Benchmarking computers and computer networks

Stefan Bouckaert, Jono Vanhie-Van Gerwen, Ingrid Moerman - IBBT, Ghent University, Belgium
Stephen C Phillips, IT Innovation Centre, UK
Jerker Wilander, FIREstation
Shafqat Ur Rehman, Walid Dabbous, Thierry Turletti, INRIA, France

Introduction

The benchmarking concept is not new in the field of computing or computer networking. With "benchmarking tools", one usually refers to a program or set of programs, used to evaluate the performance of a solution under certain reference conditions, relative to the performance of another solution. Since the 1970s, benchmarking techniques have been used to measure the performance of computers and computer networks. Benchmarking of applications and virtual machines in an Infrastructure-as-a-Service (IaaS) context is being researched in a BonFIRE experiment and the benchmarking of wired and wireless computer networks is put forward as a research topic in the research projects CREW and OneLab2. In this paper, we elaborate on the interpretation of the term "benchmarking" in these projects, and answer why research on benchmarking is still relevant today. After presenting a high-level generic benchmarking architecture, the possibilities of benchmarking are illustrated through two examples: benchmarking cloud services and benchmarking cognitive radio solutions.

Definition and benchmarking aspects

In the scope of the performance analysis of computer systems, we define *benchmarking* as the act of measuring and evaluating computational performance, networking protocols, devices and networks, under reference conditions, relative to a reference evaluation. The goal of this benchmarking process is to enable fair comparison between different solutions, or between subsequent developments of a System Under Test (SUT). These measurements include *primary performance metrics,* collected directly from the SUT (e.g. application throughput, node power consumption), and in case of wireless networks also *secondary performance metrics,* characterizing the environment in which the SUT is operating (e.g. interference characteristics, channel occupancy). The primary and secondary performance metrics may be complemented by *techno-economic metrics*, such as device cost or operational complexity.

The following terminology will be used throughout this paper:

A ***benchmark*** contains a full set of specifications needed to enable the evaluation of the performance of a certain aspect of a SUT. These specifications include *(i) a* ***scenario****, (ii)* performance evaluation ***criteria*** ¸ *(iii)* performance evaluation ***metrics,*** and *(iv)* a benchmarking ***score***. Depending on the context, the term benchmark may further also indicate a specific experiment that is executed according to the benchmark; e.g.: to run a “throughput benchmark” v.s. “the throughput benchmark is designed to fairly compare end-to-end throughput between *x* and *y* under condition *z*”.

A scenario is a detailed description of the set-up of the experiment needed to run the benchmark. For example, it may contain a network topology, protocol configuration parameters or traffic traces. The *criteria* describe the high-level focus of the output of the benchmark. Example criteria are energy efficiency or robustness to failing links. A metric is a quantitative measure of a specific quality of a SUT. Metrics are determined according to a specific methodology.

Although benchmarking, in its strictest sense, is limited to measuring performance, several additional aspects are important to make benchmarking a meaningful research activity. Therefore, the following aspects are to be considered when defining benchmarks;

Comparability should be a fundamental property of any benchmark; comparability means that two independently executed benchmarks can be meaningfully compared to each other. One of the factors influencing the comparability is **repeatability:** running an identical benchmark on an identical solution at different moments in time should result in a (close to) identical result. Furthermore, well-defined **experimentation methodologies** are a key factor in achieving comparability.

Ideally, benchmarking scores should not only be comparable to other scores obtained using the same testbed, but also with scores obtained from different testbeds with similar capabilities but potentially running different operating systems, or based on different types of hardware. The success of a specific benchmark may very well depend on whether this **interoperability** aspect is satisfied or not.

Test infrastructures may be equipped with benchmarking functionality. In this case, the **configurability** of the testbed environment is crucial: in wired networks, a benchmark may require a specific topology and links of specific quality; in wireless networks, a trace containing reference background traffic may need to be played back during the execution of a benchmark. If such benchmarking scenarios are considered, obviously, the testbed needs to support these configuration settings.

A benchmarking framework

Defining benchmarks is one challenge; executing a benchmark in an experimenter-friendly way, sharing benchmarks with the research community, and/or building a sustainable framework is equally important. Figure 1 presents a conceptual drawing that captures the building blocks of a benchmarking framework that can be implemented on top of testbed infrastructures.

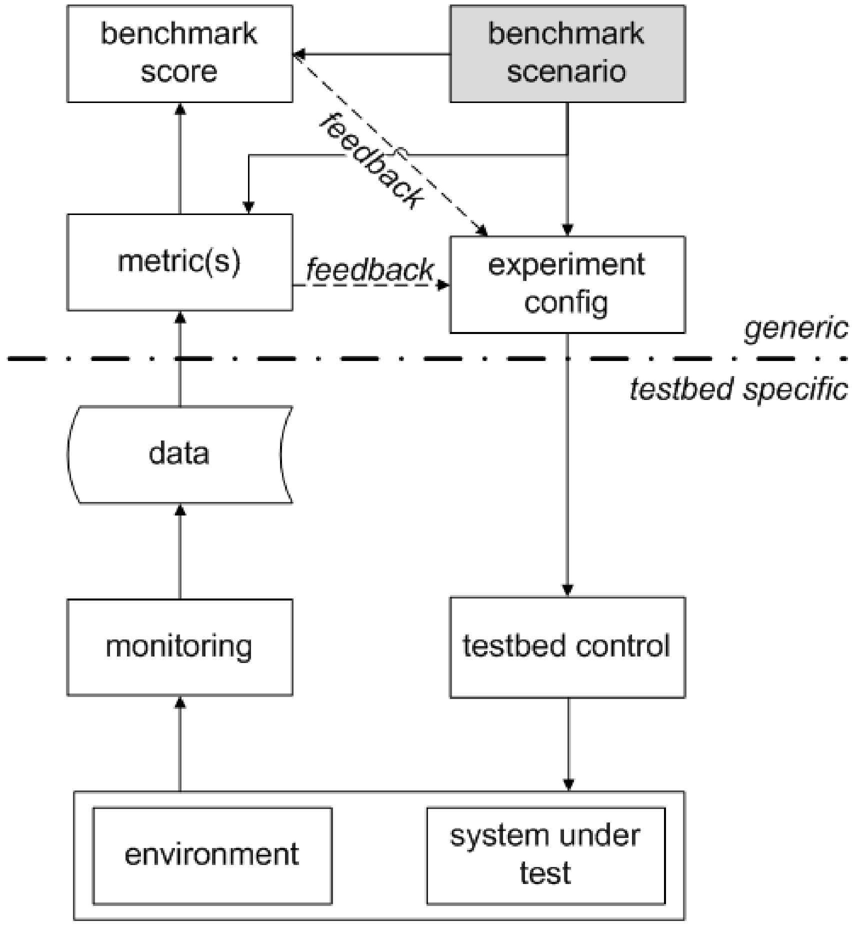

Figure 1: high level overview of a benchmarking framework

The building blocks below the dash-dotted line indicate testbed specific building blocks, while the building blocks above this line are of generic nature. As indicated in the previous section, the benchmarking scenario contains the configuration of the experiment, the metrics that will be determined, and the way for determining the benchmarking score.

Every testbed implementing the benchmark will then implement the benchmark scenario in a testbed-specific way via the testbed control functionality. The testbed control functionality is responsible for the configuration of the SUT, but also -mainly in wireless networks- for the configuration of the environment. As an example, it is possible that an experimenter wants to benchmark the reliability of a wireless system in terms of end-to-end packet loss, with a predefined number of wireless nodes generating interference by sending predefined packet sequences. These interfering nodes are considered to be part of the wireless environment rather than the SUT, since they are not the subject of the performance evaluation.

The monitoring block is responsible for gathering raw data while the experiment is running; monitoring functions may collect data from the SUT as well as from the environment. Different testbeds may collect data in different ways, be it by storing values in databases, or for example in comma separated value (CSV) log files. Note that in this representation, the testbed control is responsible for configuring the monitoring functions that are active during an experiment.

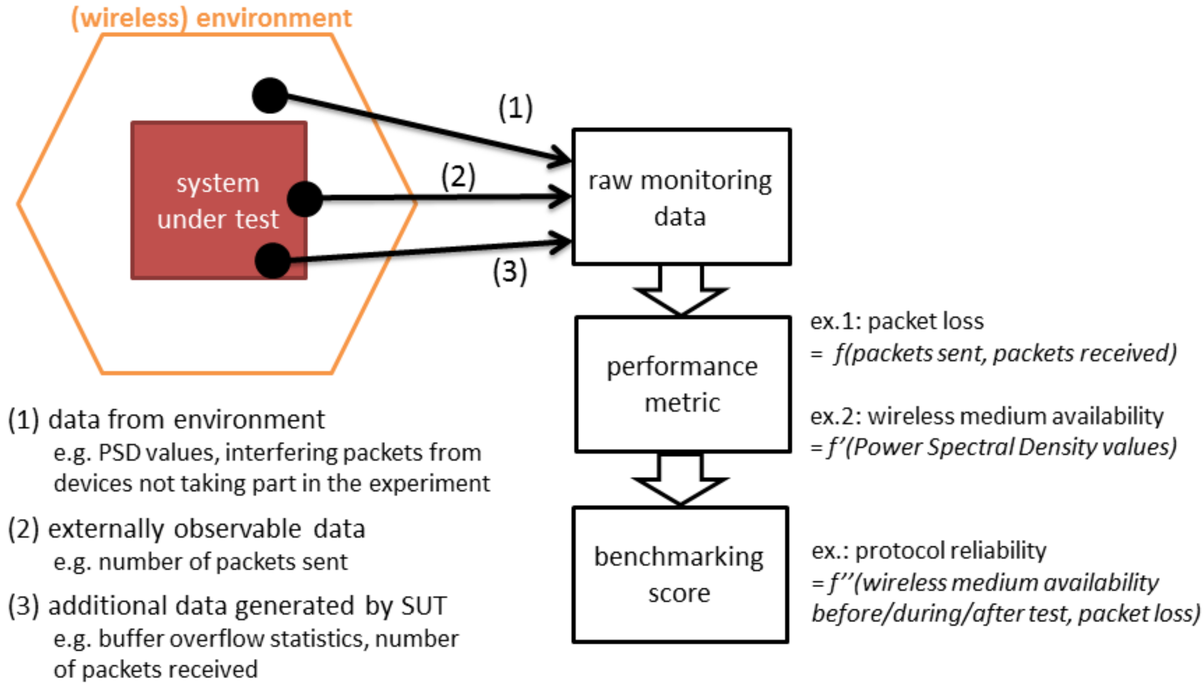

Figure 2: determining a protocol reliability score based on data collected from the environment and the SUT

In a next step, the raw data is converted according to the specifications of the metrics that are unambiguously defined by the benchmark. Thus, while the raw monitoring data may still be testbed dependent, the metric(s) are not.

Figure 2 illustrates the relation between data and metrics; Raw data can be collected from an experimental set-up in different ways: (1) dedicated solutions may be added in the environment of the SUT, to collect performance data. For example, during an experiment involving wireless networks, a spectrum analyser may be used to collect power spectral density (PSD) values in certain parts of the frequency spectrum to characterize the occupation of the wireless environment. In this example, the measured PSD values do not necessarily characterize the SUT itself, but may be of interest to the experimenter to explain anomalies in test results, caused by external interferers such as microwave ovens. (2) Certain characteristics of the SUT may be measured without interfering with the operation of the SUT itself. For example, a wireless sniffer or Ethernet packet sniffer may be used to capture packets sent by a SUT. (3) Other types of data characterizing the SUT may only be available when an experimenter has access to the internal operation of a SUT. For example, buffer overflow statistics of a particular node in a network can only be accessed by directly interfacing with that node itself.

In a next step, from this raw data, no matter what the source or exact type of it and no matter how it is stored in a particular testbed, unambiguous metrics are derived. Providing unambiguous metrics is only possible by providing clear and agreed upon definitions of how the metrics are determined from the raw data, and by ensuring that the raw data itself was measured according to standardized methodologies. An example metric is end-to-end link layer packet loss between two nodes during an experiment, which is defined as the difference in the number of link layer packets sent by the sending node to a receiving node and the number of these packets that are received at the link layer by the receiving node, relative to the total number of link layer packets sent. In this case, the raw data could be link layer packet traces, and the metric is a single number.

The metrics may be used directly by an experimenter to evaluate the performance of the solution. However, in the proposed benchmarking approach, an additional level of abstraction is introduced: one or multiple metrics can be combined to form one or multiple benchmarking scores. Each benchmarking score may shed light on the performance of the SUT related to a specific criterion. Example criteria are energy efficiency or reliability. In the example of Figure 2, a reliability benchmarking score is determined based on (wireless) packet loss and wireless medium availability before, during and after an experiment. An example outcome could be a “high” or ”good” reliability benchmarking score in case packet loss is close to zero when the relevant part of the RF spectrum is already heavily loaded. In a next experiment, the reliability score could be relatively lower, even when fewer packets are lost, because the wireless spectrum is almost unoccupied at the moment that the experiment takes place.

Obviously, determining how to combine metrics into a particular benchmarking score is a non-trivial task. However, it is worth the effort as by converting a set of metrics to a benchmarking score, additional performance analysis features are enabled;

Firstly, when scheduling a single experiment (i.e. one fixed set of configuration parameters for the SUT), benchmark scores provide no additional information compared to the performance metrics - in fact the benchmarking score hides performance details. However, a thorough performance analysis may require tens, hundreds or even thousands of tests to be scheduled. In this case, studying the individual metrics is a lot of work. If a benchmarking score is then defined in such way that it combines all relevant metrics for the criteria under evaluation, the benchmark scores greatly help to quickly classify the experiments relative to each other. Provided that the comparability and interoperability requirements are satisfied, the benchmarking scores can then be used to quickly and fairly compare the performance of different solutions, or of subsequent developments of a single solution. Moreover, the performance of a single solution in varying environments can be determined and compared.

Secondly, under the assumption that meaningful benchmarking scores can be defined, performance comparison of different solutions may by possible by non-experts, since no detailed understanding is required of the specific metrics. Furthermore, the benchmarking scores can be used as a way to reconfigure experiments, in order to automate performance evaluation. This is indicated by the dashed arrows in Figure 1: Based on the benchmarking score, a computer program can detect the influence of a particular experiment configuration on the performance. The program could then propose a new configuration (for example, run the same experiment again, but with a single parameter such as "beacon interval" set to a different value), re-evaluate the score, and auto-optimize a particular SUT. In those zones with a large benchmark score variation, a finer grained tuning of the configuration parameters could be scheduled. Eventually, a user could define ranges for different parameters of the SUT to be varied, and then let the benchmarking framework and testbed take care of intelligently scheduling additional tests, finally presenting a user with an easy-to-understand overview of the performance of the SUT.

Bonfire Case: Cloud Benchmarks

Context

Today, different Infrastructure-as-a-Service (IaaS) providers describe their infrastructure offerings in different ways and don't necessarily provide very much information, if at all, about the infrastructure being offered. For instance, Amazon EC2 describes (and prices) their infrastructure in terms of Amazon EC2 Compute Units (ECU). A machine providing the capability of one ECU is said to be equivalent to a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. Given the limited and heterogeneous information provided by IaaS providers, how can anyone know what resources they will need to execute their application with a particular quality of service (QoS)? If the application is already adapted for the IaaS provider's system then it may be possible to just try the application out and measure its performance, scaling the deployment as required. But what if the application is not yet adapted, or what if you want to choose between several IaaS providers?

Providing tools to help choose the appropriate IaaS provider and predict application performance lies within the realm of the Platform-as-a-Service (PaaS) provider as part of the wider role of helping the application provider develop, deploy and manage their application. To create Software-as-a-Service (SaaS) either an existing application requires adaptation or a new application is created. In either case, PaaS tools help in the process of creating software to execute on an IaaS provider (see Figure 1 below). The major PaaS providers today (Google App Engine and Microsoft Azure) take the role of IaaS provider as well, but this needn't necessarily be the case. In the future the IaaS market may be more developed and we will need a standard way to compare providers. This is one case where benchmarks and application models can help (see below).

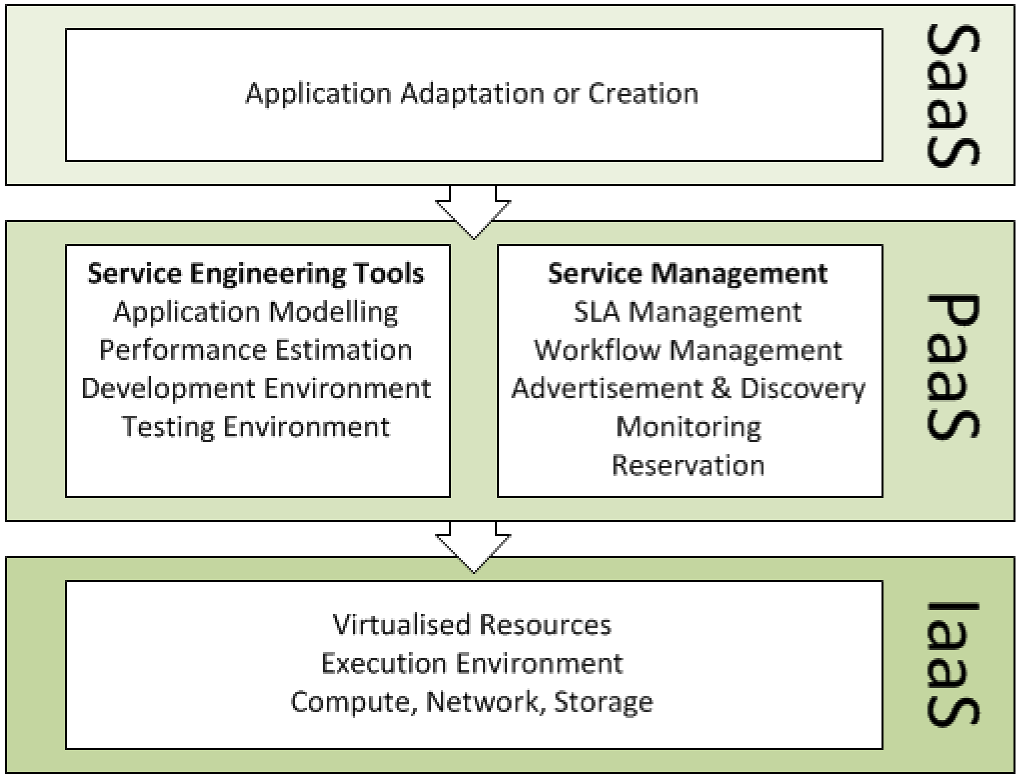

Figure 3: Application modelling is a service in the Platform as a Service layer.

The BonFIRE project has an embedded experiment researching how to measure the performance of the virtual machines deployed by IaaS providers. Rather than just executing standard benchmark tests on the virtual infrastructure, the experiment is working at a higher level of abstraction by using so called "dwarf" benchmarks - an approach first introduced by Colella in 2004 [1], which has been further developed at UC Berkeley [2]. Dwarf benchmarks are intended to capture known computational patterns and may map more directly onto application performance.

Application Modelling

This paper has discussed a general framework for collecting performance data and has described the complexity and advantages of converting this performance data into simpler benchmark scores. However, that is not the end of the story: How can benchmark scores be used to make decisions?

To give a simple example: If you had a computer application which purely did floating point operations (no IO, no network) and you had to choose which computer to buy to run your application then you could look up the SPECfp scores [3] for the CPUs in the candidate computers and buy the fastest computer (in this limited sense) for your budget.

This simple case is unrealistic. Applications do not map onto benchmark scores in such a straightforward way. Given an application and a hardware configuration you need to understand a certain amount of detail of your application's behaviour to know what benchmark or benchmarks are most appropriate to use in your analysis.

To predict application performance, an application model is required that can relate application workload, hardware characteristics and application performance. A description of the hardware in terms of Amazon ECUs or similar opaque terms from other providers is not sufficient to produce a predictive model. Would it help to know in detail the hardware being offered by the IaaS provider? Possibly, but IaaS providers are not generally going to give their customers this information as it is confidential.

The IRMOS project [4] has constructed some performance models of applications taking into account the detail of the hardware, and whilst they can be predictive, these models are time-consuming to create and require expert knowledge. Instead we need a middle ground. The BonFIRE experiment previously mentioned is researching whether virtual hardware descriptions couched in terms of dwarf benchmark scores can make application models simpler and yet still sufficiently predictive to help in a variety of ways:

- Making better provisioning decisions: Deploying the infrastructure resources required for a given application QoS rather than over-provisioning.
- Making better application scheduling decisions: Knowing the application runtime with a good reliability permits more intelligent scheduling.
- Determining the optimal application configuration: The performance of complex applications and business or industrial data processing workflows with many components can be greatly affected by their configuration such as buffer sizes and number of threads.
- Tracking uncertainty in business processes: Many processes are non-deterministic, predicting the likelihood of completing tasks allows for the management of risk.

Application modelling is a broad field covering many modelling techniques including discrete event simulations, Bayesian belief networks, artificial neural networks and finite state machines. All these models are underpinned by the description of the system that the application is running on, making benchmarking an important asset today for business and economic reasons.

CREW Case: Benchmarking Cognitive Networks

Context

During the past decades, an explosive emergence of wireless communication technologies and standards could be witnessed. Today, many people use multiple wireless communication devices on a daily basis, and expect an ever increasing wireless bandwidth. In addition, wireless communication is increasingly being used for machine-to-machine communication. The increasing number of wireless application and wireless devices goes hand in hand with a rising number of devices coexisting in the same environment, which compete for the same (scarce) spectral resources. Both in licensed and unlicensed radio frequency bands, the coexistence

issue gets increasingly problematic. Researchers investigating Cognitive Radio and Cognitive Networking protocols and algorithms, seek to resolve the coexistence problems by optimizing the use of the wireless spectrum, either by improving coordination between devices operating in the same frequency band, or by attempting to opportunistically use spectrum in other underutilized (licensed) bands.
While a large part of the research on wireless networks is still based on theoretical models and simulation results, there is also an increasing awareness in the wireless research community that experimentally-supported research may help to identify issues that cannot be discovered through theory or simulation only. This observation is reflected in the topics of international conferences that increasingly welcome experimentally-driven research and the interest of the European Commission in experimental facilities for future Internet research.

The main target of the CREW (Cognitive Radio Experimentation World) project is to establish an open federated test platform, which facilitates such experimentally-driven research in the field of advanced spectrum sensing, cognitive radio and cognitive networking strategies. The CREW platform is based on four existing wireless testbeds, augmented with state-of-the-art cognitive spectrum sensing platforms. The individual testbeds are based on different wireless technologies and include heterogeneous ISM, heterogeneous licensed, cellular, and wireless sensor components.

If the CREW platform is to become an established infrastructure where researchers are able to evaluate their cognitive solutions in an efficient and reliable way, providing a reproducible test environment is essential. Within the CREW work package on benchmarking, the goal is to design a benchmarking framework that will facilitate such a reproducible test environment. The platform will offer reliable performance indicators for cognitive radio solutions and will enable researchers to compare their solutions against existing solutions.

Example benchmark for cognitive networks

One of the problems with experimental validation in wireless networks in general, is that every environment and every device has its own characteristics, making it difficult to repeat experiments or compare results with results obtained by other researchers. Moreover, if different research teams measure the quality of their solutions in different set-ups (topology, background traffic, sources of interference), using different metrics or determine those metrics in a different way, the significance of experimental results may be low. Comparing the performance of wireless networking protocols in general or cognitive protocols more specifically is thus a challenging task. Therefore, creating a reproducible test environment for wireless networking is the primary goal of the benchmarking framework in the CREW project. One of the major CREW targets is to characterize the wireless environment and to analyse its influence on the SUT. By using benchmarking, a strict experimentation methodology is enforced that will increase the relevance of experimental results, while also speeding up the experimentation process.

As an example, one of the planned CREW use-cases evaluates a Zigbee-based sensor network (e.g. used for temperature monitoring) in an emulated home environment which also houses a

Wi-Fi access point and multiple Wi-Fi clients. The plan is to emulate a home environment inside an interference-free testbed building, by configuring a set of Wi-Fi based network nodes to transmit specific packet traces, which are modelled according to measured packet traces in a real home environment. In this emulated home environment, the Zigbee-based sensor network is evaluated using a *reliability benchmarking score*; a first time as a “standard” sensor network solution (*reference evaluation*), and next after augmenting the sensor network with cognitive interference-avoidance protocols (*novel cognitive solution - SUT*). In this cognitive case, the sensor devices use their built-in RF scanning functionality to determine the occupation degree of the different channels in the ISM band, and adjust their transmitting and receiving channels to optimize the coexistence with the Wi-Fi network.

During the experiment, advanced external sensing engines are used to collect spectral measurements spatially and temporally. These spectral measurements are used to monitor the characteristics of the wireless networks under evaluation and to detect potential unwanted interferers that are not part of the experiment. The benchmarking framework monitors whether the primary metrics are significantly influenced by external interference. If so, the experiment is not relevant and is therefore postponed or cancelled. During the benchmark both SUT and environment are evaluated using primary and secondary metrics. The benchmarking framework is further responsible for emulating the environment by replaying pre-modelled traces, so that it approximates the home environment as specified in the benchmark definition

The used benchmarking system for this use case maps entirely on the framework given in Figure 1:

- *Benchmark scenario:* a full description of the Zigbee nodes behaviour in the home environment, together with the home environment model (traffic models for WiFi access point and clients). It also contains the reference benchmark score and used primary and secondary metrics to evaluate the scenario.
- *Experiment config:* is responsible for executing the benchmark on the Zigbee nodes and configuring the Wi-Fi nodes (WiFi traces) so that they emulate the wireless home environment interference.
- *Metrics:* evaluate the Packet Error Rates (PER) and throughput on application level between the Zigbee nodes as the primary metrics. Spectrum information from the specialised sensing devices is gathered as the secondary metric. The feedback loop to the experiment configuration is used to evaluate the validity of the experiment results by comparing the measured environment to the modelled environment.
- *Benchmark score:* a single score distilled by inverting the PER and comparing the achieved throughput with the throughput of the reference benchmark.

The use of a benchmarking framework to realize the above plans leads to several interesting research tracks, to be tackled in the CREW project, including: *(i)* how to determine and create realistic and useful reference scenarios? This research question is not only to be solved for the case of experimenting with cognitive networks in the ISM bands, emulating home and office environments, but also for a wider range of cognitive radio cases in other (licensed) bands and in other wireless scenarios. *(ii)* How will reproducibility exactly be defined? When working with wireless networks and experimenting over the air, 100% exact repeatability is not feasible in every wireless testbed environment because of external interference. As a result, the “reproducibility” concept may have to be redefined, by setting acceptable margins in which results may vary. Even if perfect reproducibility is not realistic in a wireless experimental environment, it is very important to be able to characterize the wireless environment using advanced sensing techniques, and to analyse the impact of external interferers on the performance of the SUT *(iii)* Common data formats are to be defined to guarantee compatibility of benchmarks across multiple testing environments. Both input formats for a full experiment description as output formats for data gathered from experiments (e.g. spectrum sensing data) are to be defined. *(iv)* Definition of metrics and determining the methodologies to acquire the measurements.

OneLab2 Case: Benchmarking in Wireless Networks

Context

Benchmarking in wireless networks is of great interest to the networking research community because it enables fair comparison and can lead to peer-verifiable research. Fair comparison means systems are evaluated under reference conditions. However, it is non-trivial to ensure same spatial and temporal conditions. This is because wireless channels are unpredictable (random), error-prone and could vary over very short time scale (order of microseconds). Also it is impossible to ensure innocuous co-existence of collocated wireless networks. The lack of control over some aspects of the wireless experimentation has been a major hurdle in the way of comparable performance analysis. However, some of the network parameters can be configured to fixed values. We consider all the configurable network parameters to be controllable and the rest uncontrollable. Uncontrollable parameters include station workload (memory/cpu usage, etc.), network traffic load, multipath fading, path loss (attenuation), channel interference, etc. Controllable parameters entail scenario configurations (topology, traffic, wireless card configurations). Controllable parameters can also include meta-data such as system hardware/software specifications.

Simulation has been the pre-dominant technique for performance evaluation of networks. It provisions full control over the evaluation process with a certain degree of repeatability. However, the fidelity of underlying simplistic wireless propagation models is not up to the mark. Emulation improves the realism by combining aspects of real-world such as real hardware/software and simulations such as control and repeatability. However, emulation employs virtual/wired links instead of real wireless links and important channel characteristics such as path loss, packet errors, etc., are controlled and, therefore, not real.

Experiments in the real environment are the only way to provide wireless fidelity but complicate apples to apples comparison because channel conditions are highly volatile. Radio propagation characteristics such as path loss and multipath fading are highly dependent on the physical environment, geometric relationship between the objects, dielectric properties of the objects, movements of nodes/objects and the type and orientation of antennas used. Furthermore, interference from wireless-enabled devices operating in the 2.4 GHz band such as IPhones, IPads, game controllers, microwaves, cordless phones, etc., and exogenous WiFi networks is unavoidable. Wireless technologies are also evolving rapidly. Diversity enabled MIMO wireless cards and automatic power/rate adaptation schemes make it hard to study the impact of certain parameter(s) in isolation. In addition, commodity hardware and open-source wireless software such as drivers and sniffers may exhibit anomalous behavior and therefore require sanity checks and calibration.

Some of the pitfalls encountered in wireless experimentation are outlined here. These pitfalls also provide an insight into the challenges in the way of benchmarking in wireless networks. Channel characterization is a crucial first step but characteristics such as multipath fading are tightly coupled to the location, orientation of communication pairs and movement of objects in the environment. It is hard to ensure the desirable accuracy until and unless a machine-controlled mechanism is provisioned for the accurate positioning of nodes. One way to judge the similarity of two set ups in terms of distance and location, is to use path loss and Ricean K factor. But these estimates are influenced by noise floor calibrations and antenna diversity. They are easy to ignore by an unsuspecting experimenter. Empirical estimation of K factor and path loss can also be inaccurate if packet loss at the sniffers is not calibrated.

In order to overcome these challenges, we require a measurement (experimentation) infrastructure (tested), experiment control tools, data management and analysis tools (as envisaged in Figure 1) aided by a comprehensive benchmarking methodology. In OneLab2 [5], we worked on realizing fair comparison of networking protocols. As it is not feasible to repeat experiments, our methodology, testbed and tools facilitate running of large number of runs and then clustering of runs based on similarity of conditions (both controllable and uncontrollable parameters). We can, then, compare the performance for runs which belong to the same group in subsequent experimentation campaigns.

Wireless Channel Characterization

In wireless networks, channel characterization is a first natural step towards benchmarking. It enables researchers to investigate the influence of environment on wireless network performance and allows them to understand how well a protocol or an application performs in different wireless environments.

A mapping of our benchmarking system to the framework proposed in Figure 1 is as follows:

- Benchmark Scenario: It consists of a description of controllable and non-controllable parameters of both stations and the environment. This includes workload specifications, topology, and spectrum analyzer and sniffer settings.
- Experiment Configuration: It consists of scheduling and execution tools. The tools control the workflow of each experiment run and make it easier to repeat an experiment arbitrary number of times.

- Metrics: Evaluate the Packet Error Rate (PER) and Packet Loss Ratio at the MAC layer as the primary metrics. Spectrum information from the spectrum analyzer, Ricean K factor and SNR, Bit Error Rate (BER) as the secondary metrics.
- Benchmark Score: We use the packet loss as an indicator of performance and compare it with the packet loss of the reference benchmark. Packet error gives additional information about packet loss incurred because of CRC errors.

The above practice is well-supported by our experimentation methodology which has been tailored to the requirements of benchmarking in wireless networks. The methodology enables us to repeat an experiment fairly large number of times. In future, it might also be possible to repeat an experiment until the desired level of confidence is achieved for a given confidence interval. To enable comparability, we propose to cluster runs based on experiment conditions. Exploratory tests can be performed to investigate anomalies and perform calibrations. Basic sanity checks can be performed to verify accuracy of software/hardware tools. For example, performance of packet generator/sniffer, antenna diversity, time synchronization, etc. Empirical estimates of the metrics such as Ricean K factor are verified against the theoretical models.

Figure 4 demonstrates the variations in Ricean K factor caused by human traffic and its impact on the percentage packet loss. Test case 1 corresponds to the case when there is human traffic in the vicinity. On the contrary, there is no human traffic during Test case 2. Ricean K factor defines one the uncontrollable characteristics of the wireless channel. By clustering, we will be able to group experiments into clusters according to criteria such as high, medium and low human traffic. This will allow us to perform apples to apples comparison with future experiment campaigns. This is especially useful for a wireless experimentation campaign conducted over an extended period of time.

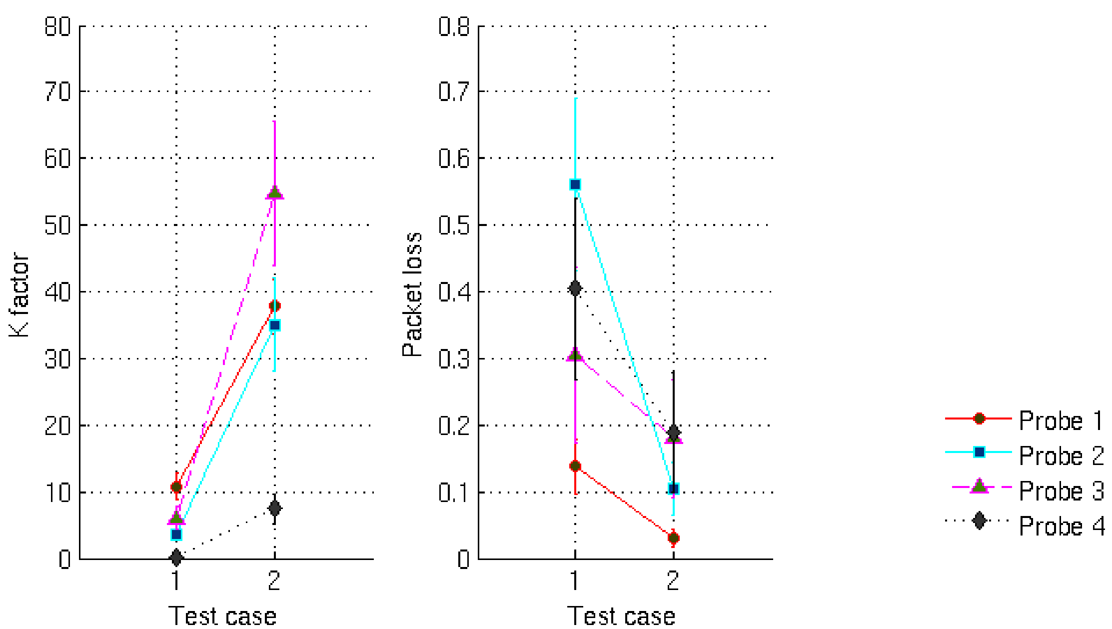

Figure 4: Ricean K factor and corresponding percentage packet loss as measured during two test cases. There is human traffic during test case 1 and no human traffic during test case 2.

Conclusions

While the benchmarking concept itself is not new, new challenges in research domains such as cloud computing or cognitive networking have renewed the interest of the research community in benchmarking, and multiple European FIRE research projects are putting effort in designing benchmarking solutions, or in closely related research topics such as measurements, performance evaluation, monitoring, or methodology design.

This paper clarifies what is meant by benchmarking computers and computer networks in the scope of some of these FIRE projects and indicates several benchmarking challenges still to be tackled using the examples of application modelling and cognitive radio benchmarks. It is not the ambition of this whitepaper to present a detailed survey of benchmarking or to fully detail the different benchmarking approaches followed in these projects. However, we hope that the presented terminology and high-level architecture can be a base for further discussions and may help to identify similarities and differences between the research done in different projects.

Acknowledgements

The research leading to this white paper has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreements n°258301 (CREW project), n°257386 (BonFIRE project) and n°224263 (OneLab2 project).

References

[1] P. Colella, Defining Software Requirements for Scientific Computing, 2004.
[2] K. Asanovic et al., "The Landscape of Parallel Computing Research: A View from Berkeley," Electrical Engineering and Computer Sciences, University of California at Berkeley, 2006.
K. Asanovic et al., "A View of the Parallel Computing Landscape," *Communications of the ACM*, vol. 52, no. 10, pp. 56-67, 2009.
[3] SPEC. (2011) Standard Performance Evaluation Corporation. [Online]. http://www.spec.org/index.html
[4] EC FP7-ICT IRMOS Project. [Online]. http://www.irmosproject.eu/
[5] EC FP7-WP9 Benchmarking. [Online]. http://www.onelab.eu/index.php