TRANSACTIONS ON EMERGING TELECOMMUNICATIONS TECHNOLOGIES
Trans. Emerging Tel. Tech. 0000; **00**:1–10
DOI: 10.1002/ett

RESEARCH ARTICLE

The path towards a cloud-aware mobile network protocol stack

Pablo Serrano[1],Marco Gramaglia[1], Dario Bega[1,2], David Gutierrez-Estevez[3], Gines Garcia-Aviles[1,2], and Albert Banchs[1,2]

[1] Universidad Carlos III de Madrid, Spain [2] Institute IMDEA Networks, Spain [3] Samsung Electronics R&D Institute UK

ABSTRACT

We are currently observing the softwarization of communication networks, where network functions are translated from monolithic pieces of equipment to programs running over a shared pool of computational, storage, and communication resources. While it is clear that almost any softwarization improves flexibility (e.g., the ability to instantiate more servers to cope with increasing traffic demand), in this paper we advocate for a complete re-design of the communications protocol stack, instead of a mere translation of hardware functions into software. We discuss two drivers for this cloud-aware re-design: (i) relaxing the tight interactions between functions, and (ii) supporting a graceful degradation of the service when resources become scarce. The potential benefits of this re-design are illustrated with the numerical evaluation of one use case.

*Correspondence

E-mail: {pablo,banchs,mgramagl}@it.uc3m.es; {dario.bega,gines.garcia}@imdea.org; d.estevez@samsung.com

1. INTRODUCTION

5G mobile networks will be characterized by a variety of services imposing a diversity of requirements.* To efficiently support this diversity, we need a change of paradigm in the provisioning of Network Functions (NFs), moving from the traditional vision where Physical Network Functions (PNFs) are tightly coupled with the hardware substrate running them, to the new vision where Vertical Network Functions (VNFs) run over instantiations of a general-purpose infrastructure. It is envisioned that this transition will introduce a tremendous improvement in flexibility, adaptability and reconfigurability, similar to the one that happened when transitioning from circuit-based to packet-based networking.

By *softwarizing* the operation of the network, VNFs (e.g., schedulers, databases, gateways) run as software components over a set of shared resources (antennas, links, servers, etc.), and can be dynamically provisioned as needed. As illustrated in Figure 1, a possible transition from the traditional vision to a fully softwarized network is the "Softwarization of Elements." That is, legacy PNFs are ported to a software implementation running in virtual containers.

This approach indeed improves the flexibility of the network: these monolithic programs run over shared computational resources, allowing, e.g., their re-instantiation on-demand, the reduction of development cycles and easier reconfiguration in general. Still, softwarization poses a number of challenges such as the efficient resource assignment to VNFs. This problem

*This has been repeatedly argued by now, in a number of position papers such as, e.g., [1], in SDOs such as 3rd Generation Partnership Project (3GPP) [2], and in industry fora such as Next Generation Mobile Networks Alliance (NGMN) [3]

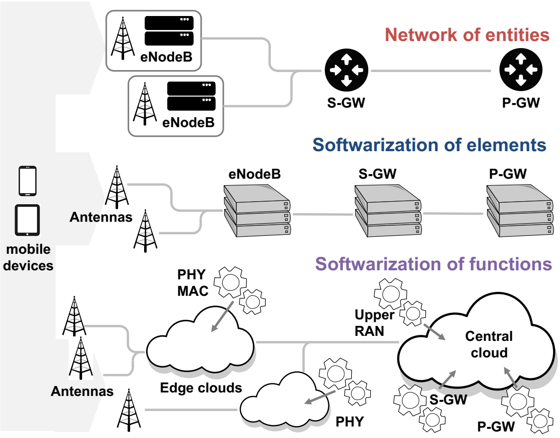

Figure 1. Different approaches to softwarization, from monolithic elements (top) to a completely *cloudified* network (bottom).

has traditionally been investigated from the network management perspective [4], but a very little effort has been done from the VNF design point of view. As VNFs are not designed to run over shared resources (that is, with virtualization in mind), there is the risk that any variation in these resources would cause service disruption: this potential *sensitivity* of VNFs to e.g. resource shortages, updates in the infrastructure, or container migrations, may preclude their wide use in future networks. Furthermore, given the indivisibility of these pieces of software, the assignment of programs to execution nodes (known as network embedding [5]) has been usually performed with a relatively coarse level of granularity, which hinders an efficient use of the resources.

A different approach to softwarization, which addresses this last issue, is depicted in Figure 1 as "Softwarization of Functions": monolithic VNFs are decomposed into smaller functions that can be instantiated and chained as required. This approach provides a finer granularity to the mobile network operation, allowing thus a better flexibility and therefore a more efficient use of resources. This transition from a *network of elements* to a *network of functions* further complicates network management, as the minimum requirements (e.g., in terms of delay and bandwidth) on the function chaining are increasingly heterogeneous. However, there is little when none research effort on reducing and simplifying these requirements (especially in terms of delay) by modifying the interactions between different functions.

As discussed above, softwarization has brought a notable amount of novelties that have an unprecedented impact on the way network management has to be performed. However, the current protocol stack and its composing blocks can be considered "legacies" of the past: the way in which physical network functions are designed has not changed much since early 3GPP Releases. Indeed, the network functions addressed by current softwarization efforts — e.g., Serving Gateway (S-GW), Packet Data Network Gateway (P-GW), or Radio Access Network (RAN) upper layers — have a practically direct mapping with the ones defined in 3GPP Release 8 [6, 7], which was published almost ten years ago.

For these reasons, we need a change of paradigm in the design of the network functions, to efficiently support all the novel features that network softwarization and cloudification bring. In other words, the time is ripe for a new class of *cloud-aware protocol stacks*, which embrace softwarization as the fundamental design criteria.

The current softwarization scenery

Recent trends in network softwarization have indeed considered the transition steps described in Fig 1. However, although the landscape of software implementations of networking stack has grown in the recent years on both OpenSource [8, 9] and Commercial [10] sides, all of them (especially RAN) are, to the best of our knowledge, a "standard" porting of the legacy functionality that allow its execution on general purpose hardware. Engineers and researchers working on these projects have indeed cared about the overall performance of the system, striving to reduce the resource usage footprint (mostly in terms of CPU and memory). Still they did not make any further consideration on improving the protocol stack to favor its execution as pure software. And while we only consider above current RAN softwarization efforts coming from the open source community or from small enterprise solutions, also the available products in the most important network equipment vendors do not consider the opportunities we discuss in this paper.

As a matter of fact, not considering a cloud-aware approach for the design of VNFs has direct implications on the proper dimensioning of the cloud. Most of the OpenSource solutions, for instance, require low latency kernels to correctly perform baseband processing, and rely on resource over-provisioning to avoid timing constraint violations that, in turn, result into frames dropped, poor user quality of experience, and even disconnections. Over-provisioning may be effective when dealing with laboratory deployments, but it results practically infeasible and extremely inefficient when real scenarios come into play.

Our Vision

We propose here to follow a new research line for the design of software mobile network architectures that aims way beyond the existing strategies such as the functional splits already proposed in different fora (e.g., by 3GPP [11] and the Small Cell Forum [12]). We believe that making the protocol stack *cloud-aware*, as we will describe in Section 2, requires significant changes to the existing functions or even the design of new ones.

We thus advocate for a complete re-design of the mobile network protocol stack with the goal of achieving a *cloud-aware protocol stack*. We identify two possible research lines that can follow this approach, detailing examples of how they can be applied in the context of a *cloudified* network. The practical implication of how these approaches may be holistically enforced in the network are outside the scope of this work, but we will illustrate in Section 3 with numerical evaluations that the potential benefits given by the proposed approach are very significant.

The rest of this paper is organized as follows: in Section 2, we discuss the two key criteria that should drive the re-design of the protocol stack, enabling a more efficient use of resources, and the need to quantify this cloud-awareness through key performance indicators; in Section 3, we illustrate via a number of use cases how to put these drivers into practice, including a numerical evaluation that assess the *graceful* operation of a function re-designed to run over a shared function; finally, we conclude the paper in Section 4.

2. THE QUEST FOR CLOUDIFICATION

We are convinced that the advantages brought by a cloud-driven VNFs design will fuel the research community. In fact, while researchers have devoted so far just a little attention to solve the problems involved by this approach, we believe that the relative maturity of current software initiatives (and the recent increase in the pace of their updates) provides the means for research in this area to bloom.

In this paper, we argue that future, fully softwarized and cloudified mobile networks will necessarily build on *cloud-aware protocol stacks*. We believe that both network management and the resulting overall performance will benefit from *making VNFs aware* of being executed in environments such as virtual machines or containers, running on shared resources. In this section, we discuss the main challenges to achieve this vision, while in the next section we describe three implementations of functionality that builds on this *cloud-awareness*, assessing how they will improve performance in a softwarized network both qualitative and quantitatively.

This approach entails two main challenges, namely *(i)* redefining the interactions between VNFs, relaxing as much as possible their temporal and logical connections, and *(ii)* support an elastic operation, to efficiently cope with changing input loads while running in an infrastructure of resources that is not over-provisioned. We detail the functional requirements of these novel design strategies in what follows, before discussing why they will also require the formal definition of novel Key Performance Indicators.

Given the high flexibility provided by the Network Function Virtualization (NFV) approach, the deployment of such *cloud-aware protocol stack* does not have a direct implication on the provided telecommunication service *per se*. The re-definition of the interactions among VNFs allows for a more flexible service orchestration, while the re-design of VNF internals may be easily provided by a code refactoring in a much faster way than the current tightly coupled HW-SW PNF approach. While having a *cloud-aware protocol stack* will benefit any kind of telecommunication service, this may be particularly relevant for the extreme ones. For example, a mission critical VNF can be optimized to reduce its memory footprint, while low latency services may exploit especially tailored orchestration patterns involving edge computing facilities, as we describe in Section 2.1.

2.1. Re-thinking network interactions

Future network architectures will heavily rely on the flexible function decomposition and allocation [13]. That is, the former monolithic PNFs are split into interconnected modules that, concatenated, provide the same functionality: e.g., a physical eNodeB is split into PHY, MAC, Radio Link Control (RLC) and Packet Data Convergence Protocol (PDCP) software implementations running in different execution containers, which can be located in different nodes of the cloudified network.

This approach (depicted in the bottom part of Figure 1) provides several advantages, as it allows heterogeneous deployments for different services (i.e., massive machine type communication, enhanced mobile broadband), which are tailored to their specific requirements. For example, depending on the latency, bandwidth, and/or computational requirements of the service, it may be better to locate certain VNFs towards the edge of the cloud rather than in a central location. How to place VNFs across the cloud is a network orchestration problem, which is constrained by the split into modules described above. However, this typical NF decomposition for the RAN protocol stack was not designed for its cloudification, and therefore the potential gains are limited. We next discuss this issue in more detail.

One key assumption of network stack designs is that certain functions are implemented in the same physical space (maybe on a different chip, but surely on the same hardware). So, non-ideal links with non-negligible delays are a problem for physical network elements that need to be decomposed into several network functions. Interfaces among them, thus, were designed considering communication links spanning some microns of silicon, and not several miles of fiber as in the case of, e.g., Cloud RAN (C-RAN) [14].

In this way, the possible inter-dependencies between these functions are overlooked, as the delivery of information between them is practically immediate. However, as we have argued above, to fully benefit from a network-wide orchestration of a cloudified stack, VNFs should support their execution on different nodes. But the design of traditional protocol stacks do not support such flexible placement of VNFs, as those with heavy inter-dependencies may introduce very high coordination overheads, or may not be even possible due to infeasible network requirements. These limitations severely constrain network orchestration, which compromises the overall gains obtained from the flexible function allocation. This is flagrant for e.g., the introduction of centralized RAN functions, where long delays in the information exchange between radio access points and the central cloud result in serious performance deterioration.

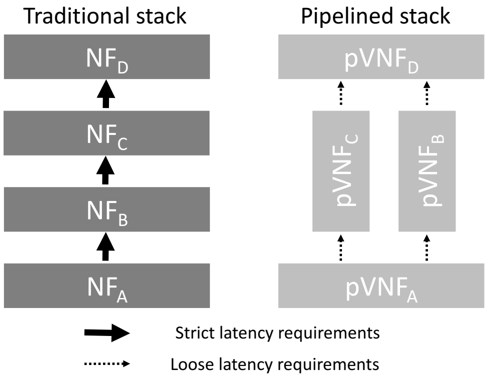

Figure 2. The transition from a traditional to a pipelined network protocol stack. Same subscript means that the same functionality is fulfilled.

Because of the above, the full protocol stack (and, in particular, the RAN) has to be re-designed with the goal of leveraging the benefits of the flexible function decomposition and allocation, so as to cope with non- ideal communication (i.e., non-zero and varying delay, limited throughput) between the nodes in the cloud. Specifically, a *cloud-aware protocol stack* should relax as much as possible, or even completely remove, the logical and temporal dependencies between VNFs, to enable their parallel execution and provide a higher flexibility in their placement. We refer to this approach as *pipelined network stack*, which is illustrated in Figure 2 (right), showing its differences as compared to the traditional stack (left): the challenge is to define new pipelined and virtual versions of the traditional NFs (marked as pVNF).

We discuss about the advantages provided by this approach by detailing a concrete example in Section 3.1, explaining the advantages that a *pipelined network stack* may achieve in terms of e.g., flexible VNF placement.

2.2. Re-designing VNFs internals

One of the most immediate and appealing advantages of a cloudified network is the possibility of reducing costs, by adapting and re-distributing resources following (and even anticipating) temporal and spatial traffic variations. However, it is also likely that in certain occasions the resource assignment across the cloud cannot cope with the existing traffic due to some peaks of resource demands. This is particularly true for C-RAN deployments, that have to deal with demand loads known to be highly variable [15]. In this scenario, allocating resources based on peak requirements would be highly inefficient, as this design jeopardizes multiplexing gains in particular when cloud resources may be scarce (e.g., a "flash crowd" at an edge cloud): here any temporal shortage might result in a system failure. VNFs, instead, shall efficiently use the resources they are assigned with. Thus, they have to become *elastic*, i.e., adapt their operation when temporal changes in the resources available occur, in the same way they have a long-established manner of dealing with outages such e.g. channel errors. Therefore, to fully exploit the benefits of softwarizing the network operation, the network function design has to take the potential scarcity into account, and be prepared to react accordingly.

In the context of wireless communications, the concept of elasticity usually refers to a graceful performance degradation when the spectrum becomes insufficient to serve all users. However, in the framework of a cloudified operation of mobile networks that has to deal with elasticity under resource shortages, we also need to consider other kinds of resources that are native to the cloud environment such as computational, memory, and storage assets available to the containers the VNFs are bound to. This has hardly been a problem for traditional network functions, that were designed to run over a given hardware substrate with exclusive access to the resources, and requires the definition of novel interfaces that will provide the amount and type of available cloud resources at a given point in time, just like, e.g., the accessible spectrum is a parameter for a RAN function.

Elasticity has also been considered by non-VNFs cloud operators, but our concept deviates very much from theirs: the time scales involved in RAN functions are significantly more stringent than the ones required by e.g., a Big Data platform or a web server back-end. Another key difference is that resources are way more scattered in our scenario (e.g. they are distributed across the "edge clouds"), which reduces the possibility of damping peaks by aggregating resources.

To better illustrate the benefits of elasticity in the cloudified mobile network operation context, we first consider the notion of "computational outage" [16], i.e., the unavailability of the required resources to perform the expected operation. In a traditional, non-elastic operation, there is a 1-to-1 mapping between outages and performance loss, as Figure 3 illustrates: if the resources are not available 20% of the time, there is a 20% performance degradation, as the function is unable to operate under any shortage. In contrast, an elastic design supports what we refer hereafter as *graceful* performance degradation, which causes that the VNF still functions under a resource shortage, this resulting in the "gains" illustrated in the Figure. Making a protocol stack *cloud-aware* through elastic VNFs requires hence a paradigm shift in their design, moving away from the tight hardware-software co-design that we discussed before, to a flexible operation in which the amount of available resources is an additional parameter.

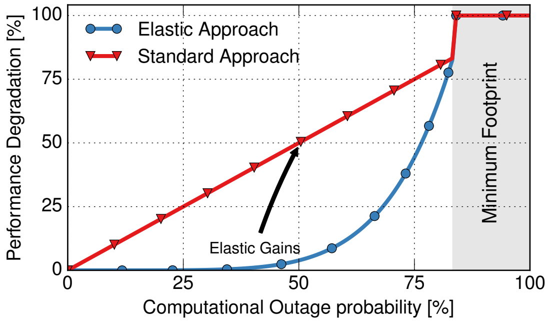

Figure 3. Graceful performance degradation achieved by elastic computation: performance is not degraded by the same relative amount as resources are reduced.

To fully take advantage of elastic VNFs, a detailed analysis of their operation is required: first, a thorough assessment of the resources consumed during execution, including statistics about temporal variations over time; second, a characterization of the correlations between VNFs operations, to serve as input for the orchestration algorithm, so it could e.g. dynamically assign resources to resilient VNFs and quickly "rescue" them when outages happen.[†]

We evaluate these discussion items in Sections 3.2 and 3.3 for two exemplary algorithms, providing numerical details about the performance gains under resource outages and the time dynamic metrics of the proposed elastic algorithms.

[†] Indeed, the quest for cloudification will end up with novel orchestration algorithms.

2.3. The need for performance indicators

Another consequence of a *cloud-aware* network stack is the need for a new framework to *(i)* quantify its behavior, *(ii)* assess its design and *(iii)* serve as input for the orchestration algorithm, which would decide where to place the VNFs depending on their elasticity. In this section we discuss about how to evaluate the benefits of the two strategies proposed above, especially focusing on the elastic VNF design, as it has more cross-fertilization opportunities.

The cloud computing community has indeed long worked on the definition of elasticity, this generally being defined as the ability to provision and de-provision resources to match the demand at each time instant as closely and efficiently as possible [17, 18].[‡] Building on this definition, it is possible to assess the degree of elasticity of e.g. a certain system, quantifying thus its ability to match the demand. However, as discussed before, the relatively coarse times scales of the traditional cloud operation prevents a direct mapping to our vision, where VNFs operate on much shorter ones. In the following, we provide some considerations on the required space of metrics to quantitatively characterize these new VNFs.

In general, the resources supporting the execution of a VNF are a heterogeneous set (e.g., CPU, RAM, spectrum, transport bandwidth), thus care should be taken when measuring them or varying their availability performing experiments, to perform fair comparisons. A VNF should be characterized by its *minimum footprint*, defined as the minimum combination of resources needed to provide any output. During its regular operation, the *footprint* is the set of percentages of time the resources are occupied because of the execution of a function.

Under ideal circumstances, i.e., no shortage or variation of the resources available, a cloud-aware VNF has to operate as reliably as a traditional network function. With this being the benchmark, we can define *reliability* as the % of time that a VNF is providing the expected output. However, while in the traditional approach this reliability referred to the availability of a communication resource (e.g. “five nines reliability”), in our vision there are more categories of resources that impact performance apart from congestion or link degradation.

Arguably, the most distinct feature of an elastic VNF is how it relates the above two points, i.e., the shape of the function that maps the available resources to the obtained outputs. This *degradation function* characterizes the way in which performance degrades as resources lack, and depending on its actual shape we could characterize different quantitative behaviors: e.g., a *graceful degradation* might be defined by a % decrement of resources causing the same or a smaller % of reduction in performance.

Additional aid in achieving resiliency can be obtained via orchestration mechanisms, that horizontally (up/down) or vertically (in/out) scale the containers executing the VNFs, but usually require a relatively long time scale to operate. For that reason, not all VNFs could easily “be rescued” in cases of low resource availability. The *rescuability* of a VNFs is hence its capability of overcoming an outage by providing a limited degradation, until new resources are available.

3. CLOUD-AWARE VNFs IN PRACTICE

In the previous sections we discussed the motivations for a thorough re-design of the network protocol stack involving both the interactions among VNFs and the VNF internals themselves. We next discuss some of the potential benefits of a cloud-driven re-design of the mobile architecture, thanks to the introduction of elasticity and the removal of rigid interaction between functions from a more practical / algorithmic perspective. To this aim, we revisit the tight dependencies within PHY and MAC layers functions that depend on each other and require close (both logical and temporal) synchronization for their operation. They also require non-negligible CPU resources for their operation (which, in the traditional approach, is 100% guaranteed by the tight integration with the hardware).

3.1. Pipelined HARQ

Traditional protocol stacks impose stringent temporal dependencies, such as e.g. for Hybrid Automatic Repeat Requests (HARQs). This is a lower layer protocol that requires the receiver to send immediate

[‡] Elasticity is a also related to resiliency, scalability, and efficiency, but with key differences: resiliency is the ability to recover from failures or to adjust easily to them, but it does not deal with efficiency; scalability is the ability to meet a larger load demand by adding a proportional amount of resources, but it does not consider temporal aspects of how fast and how often scaling actions can be performed. Finally, although a better elasticity should result in a higher efficiency, the opposite is not necessarily true.

feedback informing of the decoding success of a packet transmission. In the current stack, the time between the reception of a packet and the indication of the successful decoding is 4 ms, which forces the decoding function to be located very close to the radio interface and thus limits the possibility of centralizing such function.

One possible approach to loosen such tight synchronization could be inspired by [19] (we call it Pipelined HARQ, P-HARQ), where instead of performing the complete decoding of the frame and then sending the corresponding N/ACK, this feedback is based on the estimated channel conditions. In this way, the computationally-expensive decoding can be done in centralized and relatively far servers, while the MAC feedback, not completely accurate, can be provided by the now-simpler radio headers. Furthermore, this decoupling between functions not only supports placing the decoding relatively far away, but also reduces the performance guarantees (i.e., in terms of minimum bandwidth or maximum latency) to be provided by the link(s) connecting these functions.

The above is one particular partition of the HARQ functionality, which exchanges accuracy in the feedback, by using a simple decision making process, for flexibility in the network placement, which allows for moving the decoding away from the antenna. However, many other approaches may exist to implement HARQ, with different trade-offs in terms of accuracy, resource consumption and timing constrains.

The advantages provided by this approach on the orchestration side are manifold. This is depicted in Figure 4, in which two network slices share several antenna sites. By applying P-HARQ directly at the antenna sites, the orchestrator might place the decoding function of the non-latency critical network slices in a centralized location. Without P-HARQ, the higher and more variable delays experienced in a non-ideal fronthaul may prevent the correct operation of these network slices. The only countermeasure in this case would be the full orchestration of decoding functions directly at the antenna, increasing the cost and the rigidity of the network slices. Thus, specific deployments of a P-HARQ (enabled by relaxing the temporal dependencies in the stack) allow for a more efficient utilization of the cloud resources, enabling less expensive orchestration patterns to provide the same functionality.

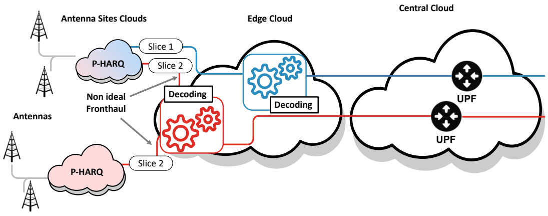

Figure 4. Opportunistic HARQ supports a more flexible orchestration of functions.

3.2. Modulation and Coding Scheme (MCS) selection for C-RAN

We now consider a C-RAN scenario where the scheduling of a number of base stations is done by a central entity. In a traditional approach, the VNF performing frame decoding has to be dimensioned for peak capacity, i.e., all Physical Resource Blockss (PRBs) using the highest MCS, which corresponds to ideal radio conditions for all users (having a set of users with good channel conditions is a common assumptions for schedulers that rely on Opportunistic Scheduling techniques [20]). However, planning for peak capacity not only requires prior knowledge of the users' demand, which is a difficult problem *per se*, but also results in resource wastage when mobile traffic falls below this peak.

Thus, let us assume a C-RAN scenario where resources are not over-provisioned, and the scheduling function for the uplink is serving a large enough set of base stations. Under these conditions, it will be likely that during short periods of time a set of users (experiencing good channel quality and hence using high MCS) require more capacity than available, as higher MCS require more iterations to be decoded [21]. A non-elastic function would fail to e.g. decode the PRBs, this resulting in an abrupt degradation of performance.

A cloud-aware MCS selection function helps to address this challenge more efficiently. Given that the "disruption" is caused by a relatively large set of users using high MCS, one elastic strategy is to purposely "downgrade" some of the MCS to be used in the PRBs, to find a combination that can be supported by the available computational resources. This version of the function, originally proposed in [22], might have better short-term fairness properties than the previous one, as users are still scheduled but at a lower rate. Also, this might support a more "graceful degradation" in the absence of resources. However, one key challenge is to find the most appropriate set of MCS to be used, as by definition the computational capacity is limited and therefore, the algorithm has to find the solution in short time.

3.3. Scheduling under resource constraints

As discussed in the previous section, running a non-elastic C-RAN scheduling function under drastic traffic variations (a typical condition nowadays) is challenging, as it results in abrupt disruptions when there are not enough resources to perform a computation. We next discuss another approach to introduce elasticity in the scheduling function, which assumes that the amount of computational resources available, i.e., the capacity, it is known. This capacity is denoted as C.

Like in the previous section, the scheduler starts by selecting those users as in a regular mode of operation, and checks if the amount of required resources to perform this decoding (i.e., iterations), denoted as $\tilde{C}$, is less than the capacity C. This estimation of $\tilde{C}$ can be done following e.g. the model introduced in [23], that provides the computational complexity as a function of the Signal-to-noise ratio (SNR) and the selected MCS. If $\tilde{C} < C$, it proceeds as a regular scheduler would do; otherwise, a different scheduling decision has to be taken (in the previous section, elasticity was achieved by downgrading all the MCS). So, another way to introduce elasticity in the scheduling function would be to use the following algorithm. Building on the widely-used Proportional Fair (PF) scheduling [24], the function selects the candidate set of users $\mathcal{U}$ that maximizes performance. Then, it follows these steps:

- *Step 1*: Compute the cost of scheduling $\mathcal{U}$, namely, $\tilde{C}$. If $\tilde{C} < C$, then $\mathcal{U}$ is scheduled and the algorithm stops.
- *Step 2*: Otherwise, discard the elements in $\mathcal{U}$ in increasing order of the PF metric (i.e., the lowest first), until $\tilde{C} < C$. Note that larger PF metrics are usually provided by higher MCS values or starving users.
- *Step 3*: Add to $\mathcal{U}$ those users that would maximize performance, but not accounting for those discarded in the previous step (that are kept in memory).
- *Step 4*: Repeat 1–3 until all base stations are scheduled.

Note that in Step 2, the algorithm discards some users with relatively high PF metric (i.e., the instantaneous rate over long term rate ratio), this resulting in a degradation of performance as compared with the default PF behavior in traditional networks. We next quantify how this degradation occurs as the capacity available C is reduced. To this aim, we simulate a C-RAN scenario with 5 Base Stations (BSs) serving 5 users each, where user i (with $i \in 1 \ldots 5$) is located at a distance $i \times d_0$, with d_0 being the distance at which the SNR is 17.6 dB. The wireless channel follows the Rayleigh model [25] with $\gamma = 3$ and $\sigma = 6$ dB, while the MCS complexity model follows the ones presented in [23].

We first evaluate the *degradation function* of this scheduling algorithm, as discussed in Section 2.3. To this aim, we first assume an unconstrained scenario to compute the reference performance and required capacity. Then, we reduce the capacity by a given *shortage*, and compute the resulting performance for the same PF scheduler (that fails to serve users if $\tilde{C} > C$) and our elastic scheduler. The resulting degradation in terms of performance are illustrated in Figure 5, for the "Standard PF" scheduler and the "Elastic PF" scheduler.

The traditional scheduler exhibits a proportional degradation of performance with the resource shortage, e.g., for a 20% reduction in resources, performance is reduced by approx. 20%. In contrast, the elastic scheduler provides a sub-linear degradation, thanks to the selection of a feasible set of users. We argue that this sub-linearity corresponds to *graceful degradation* of performance provided by the elasticity of this approach, which discards users that are likely to have good channel conditions (and hence higher MCS) to favor others with a better decoding efficiency (i.e., PF metric over complexity ratio).

We next evaluate the timings associated with the operation of the scheduling algorithm. To this aim, we initially assume the same scenario as above, with a 0 % shortage, and then we periodically add a new BS with 5 more users. We illustrate in Figure 6 the resulting performance degradation for the two scheduling functions, where dashed lines mark when a new BS is activated. As the figure illustrates, an elastic approach can prevent abrupt reductions in performance, this way facilitating *rescuability*, as the smooth degradation supports providing

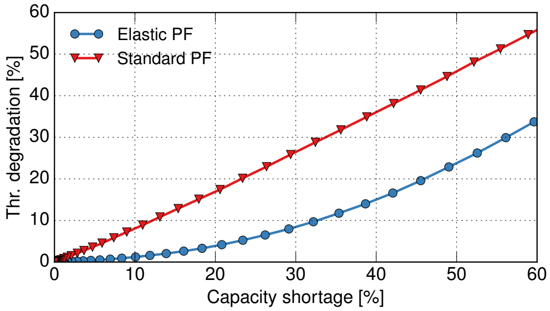

Figure 5. Graceful degradation obtained by a elastic scheduler vs. performance of a non cloud-aware standard scheduler.

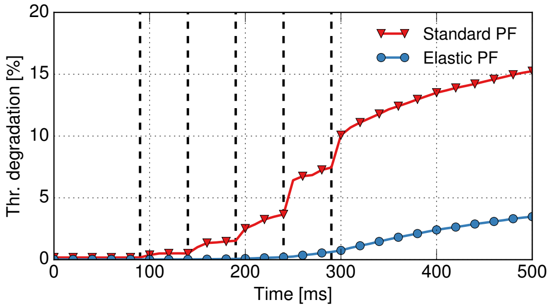

Figure 6. Performance degradation for a scenario with an increasing load.

more resources with a lower chance to incur into user Service-Level Agreement (SLA) violations.

Finally, we wanted to assess the performance of the elastic scheduler in a typical cloud deployment. The previous experiments are assuming that the resource outages happen on a permanent basis. In a real deployment, it is expected that the infrastructure provider counter this behaviour by providing more computational resources by either adding more bare metal to the cloud or by de-provisioning a less crowded ones. However, these specific actions entail orchestration operation that are likely to span tens of seconds or minutes [26]. Therefore, an elastic scheduler should avoid frame drops that happen with a much smaller time scales (typically in the order of hundreds of milliseconds [27]).

To evaluate the performance during transients, we consider the following scenario. We have a pool of 20 RAPs serving 5 users each,with only 25% of them active. The system is in a fairly low-load and the cloud capacity is dimensioned to correctly satisfy this load without any outage (i.e., a non elastic scheduler performs optimally in this case). At time $t = 10s$, we simulate a *flash-crowd* and all the RAPs suddenly become active. Now, the cloud capacity severely under-dimensioned is just 25% of the needed capacity. New resources are only orchestrated at time $t = 70s$, i.e., after one minute (in line with the boot times of new virtual machines reported in [26]). At that time, all the resources needed to serve all the user in the system are available.

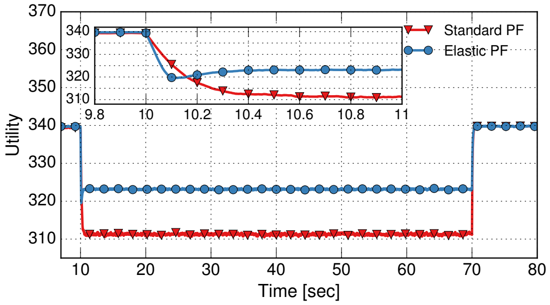

Figure 7. Graceful performance degradation in a transient resource shortage.

Figure 7 shows the utility, that corresponds to the proportional fairness logarithmic utility function [28], perceived by the user active since $t = 0s$ (i.e., the ones that are directly affected by the resource outage) when using a Standard PF scheduler and an Elastic PF scheduler. The elastic one keeps a considerable performance level throughout the experiment, avoiding dramatic breakdowns in the performance. When new resources become available, the performance returns to optimal level for both approaches.

4. CONCLUSIONS

While the softwarization of network functions will improve the flexibility of communications networks, in this paper we have made the case for a complete re-design of the protocol stack, and not a mere translation from monolithic pieces of hardware into software modules. We have discussed the need to re-think the interactions between functions, relaxing the associated requirements, and to re-design the functions' internals, to support a graceful degradation when resources become scarce. We have illustrated this need through three potential use cases, quantifying for one of them the remarkable performance improvements that could be attained. We believe that our results should influence ongoing initiatives in this area,

DOI: 10.1002/ett

when designing both the functions themselves and the frameworks orchestrating them.

REFERENCES

1. P. Rost *et al.*, “Cloud Technologies for Flexible 5G Radio Access Networks,” *IEEE Communications Magazine*, 2014.
2. 3GPP TR 22.891, “Feasibility Study on New Services and Markets Technology Enablers,” 2015.
3. NGMN Alliance, “5G white paper,” 2015. Available Online: `http://www.ngmn.org/5g-white-paper.html`
4. A. De La Oliva *et al.*, “Xhaul: toward an integrated fronthaul/backhaul architecture in 5G networks,” *IEEE Wireless Communications*, 2015.
5. A. Fischer *et al.*, “Virtual network embedding: A survey,” *IEEE Communications Surveys & Tutorials*, 2013.
6. E. Dahlman *et al.*, “3G evolution: HSPA and LTE for mobile broadband,” *Academic press*, 2010.
7. 3GPP TS 36.300, “Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description;” version 8, 2007.
8. N. Nikaein *et al.*, “OpenAirInterface: A Flexible Platform for 5G Research,” *ACM SIGCOMM Computer Communication Review*, 2014.
9. I. Gomez-Miguelez *et al.*, “srsLTE: An Open-Source Platform for LTE Evolution and Experimentation,” in *ACM WiNTECH*, 2016
10. Amarisoft, “Amari LTE 100 - Software LTE base station on PC,” Available Online: `http://www.amarisoft.com/index.php?p=amarilte`
11. 3GPP TR 38.801, “Study on New Radio Access Technology: Radio Access Architecture and Interfaces,” 2016.
12. Small Cell Forum, “Small cell virtualization: Functional splits and use cases,” *White Paper*, Release 6.0, 2016.
13. D. Sabella *et al.*, “Benefits and challenges of cloud technologies for 5g architecture,” in *5GArch Workshop*, 2015.
14. China Mobile Research Institute, “C-RAN: The Road Towards Green RAN,” *White Paper*, 2011
15. A. Checko *et al.*, “Cloud RAN for Mobile Networks: A Technology Overview,” *IEEE Communications surveys & tutorials*, 2015.
16. M. Valenti and P. Rost, “The role of computational outage in dense cloud-based centralized radio access networks,” in *IEEE GLOBECOM*, 2014.
17. E. F. Coutinho *et al.*, “Elasticity in cloud computing: a survey,” *Annals of Telecommunications*, 2015.
18. N. R. Herbst *et al.*, “Elasticity in Cloud Computing: What It Is, and What It Is Not,” in *IEEE ICAC*, 2013.
19. P. Rost and A. Prasad, “Opportunistic Hybrid ARQ – enabler of centralized-ran over non-ideal backhaul,” *IEEE Wireless Communications Letters*, vol. 3, no. 5, October 2014.
20. A. Asadi and V.Mancuso, “A survey on opportunistic scheduling in wireless communications,” *IEEE Communications surveys & tutorials*, 2013.
21. S. Bhaumik *et al.*, “CloudIQ: A framework for processing base stations in a data center,” in *ACM MOBICOM*, 2012.
22. P. Rost *et al.*, “Computationally aware sum-rate optimal scheduling for centralized radio access networks,” in *IEEE GLOBECOM*, 2015.
23. P. Rost *et al.*, “The complexity-rate tradeoff of centralized radio access networks,” *IEEE Transactions on Wireless Communications*, 2015.
24. A. Stolyar, “On the Asymptotic Optimality of the Gradient Scheduling Algorithm for Multiuser Throughput Allocation,” *Opearations Research*, 2005.
25. T. Rappaport, “Wireless Communications: Principles & Practices,” *Prentice Hall*, 2011.
26. T. L. Nguyen and A. Lebre, “Virtual machine boot time model,” in *Proc. of 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2017.
27. 3GPP, “Policy and charging control architecture,” TS 23.203, 2017.
28. T. Bu, E. L. Li, and R. Ramjee, “Generalized Proportional Fair Scheduling in Third Generation Wireless Data Networks,” in *Proc. of IEEE INFOCOM*, 2006.

DOI: 10.1002/ett