Multi-Objective Scheduling of Micro-Services for Optimal Service Function Chains

Deval Bhamare
Qatar University,
Doha, Qatar
devalb@qu.edu.qa

Mohammed Samaka
Qatar University,
Doha, Qatar
samaka.m@qu.edu.qa

Aiman Erbad
Qatar University,
Doha, Qatar
aerbad@qu.edu.qa

Raj Jain
Washington Univ.,
St. Louis, USA
jain@wustl.edu

Lav Gupta
Washington Univ.,
St. Louis, USA
lavgupta@wustl.edu

H. Anthony Chan
Huawei Technologies,
Plano, USA
h.a.chan@ieee.org

***Abstract:** Lately application service providers (ASPs) and Internet service providers (ISPs) are being confronted with the unprecedented challenge of accommodating increasing service and traffic demands from their geographically distributed users. Many ASPs and ISPs, such as Facebook, AT&T and others have adopted micro-service architecture to tackle this problem. Instead of building a single, monolithic application, the idea is to split the application into a set of smaller, interconnected services, called micro-services (or simply services). Such services are lightweight and perform distinct tasks independent of each other. Hence, they can be deployed quickly and independently as user demands vary. Nevertheless, scheduling of micro-services is a complex task and is currently under-researched. In this work, we address the problem of scheduling micro-services across multiple clouds, including micro-clouds. We consider different user-level SLAs, such as latency and cost, while scheduling such services. Our aim is to reduce overall turnaround time for the complete end-to-end service in service function chains and reduce the total traffic generated. In this work we present a novel fair weighted affinity-based scheduling heuristic to solve this problem. We also compare the results of proposed solution with standard biased greedy scheduling algorithms presented in the literature and observe significant improvements.*

Keywords — greedy scheduling; micro-services; multi-cloud; fair weighted affinity-based scheduling; Service Function Chaining; SFC; virtual machines.

I. Introduction

With the explosion of mobile and sensory devices, service demands and data traffic are growing rapidly. The popularity of Internet of Things (IoT) has significantly contributed to this trend, with millions of new sensing devices exchanging data. According to Wireless World Research Forum (WWRF), the number of wireless devices connected through networks is expected to be 100 billion by 2025 [1]. Cloud computing has been considered as a major enabler for IoT [2]. The sensing devices, as well as end-users, are generally spread across geographically distributed areas. This mandates the ASPs and ISPs to deploy the services on multiple clouds for scalability, redundancy and quicker response [1, 3, 13].

ASPs and ISPs are increasingly using virtualization technologies to deploy their services over standard high-volume infrastructures. Services, which were monolithic software in the past, are being replaced by a set of light-weight services called micro-services [5, 6]. Micro-services are generally spread across multiple clouds for point-of-presence to be closer to the distributed mobile users. These services are then chained through a process called service function chaining (SFC) [7] to create a complete end-to-end service. The goal is to permit the traffic flow smoothly through the network, resulting in an optimal quality of experience for the users.

Infrastructure as a service (IaaS) is a famous and most widely used service offered by cloud service providers (CSPs) to the ASPs and ISPs. With IaaS, cloud environment infrastructure resources such as computation, storage, and network are provisioned for service providers. A good example is EC2 (Elastic Compute 2) service offered by Amazon [22]. With the advancements in the virtualization technology, the services are being deployed over virtual machines (VMs). ASPs and ISPs send requests to CSPs and obtain the resources from their clouds to deploy the micro-services as per the requirements of the users [3]. Micro-services can easily be deployed over virtual machines (VMs) or containers allowing service providers to flexibly load balance and easily deploy their applications [5, 6]. The users benefit from the quick response and lower costs while ASPs and ISPs benefit from quick and comparatively cheap deployment options. Micro-services are usually scaled depending on the dynamic user demands. Because of the nature of the contemporary information technology (IT) and telecommunications applications, the services need to be highly available, almost as much as 99.999% [13]. Additionally, most of the contemporary applications are sensitive to the delays, jitter, and packet-loss (such as online games, healthcare applications, video streaming and others). Many of these services are required to support millions of subscribers and meet the rigorous performance standards [13, 16]. These requirements mandate the optimal placement and scheduling of the services and proper interconnection among them.

Although virtual machine placement problem has already been studied in the literature [8-12], micro-service scheduling problem is relatively novel in the research community. Also, recently, micro-services are moving from host-centric to data-centric model in which the computational resources are moving closer to end users. This results in lower response time to the end-users and lower costs to ASPs and ISPs because of shorter access links. This has led service providers to the concept of micro-clouds at the cellular base stations [3, 13]. Micro-service instances are generally smaller in capacity compared to the monolithic services so that they can be easily deployed and buffered or activated over different micro-clouds as per the user demands. The instances of the micro-services are generally short-lived and dynamic in nature. The scheduling of these services has become an important problem

to reduce the total delays, total required resources, and overall deployment costs [14].

A sample scenario is demonstrated in Fig. 1 below. We consider the example of *Netflix*, an ASP, which is a global provider of streaming movies and television series. As a result of an explosion in mobile devices [1], it would be beneficial for *Netflix* to have a subset of videos (maybe popular videos for a particular mobile user-base) cached at the micro-clouds installed at the cellular base stations with a micro-service handling the user requests and sending replies as per the demand. This will reduce the user-latencies and may result in better user experience. Also, it may result in lower operational expense (OpEx) to *Netflix* by reducing the usage of expensive wide area network (WAN) bandwidth.

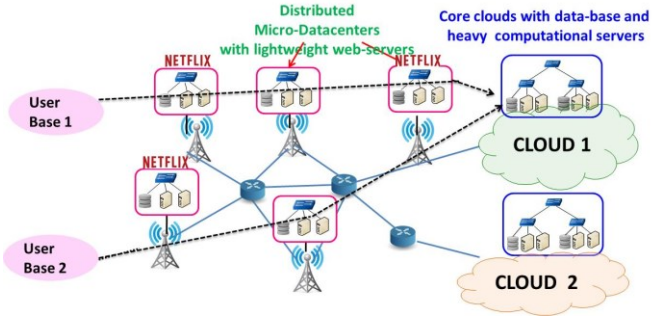

Fig. 1. Micro-Clouds at the base station for quicker response.

Researchers are working to come up with novel and innovative schemes to design efficient algorithms for appropriately placing the services [4, 8], splitting the load across instances on multiple clouds, and chaining them to improve performance parameters. However, we observe that there is a lack of research work in the domain of micro-service scheduling across multiple clouds for optimal service function chain (SFC) architecture, for both ASPs and ISPs [6]. Especially important factors such as service level agreements (SLAs), network parameters, and network latencies, have been ignored.

The micro-service scheduling problem generally comprises two sub-problems: (1) selecting types and numbers of the service instances to be scheduled and (2) selecting physical machines (PMs) or virtual machines (VMs) on which the services should be scheduled. Please note that now onwards we will use the terms service and micro-service (MS) interchangeably. Common heuristics used in the state-of-art systems for these tasks are "*greedy with bias*" [8, 9, 19]. The bias is towards some factor such as: (1) select a service with the first finish or (2) select service with the longest finish. Similarly, the bias while selecting VMs/PMs are: (1) select most-loaded VM/PM or (2) select least-loaded VM/PM [11, 16]. We implement all the four combinations along with our proposed Fair Weighted affinity-based Scheduling (FWS) approach (explained in Section IV). Our proposed novel heuristic performs scheduling of micro-services on multiple VMs/PMs spread across multiple clouds. We consider different user-level service level agreements (SLAs), such as traffic-affinity among services [8], user delays, and cost constraints. Also, we consider network parameters such as link loads and network traffic. Our aim is to reduce the overall turnaround time for the service and reduce the total inter-VM traffic generated.

The rest of the paper is organized as follows. In the next section, we discuss the state-of-the-art of the scheduling problem in the SFC context to show the limitations of existing approaches. Section III formalizes the micro-service scheduling problem. In Section IV, we propose a novel FWS algorithm for micro-service scheduling and explain the experimental setup, and in Section V, we present the comparison results. Finally, Section VI concludes the paper.

II. Related Work

The problem of scheduling virtual machines has been actively pursued in the industry and academia for years. Researchers argue that the problems need to be revisited from the SFC perspective as SFC has some unique features [18]. For example, SFC is an ordered chain of services, so the order in which the service instance needs to be visited is defined dynamically by the traffic flows [7]. Researchers have identified the importance of micro-services as an enabler for SFC and have started identifying and addressing various problems in this context [5, 6]. Lopez-Pires and Baran [18] provide a comprehensive survey of the VM placement strategies and solutions. SFC placement is a widely studied topic, and works such as in [8-12, 20] provide a wide range of VM placement strategies in a single cloud or across multiple clouds forming efficient SFCs.

Due to the time sensitive nature of contemporary applications, VM placement alone is not sufficient to yield acceptable performance in deployments of SFCs over micro-clouds. Especially from the perspective of the short-lived micro-services, scheduling is more important than the placement problem. Also, mobile users have strict SLAs as far as tariffs and delays are concerned. This mandates ASPs to create points of presence close to the mobile users, reducing access latency and overall cost. Merely placing the micro-services efficiently is not sufficient to obtain optimal results. Recently, researchers have become aware of the importance of scheduling problem for micro-services in SFCs, especially for the micro-clouds at the edges to guarantee carrier-grade performance [18].

Yoshida et al. propose a multi-objective resource scheduling algorithm (MORSA) for network function virtualization (NFV) infrastructures [15]. The tool provides different options to optimize many parameters such as delays, cost, resources, and others. However, the authors do not consider multi-cloud scenarios and neglect network parameters such varying link delays with different traffic loads. Similarly, Mujambi et al. propose a set of algorithms for network virtual function (NVF) scheduling and find Tabu search to perform the best [16]. However, the algorithms proposed do not consider the links between physical/virtual nodes, and consequently, the link delays for transferring a given function

from one node to another are considered to be negligible. This is not true, especially in the SFC scenario, where interconnection among the micro-services is critical to forming a complete end-to-end service.

Ferrer et al. have tried to formulate the NFV scheduling problem; however, important network latency constraints are ignored [17]. In addition to link delays, these constraints include many SLA constraints, such as affinity/anti-affinity between service instances [14]. Lucrezia et al. introduce network-aware scheduling capabilities in OpenStack, the open-source reference framework for creating public and private clouds [19]. The proposed solution is specific to OpenStack and needs to be upgraded to accommodate the various multi-cloud scenarios over multiple platforms. Lakkakorpi, Sayenko and Moilanen [21] provide a comparison of a set of scheduling algorithms for WiMAX base stations. The focus has been on variants of round robin and proportional fair scheduling strategies. In this work, we propose a novel fair weighted affinity-based scheduling scheme for scheduling micro-services and compare results with greedy strategies. We show significant improvements with the proposed heuristic.

III. Problem Definition

In this section, we discuss the micro-service scheduling problem in the context of SFC in more details. We start our discussion with a particular use case. We consider an ASP such as *Facebook* (*FB*) and take up a hypothetical example to explain the details. It is important to note that the scope of the problem under consideration is not only limited to the application services but is equally important for the telecommunication services, multimedia services, and network services as well [7, 8].

As shown in Fig. 2, different groups of users from various user-bases may send different types of web requests to *FB* webserver(s). For example, some users may be interested in sign-up functionality and others may just login to access other services. The sign-up requests, after passing through the firewall, are passed to a set of services, which handle user registration logic (in this case *firewall* $\rightarrow f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow f_4 \rightarrow f_5$ $\rightarrow$ *database*). However, login requests may have to be passed through deep packet inspection (DPI) in addition to the firewall to distinguish among user demands (such as wall-post, photo upload or online *FB* integrated games). Thus a complete service chain may comprise IT and telecommunication services, combined. This example is just for an illustration purpose of the service flows, and it may be different in actual *FB* implementation of the services. The important point to be noted here is the dynamic formation of complex and hybrid service chains, which comprise a different set of micro-services implemented at the application layer. Some of the services in the process such as the firewall, the deep packet inspector (DPI) and the database (DB) may stay for longer durations compared to other short-lived services.

In this example, if we consider some specific functionality, such as user registration (sign-up), wall-post on *FB* or other integrated game applications, a specific set of service instances need to be executed. Such sets of service instances may be switched on/off as user demands vary, especially at the micro-clouds, since the capacities are limited. Scheduling these service instances over the available resources is an important problem. In this work, we consider four SFCs comprising twenty micro-services in total. The SFC shapes and graphs are shown in the Fig. 3. We note that the topologies of the SFCs also indicate their execution order. For example, in SFC 1, service f_2 has to be executed after f_1. This may be because of the business logic dependence or some mandatory network traffic flow demand. For example, web-service logic handling service has to be executed before the service handling databases; or firewall must be executed before the business logic, etc.

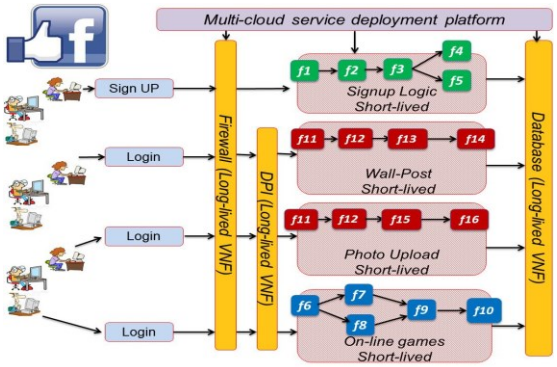

Fig. 2. SFCs for different services offered by an ASP (such as *FB*).

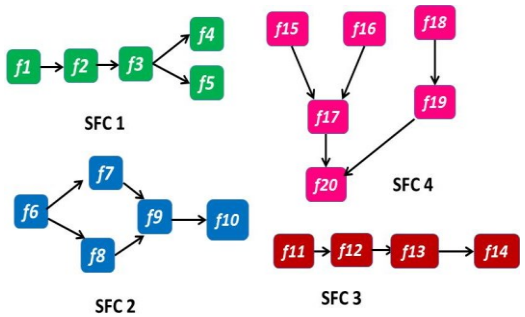

Fig. 3. Four SFCs with 20 virtual functions (VFs) used for evaluation.

However, f_4 and f_5 may be executed in parallel after f_3, since they are independent of each other. Similarly, in SFC 2, f_7 and f_8 may be executed at the same time after f_6. However, f_9 has to be executed only after both f_7 and f_8 have finished their execution. This mandatory ordered flow of services in SFCs make scheduling an important but a complex problem.

Let us now consider three SFCs from the example above displayed on the left of Fig. 4. On the right-hand side, we show the Gantt chart for the scheduling of the micro-services over available resources, using the virtual machines (VM_1 to VM_5), deployed across three clouds C_1, C_2, and C_3. Vertical lines indicate the time slots and each service needs different time to finish the execution. We assume that three user requests for these three SFCs arrive at the same time. The widths of the services indicate the total time needed to execute the services (longer services mean longer time for execution). Possible

scheduling to optimize the total time and the resources required for the three SFCs on the available resources is shown in Fig. 4. We observe that all the executions finish before time-slot t_{10} keeping VM_5 free and ready to serve another incoming request. We argue that a sophisticated heuristic is needed to solve the large scale micro-service scheduling problem within acceptable time limits. In the next section, we propose our novel scheduling FWS scheme and explain the experimental setup.

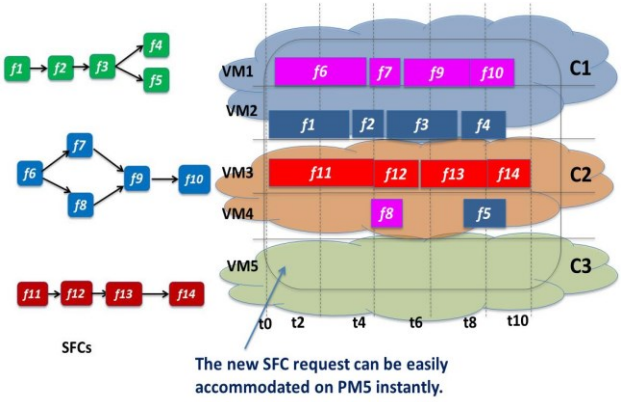

Fig. 4. Gantt chart for optimal scheduling.

IV. Heuristics and Experimental Setup

In this section, we propose a novel fair weighted affinity-based scheme (FWS) for the scheduling problem under consideration. The heuristic can be divided into two distinct parts, that is, (1) selection of next service instance to be scheduled and (2) selection of next VM (or PM) on which the service instance needs to be scheduled. Heuristic starts at the time $t = t_0$. User requests arrive dynamically with inter-arrival time exponentially distributed, that is, the arrival rate is characterized by *Poisson* distribution [16, 23]. Let U be the set of users, waiting for the service or being served at any time t. Initially, we prepare the graphs for each SFC for each user u in U. It is to be noted that the graph may have disjoint sets of sub-graphs.

A sample inline service graph is shown in Fig. 5. Three possible service chains are highlighted by solid, dotted and dashed lines (there may be several other SFCs as well). Also, the users may demand a single functionality, such as F_9 shown in the figure. Again, these graphs can be of any shape and size, depending on the service provided by a specific ASP and the types of end-user demands. We have used various resource combinations (from Amazon EC2 [22]) mentioned in Table I to simplify configurations so that resource requirements can easily be mapped to the nearest available configuration. Depending on the user resource demands, a particular VM is chosen from Table II [22] such that the requirements are the closest match. Initially, we assign labels to the services using Coffman-Graham algorithm [9]. It ensures that the service instance that needs to be executed first for the particular SFC (starting service) gets a priority as per the arrival time. The service instance with the highest value of the label is scheduled first. Further, we propose following additional steps (Table II) for fair scheduling as well as to minimize the total turnaround time and total inter-VM traffic.

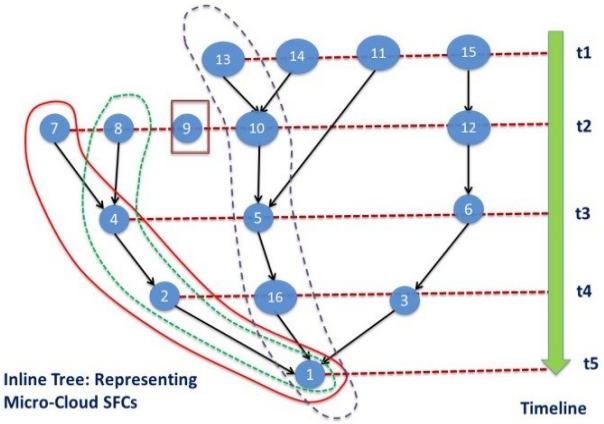

Fig. 5. An inline graph for services forming different SFCs.

Table I. Resource configuration taken from Amazon EC2.

Name	API Name	Memory	Cores	Max Bandwidth	On Demand cost
T2 Small	t2.small	2.0 GB	1 cores	25 MB/s	$0.034 hourly
T2 Medium	t2.medium	4.0 GB	2 cores	25 MB/s	$0.068 hourly
T2 Large	t2.large	8.0 GB	2 cores	25 MB/s	$0.136 hourly
M4 Large	m4.large	8.0 GB	2 cores	56.25 MB/s	$0.140 hourly

Steps 1 and 2 make sure that the longer SFCs and the SFCs which have waited longer in the queue get a fair chance for their scheduling. Step 3 makes sure that the services for the same SFC get scheduled on the same machine, if possible, to minimize the total traffic generated. Otherwise, it tries to schedule the service on the machine with which inter-VM traffic will be minimized, and all capacity constraints are satisfied. We may combine two or more services and deploy them on a single VM as well, provided a VM of that capacity is available. Availability of the VMs depends on the cloud capacity. If a service instance is not serving any user demands, it is buffered in the cloud. In the buffered stage, the service uses fewer resources (such as storage only to save the state). However, it can be brought up quickly whenever relevant user demand arrives, saving resources and time [16]. For simplicity, we assume clouds have infinite buffering capacity.

Table II. FWS algorithm for service scheduling.

Step 1: *We assign weight w to the services, such that:*
$w \propto$ *(number of dependent services in that chain) and*
$w \propto$ *(time spent by the services in the waiting queue).*

Step 2: *If there are ties between two services for scheduling (that is, services having the same labels), the service with higher weight is selected.*

Step 3: *While selecting the VMs/PMs for service deployment, the affinity between services is taken into consideration. Two services belonging to the same instance of an SFC are considered to have higher affinity, and we try to place them on the same VM/PM. This ensures minimum delays and less inter-VM traffic overhead.*

We consider a 20-node topology out of which, 16 are the micro-clouds deployed at the edges such as cellular base stations, closer to end users and four are the core public clouds, with larger capacities, as shown in Fig. 6. Computation and/or data extensive services which need more processing and/or storage capacities and which tend to run for longer times, such as firewall, database services, are generally deployed at core clouds. We assume that each service instance produces data in the range of 5 kB to 20 kB. Also, the number of user requests each micro-service instance can handle at average load is selected from a range of 20 to 100 requests/sec. Time needed for execution of each service is chosen from the range of 10 to 100 milliseconds (ms) [18]. All the values are selected randomly from the given ranges. Also, we assign each user request with some delays and cost constraints it may tolerate. We also make sure these constraints are satisfied while scheduling the micro-services on the clouds. In the next section, we present our results.

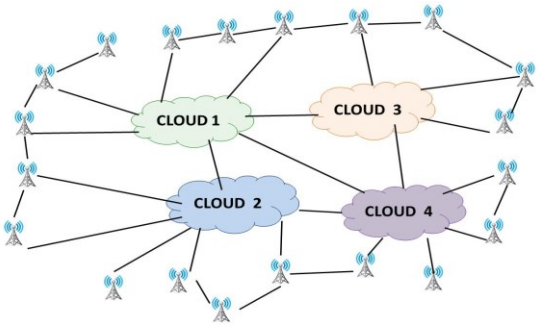

Fig. 6. 20-node topology with 16 micro-clouds and 4 core clouds.

V. Results

We now present the results obtained through the experimental setup. In addition to our FWS approach, we have implemented four additional algorithms based on the greedy biased approach for comparison. Service labeling step is common for all the heuristics. Table III displays the basic steps for the following strategies:

1. Least-full first with First Finish (LFFF)
2. Most-full first with First time (MFFF)
3. Least-full first with Decreasing time (LFDT)
4. Most-full first with Decreasing Finish (MFDT)

Table III. The selection criterion for greedy biased heuristics.

	Machine Selection	
Micro-Service	LF	MF
FF	Select Least full machine first, Select the service of SFC having first finish time	Select Most full machine first, Select the service of SFC having first finish time
DT	Select Least full machine first, Select the service of SFC with longest finish time	Select Most full machine first, Select the service of SFC with longest finish time

Graphs in Fig. 7 show the comparison of the four approaches mentioned above and our FWS approach in terms of the total inter-VM traffic generated. FWS approach (thick yellow line) performs best with the least inter-VM traffic. For example, with 3000 user demands, greedy algorithms produce more than 20 MB of data, whereas FWS only produces less than 10 Mbps data, which is an improvement of 50%.

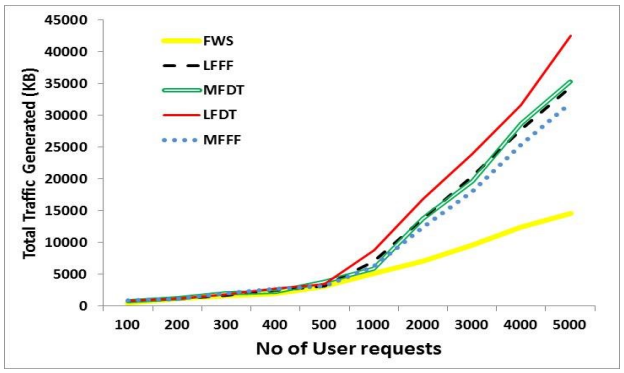

Fig. 7. Total traffic generated (in KB).

Similarly, Fig. 8 shows the average turnaround time for each user where FWS performs best. For example, with 4000 user demands FWS results in turnaround time of less than 220 ms whereas the other algorithms need around 330 ms.

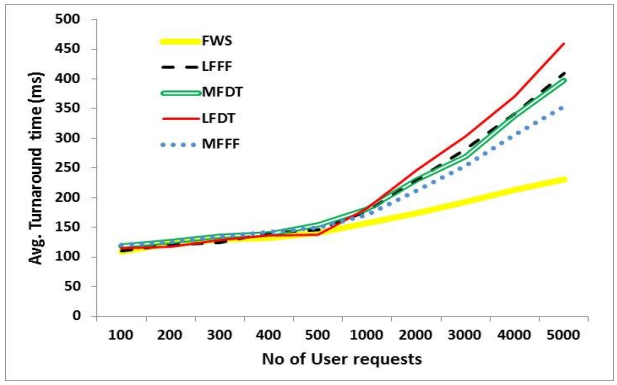

Fig. 8. Total turnaround time (in milliseconds).

However, the average turnaround time alone is not sufficient to measure the performance, especially in the context of the time sensitive applications. Most of the time, if the user demands are not satisfied within the given time constraints, it is as bad as service denied. Hence, we also find out the percentage of user demands which got satisfied in the given time constraints (Fig. 9). We observe that a significantly higher percent of the user demands got satisfied with the FWS approach. The total percentage varies from 100% to 96% as user demands vary from 100 to 5000. On the contrary the percentage drops to 70% for LFFF, 62% for LFDT & MFFF and 74% for MFDT.

We have also analyzed the effect of traffic loads on average turnaround time or average time to schedule all the services. We observe exponential growth in the total turnaround delays as traffic loads in the network grow. We generated dummy traffic to obtain different average traffic loads. The links were modeled as M/D/1 queues, and by the standard formula, we calculate the delays in the links as given in Equation (1) below [23]. We note that T_{ij} is the total delay

on the link (i, j). λ_{ij} is the arrival rate of packets and μ_{ij} is the processing rate of the same link. $T_{ij} = \frac{1}{2\mu_{ij}} \times \frac{2-(\lambda_{ij}/\mu_{ij})}{1-(\lambda_{ij}/\mu_{ij})}$ (1)

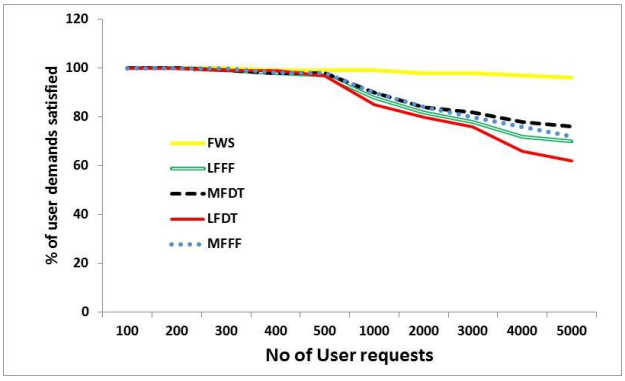

Fig. 9. Percentage of user demands satisfied.

In Fig. 10 we observe that even at 90% traffic load the total delays with the proposed FWS scheme still remain within the range of 250 ms which is proposed to be acceptable within the limits for the contemporary real-time applications [24], while for other schemes, it varies from 400 to more than 600 ms.

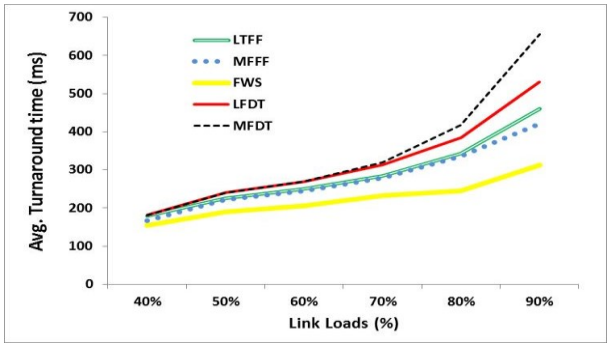

Fig. 10. Average turnaround time.

VI. Concluding Remarks and Future Work

In this paper, we have proposed a novel FWS approach for micro-service scheduling in the multi-cloud scenario to form optimal SFCs. We take into account different delay and cost related SLAs. Also, we consider link loads and network delays while minimizing the total turnaround time and total traffic generated. The proposed approach demonstrates significant improvement compared to standard biased greedy approaches. However, there is still a wide area open for the research in developing novel scheduling algorithms, such as proactive scheduling, to produce better results.

Acknowledgement

This publication was made possible by the NPRP award [NPRP 6-901-2-370] from the Qatar National Research Fund (a member of The Qatar Foundation) and Huawei Technologies. The statements made herein are solely the responsibility of the author[s].

References

[1] L. Sorensen and K. E. Skouby, "User Scenarios 2020 - a worldwide wireless future report," WWRF, July 2009.

[2] M. Daniele, et al., "Internet of things: Vision, applications and research challenges," Ad Hoc Networks 10.7, 2012, pp. 1497-1516.

[3] Q. Zhang, L. Cheng, R. Boutaba, "Cloud computing: state-of-the-art and research challenges." Journal of Internet services and applications 1.1, 2010, pp. 7-18.

[4] European Telecommunications Standards Institute (ETSI), "Network Functions Virtualisation," NFV white paper, October 2014, https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf

[5] K. Indrasiri, "Microservices in Practice: From Architecture to Deployment," [Online] Available: https://dzone.com/articles/microservices-in-practice-1

[6] D. Namiot, S. Manfred, "On micro-services architecture," International Journal of Open Information Technologies, 2014.

[7] M. Boucadair, Ed. "Service Function Chaining (SFC) Control Plane Components & Requirements," Internet-Draft, draft-ietf-sfc-control-plane-07, August 2016, 29 pp.

[8] D. Bhamare, R. Jain, M. Samaka, G. Vaszkun, A. Erbad, "Multi-Cloud Distribution of Virtual Functions and Dynamic Service Deployment: OpenADN Perspective," 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, March 9-13, 2015, pp. 299-304.

[9] Y. Kwok, I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," ACM Computing Surveys (CSUR), 1999, pp. 406-471.

[10] S. Mehraghdam, M. Keller, H. Karl, "Specifying and Placing Chains of Virtual Network Functions," IEEE 3rd International Conference on Cloud Networking, CloudNet 2014, pp. 7-13.

[11] R. Ruiz, T. Stützle. "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem." European Journal of Operational Research, 2007, pp. 2033-2049.

[12] M. Luizelli, L. Bays, L. Buriol M. Barcellos, L. Gaspary, "Piecing Together the NFV Provisioning Puzzle: Efficient Placement and Chaining of Virtual Network Functions," International Federation for Information Processing (IFIP), 2015, pp. 98-106.

[13] D. Bhamare, R. Jain, M. Samaka, A. Erbad, "A Survey on Service Function Chaining." Journal of Network and Computer Applications, 2016, pp. 138-155.

[14] F. Riera et al., "Virtual network function scheduling: Concept and challenges," International Conference on Smart Communications in Network Technologies (SaCoNeT) 2014, pp. 1-5.

[15] M. Yoshida, et al., "MORSA: A multi-objective resource scheduling algorithm for NFV infrastructure." 16th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2014, pp. 1-6.

[16] R. Mijumbi, et al., "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1-9.

[17] F. Riera et al., "On the complex scheduling formulation of virtual network functions over optical networks," 16th International Conference on Transparent Optical Networks (ICTON), 2014, pp. 1-5.

[18] F. Lopez-Pires and B. Baran, "Virtual machine placement literature review," Polytechnic School, National University of Asuncion, Tech. Rep., 2015.

[19] L. Francesco, et al., "Introducing network-aware scheduling capabilities in OpenStack," 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1-5.

[20] M. Xia, et al., "Network function placement for NFV chaining in packet/optical datacenters," Journal of Lightwave Technology 33.8, 2015, pp. 1565-1570.

[21] J. Lakkakorpi, A. Sayenko, J. Moilanen, "Comparison of Different Scheduling Algorithms for WiMAX Base Station," IEEE Wireless Communications and Networking Conference (WCNC), 2008, pp. 1991-1996.

[22] EC2Instances.info, "Easy Amazon EC2 Instance Comparison." [Online]. Available: http://www.ec2instances.info/

[23] R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling," Wiley- Interscience, New York, NY, April 1991.

[24] ITU-T Recommendation Y.1541, "Network performance objectives for IP-based services," 2011.