Experimentation As an ISP-Service

Zaineb TAHRI, Ramzi OUAFI, Mohamed Karim SBAI
Esprit School of Engineering,
Tunis, Tunisia
Email: name.surname@esprit.tn

***Abstract*—Experimenting innovative ideas requires deploying large scale heterogeneous testbeds. We propose to involve ISPs in experimentation to boost creativity, flexibility and realism of experiments. Using new network paradigms, ISPs allocate virtualized resources to experiments and benchmark testing conditions. The level of isolation between experimentation and production traffics must be gauged.**

I. Introduction

The success story of the Internet can be owed to its openness to innovation and creativity. Indeed, this network of networks supports a wide spectrum of innovative services and applications. Users are, not only, free to publish and retrieve any type of content and media, but also, can enjoy a panoply of services offered by Internet Service Providers (ISPs). Therefore, when designing architectures and services for the Internet of the future one must preserve this polyvalence of the current Internet, and go one step further to yield more innovation and imagination.

Usually, research scientists and developers of inventive ideas in their innovation process have to evaluate their achievements (algorithms, protocols, services, etc) in a realistic environment such as a network or a set of machines connected to a network. Besides, service providers (CDNs, Clouds, etc) needs to monitor the relevance of their services and measure the level of satisfaction of their clients. To achieve these goals, traditionnaly, they have to choose among the following candidate evaluation environments: simulation tools, emulation, and testbeds. Although latter testing environments succeed to meet some of the requirements of their users; users still have to combine one or two of these methods to achieve the evaluation goals.

Nevertheless, interesting efforts have been spent by the community to ameliorate the performance evaluation platforms. A particular attention has been made to the flexibility of the testing scenarios. Indeed, many testbeds consider heterogeneous technologies and offer a diversity of resources for experimentation. They differ from each other by their level of programmability and the scale of the experiments they can host. The realism of the experimentation depends a lot on the benchmarking of the testing conditions and relies on the coexistence of many factors such as exogenous traffic, available computing and network resources and specific devices (sensors, mobile handhelds, etc.). In meanwhile, many approaches propose to federate testbeds to obtain wider ones with more heterogeneity, more flexibility and a higher availability of testing resources. Hence, an enhanced realism and a larger scale of experiments can be ensured. However, we show, in this paper, that involving ISPs in the experimentation chain improves considerably creativity, flexibility, and realism of experiments. An ISP network is the best field to host inventive experiments of the future. In fact, from one side, an ISP owns the largest amount of resources (computing power, storage capacity, network links, etc.). Due to the diversification and the size of activities of an ISP, the resources that it can offer for experimentation are sufficiently heterogeneous in their technologies and their functionalities. From another side, new paradigms for network architecture and operations have been invented recently. Software Defined Networking (SDN)[1][2] and Network Functions Virtualization (NFV) [3][4] make effortless network deep-programmability and improve network management flexibility. Profiting from these advances, ISPs can allocate virtualized resources to experimentation and manage the level of isolation between experimentation and production traffics. We propose, in this work, that an ISP offers experimentation as a service. It receives the specification of an experiment as a set of required testing resources and experimentation traffic to handle. The role of the ISP is to map the demand of the experimenter to slices of physical resources and to schedule experimentation traffic flows. The ISP must ensure both network conditions required for the experiment and the friendliness of experimentation traffic with the production traffic. The presence of exogenous production traffic, however, increases in some scenarios the realism of the experiments. We design our service as an SDN control application that ensures scheduling of experimentation flows, their routing and monitoring of testing conditions. We pay attention to the realism of the experiment and gauge the level of isolation between experimentation and production traffics. To validate our idea, we implement it in the NS-3 network simulator [5] and evaluate its performance for different specifications of experiments and tunings of production traffic. The remainder of this paper is organized as follows. In Section II, we present both technological and functional motivations behind proposing experimentation as an ISP service. Then, in sections III, and IV, we detail respectively the requirements and the design of our experimentation service. Finally, and before thinking of perspectives of our work, we show in V simulation results that validate our proposal.

II. Motivation and context

The networking community has invented a variety of performance evaluation testbeds with different levels of heterogeneity and complexity. An experimenter can choose among a set of standalone testbeds or bigger heterogeneous federation of testbeds. In this paragraph, we study the limitation of these testbeds in the perspective to prove the necessity of involving ISPs in the experimentation chain. Furthermore, we overview the technological advances on which we base the architecture of the ISP experimentation service.

A. Towards realistic performance evaluation platforms

Considerable efforts have been made recently to improve performance evaluation platforms. We show, however, that without the implication of network operators. The experiments conducted, mainly in case of new challenging test scenarios, cannot reach an interesting scale or level of quality.

1) Limits of current Testbeds: The research community can, nowadays, use many testbeds to evaluate the performance of their solutions. PlanetLab [6] and ORBIT [7] are two examples of these testbeds that give an open access to experimentation resources to experimenters and researchers. The testbeds differ from each other mainly with the strategy used for reserving and sharing resources among the set of experimentations and with the network technologies and testing services they offer. For instance, PlanetLab uses the slice concept in allocating network resources to its users and hence provides a widely available leasing service for networked-computing resources. A testbed using the concept of *sliceability* must maintain some degree of isolation for simultaneous experiments. Indeed, there is a shared simultaneous access to shared physical resources. The performance of the testbed depends strongly on the level of isolation and has generally been overlooked in the literature. A study of the impact of sharing test resources on the benchmarking of the testbed is one of the objectives of this paper. Sometimes a complete isolation of experimentation resources is not recommended by the experiment for instance testing with the presence of exogenous traffic can be one of the objectives of the experimentation. The presence of production traffic in the network paths or concurrence to access a storage or computing resource can be required for the realism of the experimentation.
When technologies are considered, each testbed offers to experimenters a set of resources such as PCs, routers, switches, wireless spectrum, etc. The offered technologies differ from one testbed to another. Some of them, like PlanetLab, focus on interconnecting machines located at different geographic locations through the Internet whereas others, like ORBIT, concentrate on providing the possibility to do wireless experimentations such as wireless ad hoc networking, 3G/4G mobile networking, etc. The available technologies and their level of heterogeneity give an idea of the spectrum of experiments that can be conducted on the testbed. Some experimentations need specific components such as sensors, specific mobile devices, specific wireless bands, etc. One can then think that one single testbed cannot meet the requirements of all experimenters. Moreover, the scale of the testbed in terms of the number of available resources and how they are allocated to users impacts the scale of experiments that can be conducted on the testbed.

2) Approaches for federating Testbeds: In the last years, the community has proposed to put together the resources of many testbeds in wider federated testbeds. The objective is to create synergy among testbed holders and foster efforts to design and develop common experimentation techniques and tools. Hence, they opt for increasing the deep programmability of testbeds. Deep programmability refers to the ability of an experimenter to influence the behavior of computing, storage, routing, and forwarding components deep inside the network, not just at or near the network edge. These efforts are still in their first steps and need more consideration from the community. However, federating approaches such as GENI [8] and FIRE [9] offer a slice-based federation architecture of heterogeneous network and computer technologies. They offer a layer-two overlay over resources. They aim to offer experimenter-specified topologies via a programmable layer-two network. Hence, they opt to scale-up the experimentation possibilities, to increase the flexibility of experimentation scenarios, and to facilitate the experimentation resources management and control.
Nevertheless, these approaches of federating testbeds cannot offer the same network resources as owned by ISPs and cannot provide as many heterogeneous technologies as existing in a real world production network. Furthermore, the network conditions present in an ISP network such as concurrent production traffic and real physical resources cannot be fulfilled by a federation of existing testbeds. Hence, the realism of the experiments in such a federated testbed cannot be improved without implicating ISPs in the experimentation process.

B. Technical considerations

Recent technological evolutions changed the way network functionalities are implemented and the way management and administration of networks are ensured. Software Defined Networking (SDN) and Network Function Virtualization (NFV) encouraged us to propose an experimentation service supported by ISPs.

1) SDN-based ISP network management: The introduction of the SDN paradigm inside IP-based ISP backbones is, nowadays, proposed by the networking community [10]. They propose the coexistence of regular IP forwarding/routing and SDN-based forwarding for different types of advanced services (VPNs, Virtual Leased Lines, Traffic engineering) on the same network nodes. In this paragraph, we describe such an hybrid IP/SDN networking scenario from the point of view of the possible services and the network architecture to be implemented.
In today's packet-switched networks, housing control and data functions in the same box necessarily complicates the equipment, which aside from making routing decisions and switching packets. It also needs to have all the necessary intelligence for aggregating and disseminating routing information, using fully distributed routing protocols. Furthermore, in IP/MPLS networks, another layer of complexity is added with the need for distributed signalling and label distribution mechanisms. All of these features contribute to control plane load, increased fragility and increased cost.
In the SDN-based ISP architecture, the community proposes the use of MPLS forwarding elements as data plane switches that connect to a logically centralized Controller using the *OpenFlow* protocol [11][12]. The Controller runs a network-wide OS and a set of network control applications to realize all the functionalities of existing intra-domain protocols. Moreover, the *OpenFlow* protocol allows discovering the network topology, and keeping network-state updated via statistics, status and error messages. It also provides mechanisms to manipulate the flow tables of data plane switches. The network-OS based applications can then take network control decisions, can classify traffic into flows, and can pilot flow-level switching and forwarding. The logically-centralized

nature of the Controller implies that while the decision making is performed in a centralized manner, the Controller itself can be distributed over multiple physical servers for fault-tolerance and performance.
The underlay just becomes a set of common IP circuits with next hop reachability. This opens the door for customers to go directly to the local market to procure more cost-effective bandwidth with the right mix of transport technologies and SLAs (Service Level Agreement) required for the business, without compromising or fragmenting the logical routing topology. What if a network administrator could build an underlying network with various transport providers and glue the transport together with a unified overlay providing centralized policy management via a Controller to create logical segmentation for multi-tenancy? Essentially, this would drive up the efficiency rate, creating a more cost-effective network. Such a flexible and efficient ISP network can be the best field to support an experimentation service.

2) Network Function Virtualization and middleboxes: Processing inside the network: In the ISP networks of the future, middleboxes offering a panoply of processing abilities are deployed in the core of the network. Thanks to the Network Function Virtualization (NFV) philosophy, these middleboxes can play many roles for the ISP. Indeed, they can ensure a wide spectrum of functionalities ranging from Traffic engineering tasks to service-aware processing of flows (transcoding, caching, etc.).
Having this in mind, one can expect that ISPs can profit from these in-network processing services as a building block of an experimentation service. These resources can be used as experimentation traffic generators, as experimentation monitoring vantages, etc.

III. Requirements of the experimentation service

An ideal experimentation platform must be composed of a big number of machines/equipments and real production traffic. Such platform cannot be obtained without involving the ISPs. The goal is that they offer experimentation as a service. In this paragraph, we discuss the requirements of this ISP-service.

A. Flexibility of the testing scenario

The experimentation service must offer to its users: the experimenters, a set of network equipments (switches, routers, etc.) and communication links with different characteristics. The ISPs have then to be able to allocate these network resources to an experimentation. The scale of this experimentation and the equipments dedicated to it should mainly depend on the specifications of the experimenter. However, for security reasons, ISPs cannot provide direct access to their machines. They can rather offer a virtualization of their resources to the experimenter. Indeed, the ISP must receive from the experimenter an experimentation scenario which describes the models of the needed equipments (their characteristics) and have to map this description to a well-selected subset of its physical equipments. This procurement of an abstract model of the testing topology is a key factor of the flexibility of the testing scenario. The ISP takes over the translation of the abstract model to a sub-physical topology. In this paper, we overlook the procurement translation process and we confuse virtual topology to the physical one. Hence, in our study, all the equipments can be involved in an experimentation. However, to define an experimentation, the description must include not only the model of the topology but also a description of the traffic and the network conditions it must encounter. An experimentation is a set of flows, each flow has a couple of source and destination nodes, a type of traffic, a specific network load and network conditions to undergo on the path. A chronological order of the flows can be imposed by the experimenter.

B. Realism and correctness of the testing conditions

An experimentation, being described as a set of flows, the experimentation service must distinguish between these flows and offer them the expected network conditions. Moreover, it has to schedule flows both in time and space. When time is considered, some flows are to be launched before others and someother can be launched simultaneously. The logical realism of an experimentation depends on this sequencing and parallelism of flows. Nevertheless, the temporal scheduling is not without constraints on network conditions to offer to flows. In fact, the experimenter indicates the QoS required on the paths of the flows. The experimentation service must monitor the network paths to opportunistically allocate them to experimentation flows. This opportunistic allocation must occur in a bounded time limit in order to decrease the waiting time of the experimenter. After launching a flow, the service must verify whether it has obtained the desired network conditions. This validation phase is out of the scope of this paper. But, one can imagine that flows that have not encountered the required conditions, can be reconsidered by the scheduler and then can be relaunched. In the case of dependent flows, it can imply relaunching the whole set of flows of an experimentation or a sub-dependent set of them.

C. Friendliness with production traffic

Allocating network resources to experimentation traffic, the ISP risks impeding production traffic. The latter traffic represents the main activity of the operator and must be ensured in normal conditions. The scheduling of the production traffic ought to take into consideration the existence of this crucial traffic. A first solution would be to dedicate separate physical resources to each type of traffic. This does not only limit the resources available for experimentation but it also decreases their realism. We suppose rather that communication links are shared between both experimentation and production traffics. The experimentation service has to guarantee the non-aggressiveness of the experimentation traffic. For instance, if the bandwidth required for an experimentation flow exceeds the available bandwidth on the link, the scheduler would better postpone launching the flows to an appropriate time.

IV. Design of the experimentation service

In this section, we detail the design of our SDN-based experimentation service.

A. Global architecture

The design of an experimentation service that fulfills the requirements described in Section III needs a particular

organization of the ISP network. The latter network should be characterized by a centralization of the knowledge about equipments and topology mechanisms and routing/communication decisions. The network must be able to distinguish among flows and provide them with differential processing. In this perspective, we choose that the ISP network architecture must follow the Software Defined Networking (SDN)paradigm.
In an SDN network, the control plane is decoupled from the data plane. Decisions are taken by a central entity called *Controller*. The latter communicates with the interconnection equipments to impose processing rules of flows and to collect measurements on the traffic. This communication is ensured on an outband channel through a specific API.
In our study, we suppose that switches are administrable by *OpenFlow* [12]. The different policies of the ISP are implemented in the control plane thanks to control applications. It allows the Controller to specify rules that will be implemented in the switches. In this work, we design an experimentation service dedicated to ISPs networks. If the ISP architecture follows the SDN philosophy, it can on one side, support the ordinary production flows and existing protocols; on the other side, it can support the experimentation flows. Each category of traffic will be managed by a specific business application. In particular, our experimentation solution will then be designed as a business application. Figure 1 shows the location of our service inside the global SDN architecture.

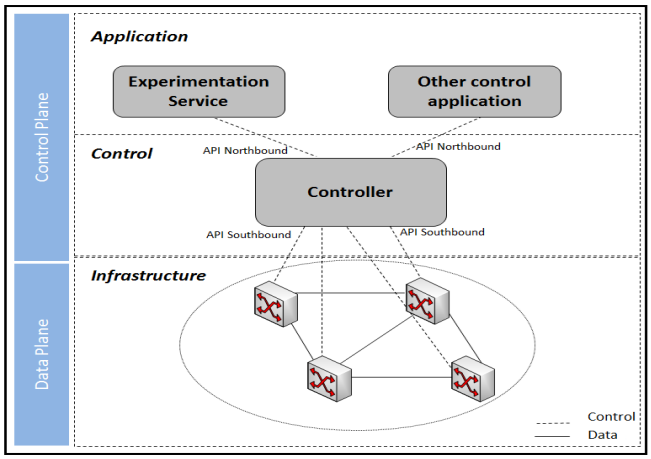

Fig. 1: Global architecture

B. Components of the experimentation service

Our application *Experimentation-As-A-service* contains itself a set of components which communicate together to result into the global service. Figure 2 represents the detailed design of our experimentation business application. In this application, the component *Monitoring* has the role of discovering the network topology and collecting statistics on communication links from the "Controller". This latter sets up the essential rules to collect this information.
Once obtaining the measurements on network conditions of the links, the component *Routing* can decide on the path that should borrow different experimentation flows following constraints imposed by another component called *opportunistic Scheduler*. The latter has the role of managing the experimentation flows and to launch them when the routing can assure the desired conditions. The effective launching of flows is obtained thanks to a communication with the *Controller* who sets up the necessary rules at switches located on the path.

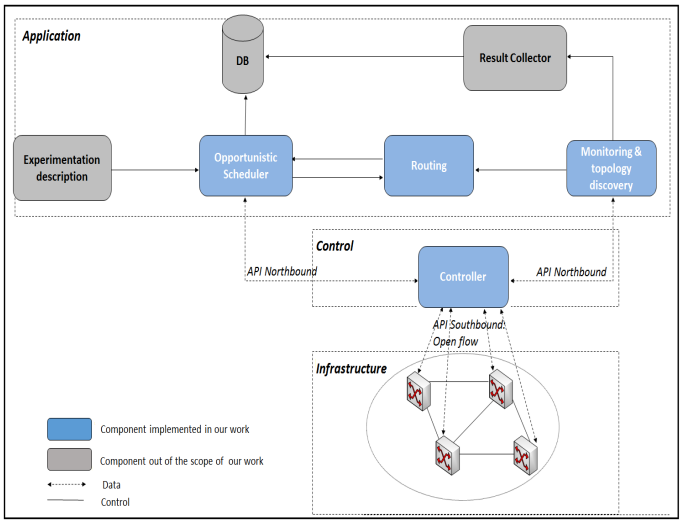

Fig. 2: Detailed design of the experimentation service

1) Monitoring component: The Controller owns a centralized vision of the current ISP-network. The Monitoring component retrieves this topology from the Controller. Then, it selects communication links to monitor and equipments to be involved in the collection of statistics. It dresses a monitoring plan of the network. After that, it asks periodically the Controller to probe the appropriate equipments. The choice of the probing time period is determinant for the freshness of information used in routing and scheduling. However, to avoid overloading the network with control traffic, it is mandatory to avoid using very short probing periods.
For instance, after obtaining statistics on the ports of the concerned switches from the Controller, the Monitoring component compute a per-link available bandwidth. Noting $A = C - U$, where C is the capacity of the link and U is the used bandwidth. Noting Δt the period of probing $U = \frac{B_{\Delta t}}{\Delta t}$. $B_{\Delta t}$ is the amount of data sent on the link during Δt. As *OpenFlow* primitives allow to retrieve, the cumulated amount of data sent or received on the link at instant t, we take $B_{\Delta t} = B(t + \Delta t) - B(t)$. Hence, the Monitoring component is able to compute the available bandwidth on all links of the topology.

2) Experimentation Flow Routing component: The role of the Routing component is to find out routes that responds better to criteria imposed by the Opportunistic Scheduler component which needs to find a path from a source to a specific destination that verifies well-defined QoS constraints. We suppose that decisions are independent and that the search algorithm uses a unique criterion. A constraint is imposed for a specific metric on the path.
The problem of searching the adequate path for a flow can then be formalized as following:
The ISP-network can be represented as a graph $G(S, L)$, where S is the set of switches and L is the set of links between them. This graph is then an oriented graph as values of the metrics

on a link depend on the considered direction. Each link is weighted by the current measurement of the considered metric. The measurement function M affect to each link $L_i[S_i, S_{i,t}]$ a measurement $M(L)$.
When an end-to-end path P is considered, the end-to-end measurement on this path is taken as an aggregation of measurement on the links $M(P) = agg(M(L_i))$
For example, when the available bandwidth is considered, the aggregation is simply the minimum of the available bandwidths.
A constraint on the path P for the metric M, imposed by the scheduler can be expressed like:

- Constraint with an upper bound: $M(P) \leq M_{max}$
- Constraint with a lower bound: $M_{min} \leq M(P)$
- Constraint with an approximate value : $M(P) = \mu \pm \beta$

Where μ is the expected value and β is a fault tolerance parameter.
Searching a path that respects one of these constraints is an NP-hard problem. The search space increases exponentially with the number of links in the graph. In our work, we propose using heuristics to simplify the search problem and make the search algorithm converge rapidly to adequate paths. These algorithms are widely discussed in the graph theory literature [13]. In our work simulations presented in Section V, we consider some simple search algorithms that reduce the search space by considering greedy search heuristics. It considers a spanning tree on the graph taking the source host as a root. At each iteration, it eliminates some branches of the tree following the estimation of the metric on the remaining part of the path. The Routing component informs the scheduler with the best possible paths for the experimentation flows to launch.

3) Opportunistic scheduler of experimentation flows: The scheduler component starts from a description of an experimentation as a set of flows. Each of these flows imposes a QoS constraint. In addition to these constraints, a chronological order for launching flows is wished by the experimenter. A chronological constraint is described as a temporal dependency. One says that a flow f_j depends temporally on a flow f_i if and only if one can launch flow f_j only when the flow f_i has already been launched and transmitted in good conditions. Hence, the scheduler takes as an input a graph of temporal dependencies as illustrated in Figure 3. At the beginning of the

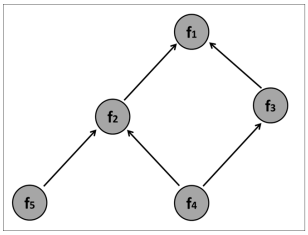

Fig. 3: Temporal dependencies

execution of the scheduling algorithm, the only flows that can be launched are those who do not depend temporally on any other flows. In figure 3, only the flow f_1 can be added to the list of the flows that the sheduler can launch in the beginning of the experimentation. When the flow f_1 is launched, f_2 and f_3 can be added to the list of flows to launch.

We describe in Figure 5, the pseudo-algorithm that manages the list of flows to launch. Here is the description of some functions of this pseudo-algorithm.

```
Algorithm Scheduling_of_experimentation_flows
var
        L:List of flows to launch
        G_d:Graph of temporal dependencies
        F:Set of launched flows

Begin
        L=Independant(G_d)

While (G_d ≠ ∅) do
        F = Launch_Flow (L)
        G_j = Eliminate(F, G_d)
        L = (L \ F) U Independant(G_d)
done
End
```

Fig. 4: Pseudo-algorithm of the scheduler

- Independent (G_d) : This function returns the set of flows of G_d that do not depend on any other flow.
- Launch-Flow (L) : This function launches flows of L whose required network conditions can be fulfilled by the Routing component. The list of flows effectively launched is returned in the list F.
- Eliminate (F,G_d): eliminates from the graph G_d the launched flows.

The previously described algorithm ensures that experimentation flows encounter the desired network conditions and do not disturb the production traffic.

V. Performance evaluation

To evaluate the performance of our experimentation service, enhance the existing NS-3 OpenFlow model to support a complex SDN-based ISP network (topology discovery, network monitoring, route computing and equipment management). Furthermore, we implement in our NS-3 model the experimentation scheduling component described in Paragraph IV-B3.

A. Simulation scenarios

In this paragraph, we describe the simulated system which we use in evaluating the performance of our experimentation service idea. We present the considered network topology, the simulated production and experimentation traffic and the tuning of the experimentation service.

1) Network Topology: We simulate an SDN network with a number of nodes that can vary from one simulation to another. The topology of the network follows an hierarchical logic. In such a network, one can distinguish two main zones:

- *Core network*: It is totally meshed network of switches allowing the connection of distribution networks together.
- *Distribution network*: A distribution network is connected to two border switches of the core network. This redundancy is interesting since it allows fault-tolerance and gives us the opportunity to simulate multiple-trajects scenarios. The other switches of a distribution network allow direct access to host nodes.

A sample configuration of topology parameters is given in Table I. These parameters are those considered for simulation results discussed in Paragraph V-B.

Settings	Values
Number of switches in the core network	4
Number of switches per distribution network	4
Number of hosts connected to a switch of a distribution network	2
Capacity of links	10 Mbps

TABLE I: Simulation settings

2) Simulated traffic: Before starting an experimentation, we need to load the network with some production traffic and then specify the experimentation flows to schedule.

- *Production traffic*: In each simulation, we consider a number of production flows (N_p). For each production flow, we select a random couple of hosts as the source and the destination, use the UDP transport protocol, and consider a flow rate of $1Mbps$.
- *Experimentation flows*: We precise the number of experimentation flows (N_e) and the ratio of temporal dependency between them. The graph of temporal dependencies of flows is generated randomly and respects the dependency ratio. The sources and destination of the experimentation flows are selected randomly among the sets of hosts of the network. Each experimentation flow is a $1Mbps$ UDP connection.

Hence, the density of experimentation flows, in a simulation, is computed as follows: $d = \frac{N_e}{N_e+N_p}$

B. Simulation results analysis

In this paragraph, we describe the results of simulations conducted following the scenarios mentioned in the previous paragraph. The goal is to validate the experimentation service in regard of various performance criteria.

1) Usage of network capacity: As a first step in our analysis, we consider an experimentation service which launches an experimentation flow only if the available bandwidth on the shortest path is sufficient to support the rate of the flow. Hence, an experimentation flow must wait for sustainable network conditions before being granted network capacity. Although this scheduling policy ensure the performance level required by the experimenter, it does not exploit the total available bandwidth in the network. In fact, flows are not allowed to follow alternative paths. Figure 5 shows that even if one increases the number of experimentation flows, the usage of network capacity for both production and experimentation flows does not exceed 50%.
In a second scenario, the ISP gives permission to launch experimentation flows on alternative paths even when they are longer. In this way, when the shortest paths are occupied by production traffic, experimentation flows can skirt and borrow paths with sufficient available bandwidth. This means that experimentation flows can be launched in a shorter time period than the scenario where only shortest paths must be used. One can observe the results of this configuration in Figure 6. Indeed, the values of the total used capacity can reach values near the total available bandwidth in the network. Then, the service can launch more flows in parallel and save flow waiting times. It is noteworthy that the results of the two previous scenarios, the temporal dependency between experimentation flows is not taken into consideration. The scheduler can profit to the maximum from the network parallelism because it has the liberty to launch as many experimentation flows as the network conditions can support. In case of temporal dependency between experimentation flows, the usage of the network capacity is slowed by the chronological order between flows. Figure 7 shows the average usage of the total network capacity as of function of the rate of the temporal dependency between experimentation flows. This curve shows clearly that the capacity usage decreases with the increase of the temporal dependency. For a 0% dependency, one obtains the maximal network usage obtained in Figure 6.
All these results prove that the ISP respects the requirements

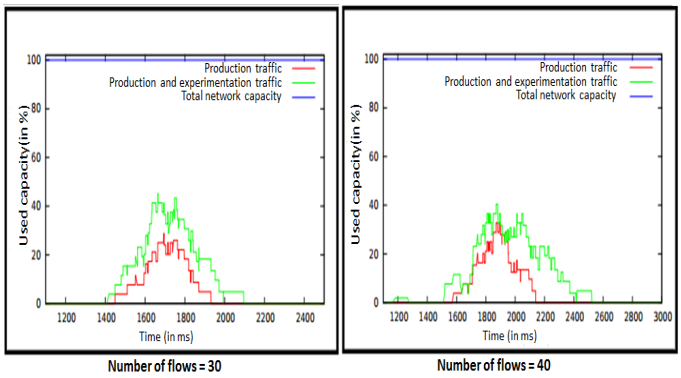

Fig. 5: Usage of network capacity (shortest path only)

of the experimenter on network conditions to be encountered by experimentation flows and that experimentation traffic do not disturb the production traffic as it is not launched on paths where the available bandwidth is not sufficient. The friendliness with the production traffic is discussed in depth in the following paragraph.

2) Friendliness with production traffic: To show that the production traffic has a greater priority than experimentation traffic, we compare, from one side, the waiting times of experimentation flows and production flows and on other side, we conduct a study on the usage of network capacity by each type of flows.

- Study of the waiting time: In this study, we neglect the temporal dependency between experimentation flows. We rather consider that all experimentation flows can be launched in parallel when network conditions are sustainable. In this case, one can make an equitable comparison between waiting times of experimentation and production flows. In Figure

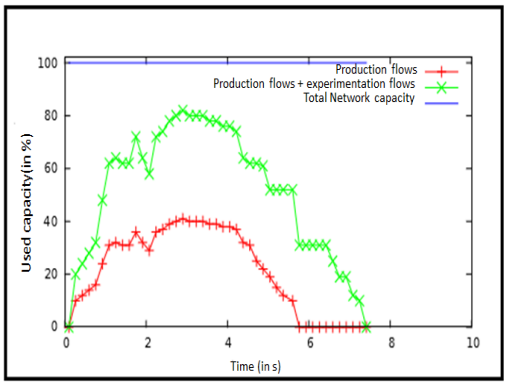

Fig. 6: Network capacity usage in the multiple traject case

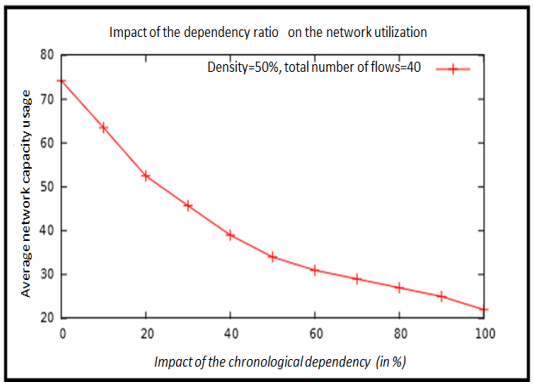

Fig. 7: Impact of dependency on capacity usage

8, we display the simulation results of the waiting time of each type of flows for several value of the density of experimentation flows. The latter value is computed as being the ratio of the number of experimentation flows to the total number of flows. We consider scenarios where the total number of flows takes two different values 30 and 40. The figure shows clearly that the waiting time of production flows still null independently of the density of experimentation flows. This proves that production flows have a higher priority than experimentation flows. Whenever a production flow arrives to the network, it is immediately granted the available bandwidth. However, experimentation flows have to wait for the availability of network resources before being launched. The waiting time of experimentation flows increases in accordance to the growth of the total number of flows and with the density of experimentation flows.

- Study of available bandwidth sharing: We base this study on simulation results shown in Figure 5 and Figure 6. These results testify the friendliness of the sharing of the available bandwidth between production and experimentation traffic. In fact, the shape of the total used bandwidth is similar to the shape of the curve of the bandwidth used by production flows. The latter flows occupy mainly the available bandwidth whereas experimentation flows obtain the remaining proportion. So, the experimentation traffic does not disturb the production one.

3) Quality and duration of experiments: In the remaining part of the simulation results analysis, we focus on the satisfaction of the experimenter. The latter ideally aims to launch an experimentation, in the shortest possible time, while respecting not only the required network conditions for each of the experimentation flows but also, the chronological order imposed by the experimentation logic. When the quality of the path granted to an experimentation flow is considered, the scheduler of our experimentation service is exigent and ensures that network conditions are those required by the experimenter before launching any experimentation flow. A relaxation of the experimentation quality constraint to gain in experimentation duration can be studied as a future work.

We play up the results of the total experimentation duration as a function of several criteria. This duration is defined as being the delay between the time by which the experimentation description has been submitted and the time by which experimentation results are delivered to the experimenter.

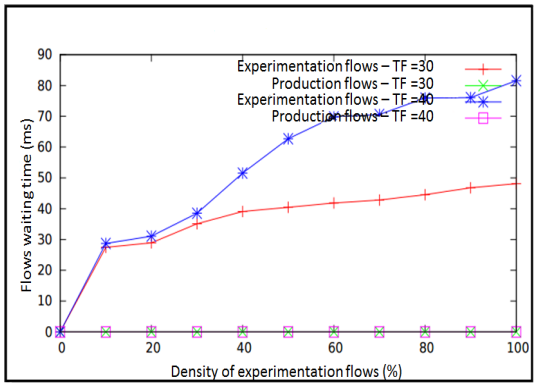

Fig. 8: Measurements of waiting time

- Impact of the size of the experimentation: In the objective of studying the effect of the size of an experimentation on its duration, we consider an experimentation with a slight temporal dependency (10%) and a density of experimentation flows of 50%. We measure after that the obtained experimentation duration. The results are shown in Figure 9. As expected, the experimentation duration grows linearly with the experimentation size. The experimenter waits a time which is proportional to the size of the job which it has submitted to the service.

- Impact of the chronological dependency between flows: The goal is to vary the chronological dependency between experimentation flows and study its impact on the experimentation duration. This study is presented in Figure 10, which shows that experimentation flows must wait for each other and then the experimentation duration will increase in consequence. The delay depends mainly on the ratio of temporal dependency and remains acceptable even for high dependency ratios compared to the independent experimentation flows.

- Impact of production traffic: Figure 11 indicates the impact of the presence of production traffic on the experimentation duration. It shows that for low values of the traffic density, the experimentation duration reaches its maximal values. It is clear that for congested networks, the experimenter must wait more to obtain results.

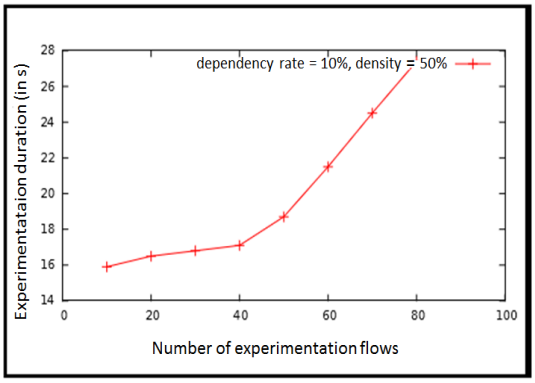

Fig. 9: Impact of the size of the experimentation on its duration

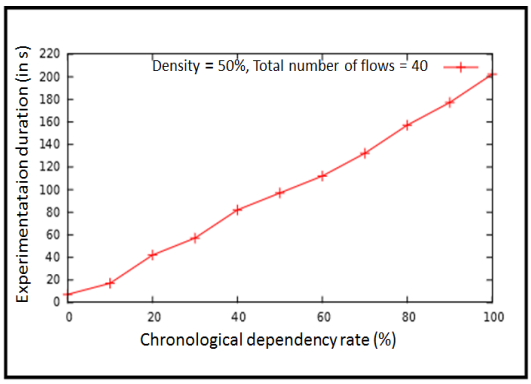

Fig. 10: Impact of the chronological dependency between flows

The simulation results prove the efficiency of the ISP experimentation service in terms of experiments duration and quality. In addition, they show the friendliness of experimentation traffic with production traffic and the outstanding performance of our scheduler in ensuring network conditions required for experiments and managing network capacity usage.

VI. Conclusion and Perspectives

Innovating ideas for networks and services of the future need inventing large scale heterogeneous testbeds. Although efforts made by community to enhance testbeds and federate them. We show in this work that without involving ISPs in the experimentation process, that new challenge testing scenarios will suffer from poor flexibility, realism and resource shortage. We propose instead that ISPs offer experimentation as a service. The proposed service is ensured thanks to an SDN derived business application that controls scheduling of experimentation traffic, resource allocation to experiments, monitoring and collection of results.

Our goal is to ensure isolation between experimentation and production traffic while ensuring realistic network conditions required by the experimenter. The presence of exogenous traffic is mandatory in some testing scenario however an attention to the friendliness of experimentation traffic has to be paid.

Our simulation proves the efficiency of the ISP experimentation service in ensuring benchmarking of experiments and in managing shared test resources.

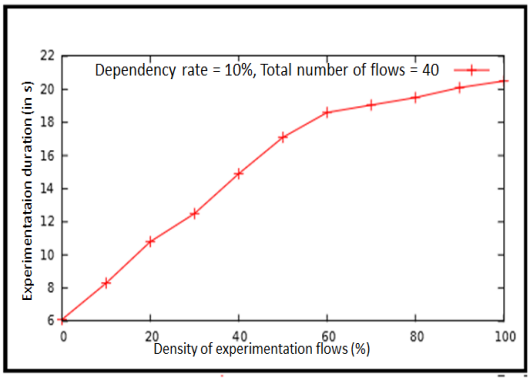

Fig. 11: Impact of production traffic on experimentation duration

As a future work, on one hand, we are working and cooperating with an ISP to implement the experimentation service in a real world production network. On the other hand, we are studying the automation of the experimentation workflow for the support of advanced testing scenarios and focusing on the allocation of in-network middleboxes to ensure deep programmability of experimentation functionalities (traffic generation, results aggregation, etc.).

Acknowledgment

The authors would like to thank Ibtissem Dhiab for her internship at Esprit Tech.

References

[1] Nick Feamster, Jennifer Rexford, and Ellen Zegura. 2014. The road to SDN: an intellectual history of programmable networks. SIGCOMM Comput. Commun. Rev. 44, 2 (April 2014), 87-98.

[2] D. Kreutz "Software-defined networking: A comprehensive survey", Proc. IEEE, vol. 103, no. 1, pp.14 -76 2015

[3] OpenFlow-enabled SDN and Network Functions Virtualization, 2014 [online] Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-sdn-nvf-solution.pdf

[4] Batalle, J.; Ferrer Riera, J.; Escalona, E.; Garcia-espin, J.A., "On the Implementation of NFV over an OpenFlow Infrastructure: Routing Function Virtualization," Future Networks and Services (SDN4FNS), 2013 IEEE SDN for , vol., no., pp.1,6, 11-13 Nov. 2013

[5] NS-3 Network Simulator, https://www.nsnam.org.

[6] PlanetLab testbed portail, https://www.planet-lab.org

[7] ORBIT testbed portail, http://www.orbit-lab.org

[8] M.Berman, C.Elliott, L.Landweber, GENI: Large-scale distributed infrastructure for networking and distributed systems research, ICCE, 2014

[9] Future Internet Research and Experimentation Initiative, http://www.ict-fire.eu

[10] B. Raghavan "Software-defined Internet architecture: Decoupling architecture from infrastructure", Proc. 11th ACM Workshop on Hot Topics in Networks, pp.43-48

[11] J. Kempf et al. OpenFlow MPLS and the Open Source Label Switched Router. In Proc. of ITC, 2011.

[12] N.McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L.Peterson, J.Rexford, S.Shenker and J.Turner, OpenFlow: Enabling Innovation in Campus Networks, ACM SIGCOMM Computer Communication Review 69 Volume 38, Number 2, April 2008

[13] Pohl, I. 1970. Heuristic search viewed as path finding in a graph. Artificial Intelligence 1:193204.