software for new features of WCDMA Air interfaces. We conduct manual and automatic tests. Depending on SW development phase, those tests are integration, verification, or regression. The problems of heterogeneous networks and the multitude of technologies and used devices make our work interesting and full of challenges.

Adam Chamera

Tester / Integrator
MN Mobile Networks

Figure 2 Taking advantage of ML algorithms, Root Cause Analysis utilises the output from other Working Items when finding the cause of the given problem [12]



The broadband access technology would not develop so rapidly, if it was not for the Laboratory of the European Software and Engineering in Wrocław. The best ideas and projects in this field are born here. You can be their creator, too, since in every laboratory, especially in ours, the most important thing is people.

OUR LABORATORIES IN NUMBERS

01 over
3000 km
of cables,

which
is the distance
from Gibraltar
to Wrocław



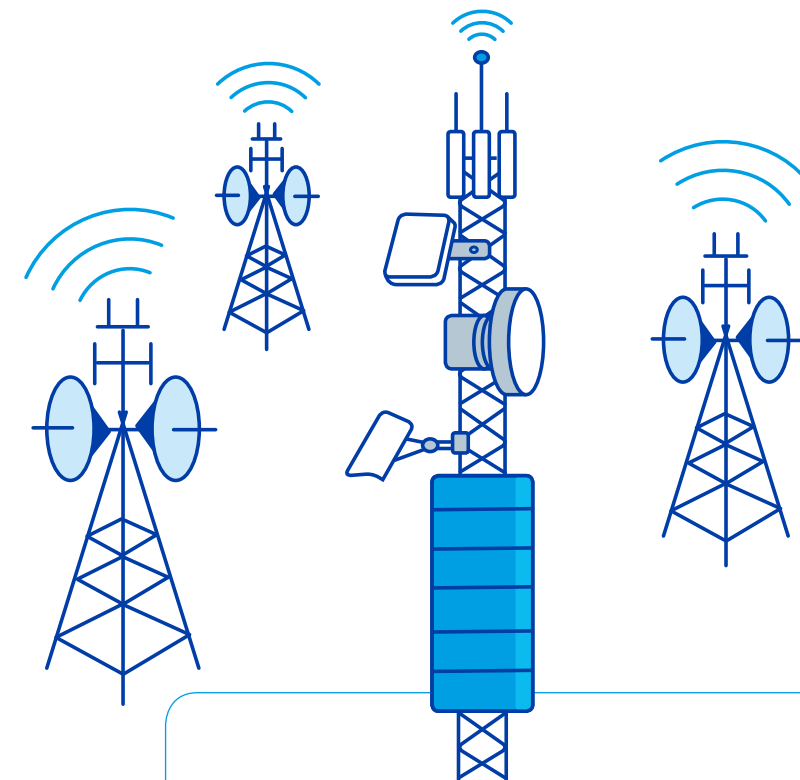
02 **6000 m²,**
15 basketball
courts



03 **8450**
testing devices

with a total weight
of **1000 tonnes**,

which is as much
as **6 Jumbo Jets**



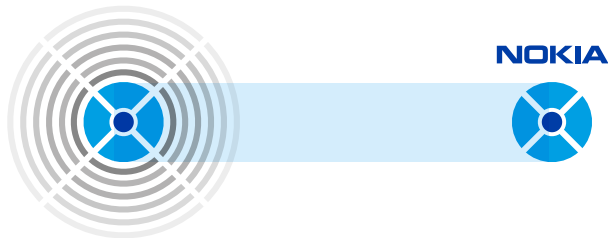
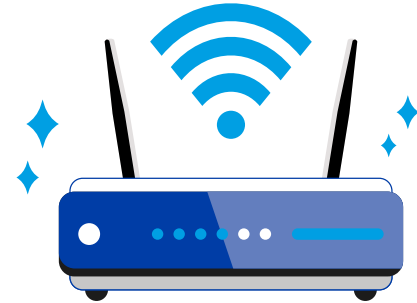
05 **1700 BTSs**
(base stations),

that can cover an area
of **19000 km²,**
which is as much as the area
around the Lower Silesian
Voivodeship

06

The total available transmit power is **600 000 W** (base stations),

Compared to the power of an average WiFi router (0,1 W) you get the equivalent of **6 million** such devices



07

The transmission of radiation is **7 times lower** than the permissible standards

08

over **3000** mobile phones and modems for testing



09

All Nokia branches around the world have remote access to the resources of the Wrocław laboratory

that means that the potential of the laboratory is used **24 hours a day**

