A SCHEDULING FRAMEWORK FOR PERFORMING RESOURCE SLICING WITH GUARANTEES IN 6G RIS-ENABLED SMART RADIO ENVIRONMENTS

Christos Liaskos[1,2] and Kostas Katsalis[3]
[1]Computer Science Engineering Dept., University of Ioannina, Ioannina, Greece, [2]Foundation for Research and Technology - Hellas, Heraklion, Greece, [3]DOCOMO Euro-Labs,

NOTE: Corresponding author: Christos Liaskos, cliaskos@uoi.gr

Abstract *– Smart radio environments (SRE) transform the wireless propagation phenomenon in a programmable process. Leveraging multiple reconfigurable intelligent surfaces (RIS), the wireless waves emitted by a device can be almost freely routed and manipulated, reaching their end-destination via improbable paths, with minimized fading and path losses. This work initiates with the observation that each such wireless communication customization occupies a certain number of RIS units, e.g., to form a wireless path with consecutive customized reflections. Therefore, SREs can be modeled as a resource of constrained capacity, which needs to be sliced among interested clients. This work provides a foundational model of SRE-as-a-resource, defining service level agreements (SLAs) and objectives (SLOs) for the SRE client requests. Employing this model, we study a class of negative drift Dynamic Weighted Round Robin policies, that is able to guarantee specific SRE resource shares to competing user requests. We provide a general mathematical framework where the class of policies mapping user requests to resources requires no statistical knowledge regarding the arrival distribution or the duration of each user communication. We study the meaning of work conserving and non work conserving modes of SRE operation, and study the convergence properties of our scheduling framework for both cases. Finally, we perform the feasibility space analysis for our framework and we validate our analysis through extensive simulations.*

Keywords – Smart radio environments, intelligent metasurfaces, resource slicing, theoretical foundations, 6G, service differentiation, scheduling, SLA, SLO.

1. INTRODUCTION

Up until recently, resource slicing and scheduling in wireless communications was a matter of the end-points, i.e., configuring the modulation attributions and the MIMO assignment between the transmitter and the receiver system. With the emergence of SREs, the role of the propagation environment, i.e., the scattering objects in their vicinity, can be programmed in real-time [1–3].

An SRE–initially appearing in the literature under the term Programmable Wireless Environment (PWE) [4]– is a complete, end-to-end system that transforms the propagation of electromagnetic (EM) waves into a software-defined process. SREs are created by coating all major planar objects, such as walls and ceilings in an indoors space, with addressable RIS units [5]. An RIS is as a rectangular, planar structure resembling a tile. An RIS unit can receive commands via well-defined application programming interfaces invoked from a central server [5]. Employing these commands, the server orchestrates the SRE RIS tiles, modifying the way they interact with impinging EM waves in order to best serve a set of client devices. Carefully orchestrated RIS units result into SREs that can effectively mitigate path loss, multi-path phenomena, Doppler effects and interference in challenging Non-Line-of-Sight (NLOS) cases [6].

Based on the SRE workflow, it is apparent that RIS are resources that need to be efficiently shared between different users and effectively exploited. However, and despite the early and enthusiastic adoption of the RIS technology in 6G communications, the topic of SRE resource scheduling has been widely overlooked in the literature. Nonetheless, the importance of dynamic control schemes that are able to provide scheduling guarantees is a foundational concern for any modern system (e.g., in operating systems and thread schedulers [7], hypervisor operations [8], queueing systems [9]), since resources are always bounded and because of the increased complexities in today's communications infrastructure.

In this work we introduce a resource sharing model for SREs, enabling the differentiation of services by means of providing RIS units for different classes of customers, according to specified Service Level Objectives (SLOs), defined in enterprise Service-level agreements (SLAs) of the SRE operation. We define the nature of SLAs and SLOs for SREs, based on the established types of wireless propagation manipulation in the literature, i.e., wireless routing for signal-to-noise ratio (SNR) maximization, wireless power transfer, energy harvesting and physical-layer security [6]. Then, assuming a completely stochastic environment with unknown mean values regarding the arrival and service processes for every customer class and without relying on prediction techniques, we present a class of negative drift Dynamic

Weighted Round Robin (DWRR) policies used to provably apply guaranteed service differentiation, and enforce a defined ratio of SLA types served by the SRE over time. The rationale is that the SRE should not serve just one type of SLA (e.g., just wireless power transfer) under saturated user request arrivals.
Moreover, in schedulable systems, the distinction between work-conserving and non work-conserving modes of operation is of paramount importance. The reason is that the actual performance is largely affected by the way service-capacity redistribution is made at run time. For example, in hypervisor technologies [10], the selection of the mode can be made by tuning specific configuration parameters (e.g., setting the "cap" in the XEN hypervisor, when using Credit scheduling [11]). Thus, we study the concept and meaning of work conservation in the SRE workflow, and propose resource scheduling approaches in the conserving and non-conserving cases.
The contributions of the present work are summarized as follows:

- We propose a foundational, resource slicing layer in the SRE workflow stack. This allows for scheduling the assignment of the SRE resources, i.e., its RIS tiles, per user request and user type.
- We formalize the concepts of user requests, SLAs, SLOs and work-conservation for the case of SREs, deriving their definition from the physical capabilities of the RIS.
- In the general case of arbitrary user request arrival and service time distributions, under non-work conserving mode of operation, we prove that a class of DWRR request-to-resource mapping policy exists such that the system not only regulates but also convergences to the SLA goals. Under work conserving mode of operation, we prove that the same class of DWRR policies can satisfy a minimum guaranteed service.
- We define the feasibility space of these DWRR policies for both of work conserving and non-work conserving modes of operation.

The paper is organized as follows. We present the related work in section 2. In section 3, we describe the proposed resource allocation model, the detailed provisioning objectives and the corresponding mathematical formulation. In section 4, we define a class of DWRR scheduling policies and in section 5, we provide the theoretical framework regarding convergence and optimality of the defined class. In section 6, we evaluate the performance of the proposed policies. We conclude our study in section 7.

2. RELATED WORK

We divide our survey of the related studies in terms of SREs and scheduling.

SREs SREs can be perceived as a sub-part of PWEs, an approach for exerting deterministic control over mechanical, acoustic and thermal energy propagation, and any application setting [12, 13]. Within the EM domain the PWE focus is to craft software-defined EM vector field distributions at any location in a 3D space, employing metasurfaces. Metasurfaces are essentially passive converters of surface current distributions. Impinging waves create a surface distribution "A" upon a metasurface, and embedded control elements convert it to a state "B" that yields the required EM field as a global response [14]. This general model is denoted as software-defined metasurfaces (SDMs) [5].
PWEs are defined as complete end-to-end systems offering software-defined wireless propagation, and following a layer organization approach [5, 15]. At the physical layer, PWEs consider any type of metasurface equipped with a hardware gateway offering standard connectivity. At the network layer, PWEs offer the infrastructure, protocols and workflows covering the system operation from user registration, authentication and statement of requirements to the corresponding wireless channel optimization. The control layer is aligned with the SDN paradigm offering direct integration to existing network infrastructure. The central controller hosts software interfaces for offering an abstracted graph model of the wireless environment. The application layer allows for customized wireless propagation-as-an-app for increased communication quality, security and wireless power transfer.
SREs focus the PWE concept explicitly on the signal processing aspects of wireless communications, and especially on the use of metasurfaces for engineering a channel matrix [2, 3, 16–18]. This direction introduced the more marketable terms SRE and RIS instead of PWE and SDMs, and popularized the concept to the general public. Based on these premises, the goal is to iteratively optimize the phase shifter states (free variables) of a reflectarray-RIS [19], in order to maximize a scalar quantity representing an aspect of the channel transfer function (fitness function). Additionally, given that SREs focus on the signal processing aspect of PWEs (e.g., deriving stochastic channel models), the required protocols, system workflows and integration-to-infrastructure processes are left undefined, implying an underlying PWE infrastructure. In a layered sense, PWE is a complete, top-to-bottom systemic approach, while the smart radio environment is an extensive set of studies focusing on the application of PWE in wireless channel engineering while employing reflectarrays. For this reason, SREs and PWEs bear major similarities and share common elements [20], to the point that they are considered synonyms in the literature.
In terms of resource slicing in SREs, the challenge is acknowledged in the literature [6, 21], but no related solutions exist. While there exist algorithms that can fully configure an SRE *given* a resource allocation quota per user request [6, 22, 23], there are no algorithms to

derive this quota in the resource allocation sense. The present work fills-in this gap.

Scheduling There is an abundance of studies in the area of guaranteed resource allocation in general, across multiple application domains [8], [24], [11], [25], [26]. Proposals to change priorities dynamically, using feedback-based stochastic control are presented in multiple works [27], [28]; the authors of these works use a similar approach to ours but focus on rate control through scheduling. Service differentiation based on feedback control is also investigated in [29], while in [30] feedback control together with rate predictions is used to provide service differentiation. On-line measurements with prediction and resource allocation techniques is examined in [31] and a prediction based on a linear approximation technique was proposed in [32] where convergence to the goal vector was presented only through simulations.
Particularly related to the present work is the negative drift concept and has been applied in multiple fields like mechanical engineering [33]. Related analysis regarding stochastic analysis tools (namely Lyapunov analysis in systems where negative drift is applied to change priorities) were used in several works [34], [35], [36]. In [37] and [38] aggregate packet rate guarantees to individual clients is combined with negative drift and measurement-based admission control techniques.
The decomposition of DWRR policies was originally introduced in [39] and [40], where a theoretical analysis was presented regarding saturated arrival conditions. In this work we extend the mathematical framework to include the stochastic arrivals case, we present proof of convergence for the non work-conserving mode of operation, while we investigate time dependence through analysis in the limit and SLO satisfaction for all modes of operation. In addition, we make a feasibility space analysis and present the dependence on the service redistribution algorithm in the final performance.

3. THE PROPOSED RESOURCE SLICING MODEL FOR SRES

3.1 Prerequisites: The SRE workflow

SREs follow the operation workflow summarized in Fig 1 [15]. We proceed to describe it in a bottom-to-top approach.
An SRE is created by placing RIS units over large surfaces, such as walls. Each RIS has a gateway providing basic connectivity capabilities. The gateway itself is intended to be extremely lightweight, supporting basic commands for getting and setting their EM behavior. RIS units are hierarchically organized in smart walls, with each wall being controlled by a dedicated server, as shown in Fig. 1. The smart wall server provides advanced capabilities, such as standards-compliant communication with existing infrastructure, as well as RIS addressing, monitoring and secure interaction. The smart wall servers are connected to a central SRE controller, whose operation is summarized as follows:

1. Receive the user requests for service from the SRE (whose acquirement is described in the next paragraph).

2. Perform resource allocation, i.e., assign them to a wall and a number of RIS units within it (i.e., a coarse RIS selection, which is the scope of the present work). Notice that, while PWEs allow for multi-wall wireless paths [15], their simpler counterpart, the SREs, focus on single-wall communication. In this study, we adopt this simplification.

3. Based on the resource allocation outcome, invoke an SRE configuration algorithm (e.g., KpPaths [6] or NNPaths [22]) to perform the specific RIS selection and configuration.

4. Finally, when a user has been serviced (e.g., when exiting the SRE), update the RIS usage statistics over time.

The users requests are gathered as shown in Fig. 1. A beacon advertises the existence of an SRE (much like SSIDs are broadcast for WiFi). The user passes authentication and states a communication objective type (e.g., wireless power transfer, advanced physical layer security, or SNR maximization). These are relayed to the SRE controller for further processing as described above. Finally, the user enters the SRE, while receiving beamforming directions in order to aim at the smart wall that will serve his device.

3.2 The Proposed SRE Resource Slicing Model

We present the proposed SRE resource slicing model, depicted in Fig. 2. The model comprises a set of service domains, $\mathcal{D}$, indexed by $i = \{1, \dots, D\}$, which represent the SRE communication service types, such as SINR maximization, physical-layer security, wireless transfer, Doppler effect mitigation, etc. Additionally, we consider a set of smart wall servers, $\mathcal{M}$, indexed by $m = \{1, \dots, M\}$.
Each service domain is associated with a given percentage p_i of aggregate RIS resources to be committed for serving domain requests over an infinite time horizon. The SRE controller uses one First-In-First-Out (FIFO) queue per domain to hold user requests, and distributes them from the domains to the set of smart walls, seeking to uphold the domain SLAs in the process. The controller has no a priori knowledge of the execution time of the requests, i.e., how much time a user will stay in the SRE until exiting. This parameter is provided a posteriori as feedback, in order to schedule future service requests more effectively. It is clarified that a service request is modeled as a non-preemptive process that will occupy a server for a given, non-infinite amount of time.

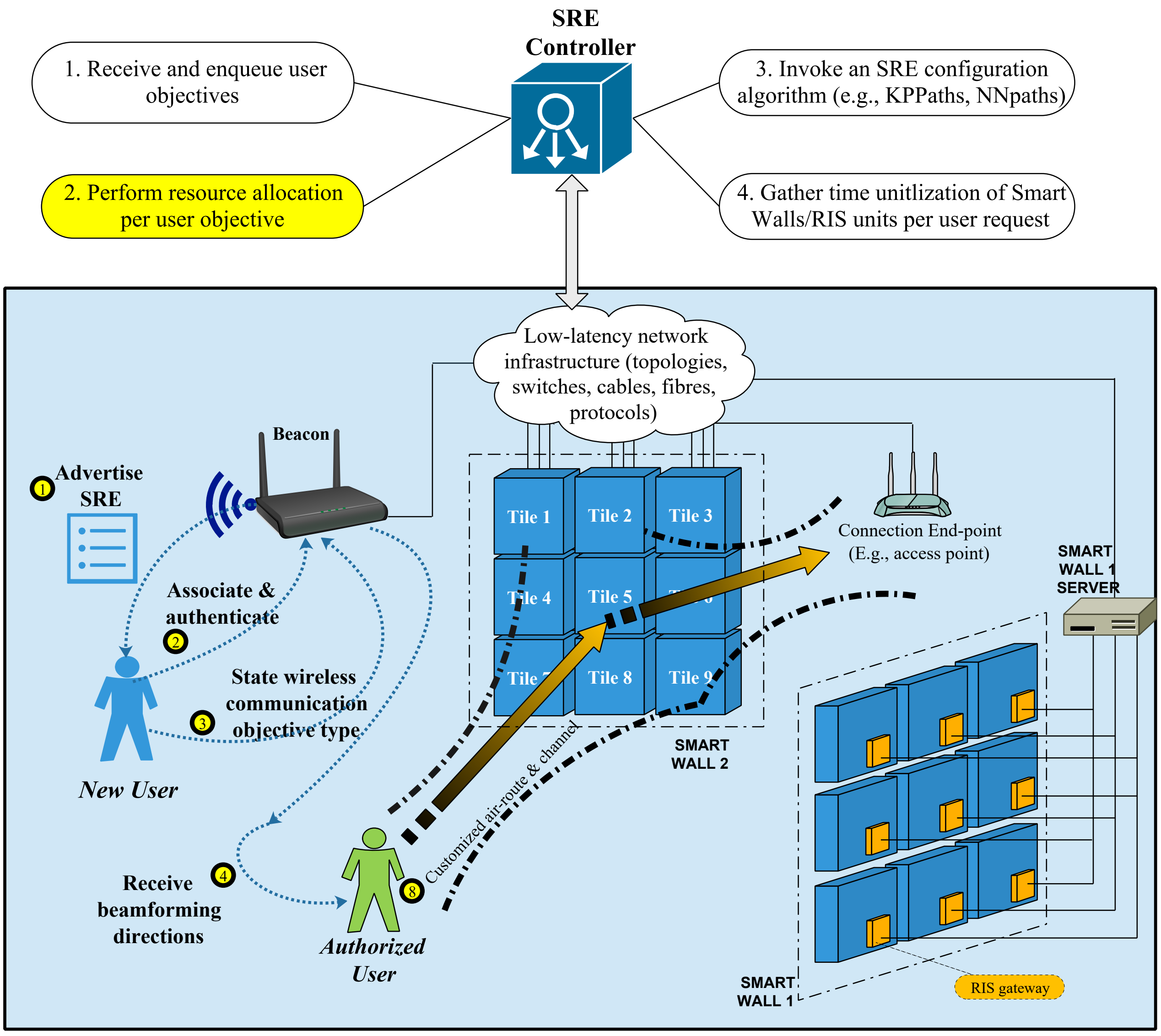

SRE Controller
1. Receive and enqueue user objectives
2. Perform resource allocation per user objective
3. Invoke an SRE configuration algorithm (e.g., KPPaths, NNpaths)
4. Gather time unitlization of Smart Walls/RIS units per user request
Low-latency network infrastructure (topologies, switches, cables, fibres, protocols)
Beacon
1 Advertise SRE
Associate & authenticate 2
State wireless communication 3 objective type
New User
Receive beamforming directions 4
Authorized User 8
Customized air-route & channel
Tile 1
Tile 2
Tile 3
Tile 4
Tile 5
Tile 7
Tile 8
Tile 9
SMART WALL 2
Connection End-point (E.g., access point)
SMART WALL 1 SERVER
SMART WALL 1
RIS gateway

Fig. 1 – Operation workflow of an SRE.

The following modeling assumptions are adopted, without loss of generality.

- We assume that the servers have no queuing capabilities. This task is handled exclusively by the controller.
- Whenever a server dispatches a given service request, it signals the controller for a new request/packet. The controller then assigns a new request to the server. This communication is instantaneous. In other words, the communication overhead between the controller and the servers is assumed to be a part of the requests.
- If all domains queues are empty, a server is fed with interruptible null-requests from a null-domain (Fig. 2). The null-domain is essentially an artificial structure, which facilitates the mathematical analysis. Its physical meaning is that the corresponding server is deactivated, e.g., to conserve energy, or that it enters a default type of service, such harvesting ambient EM waves.
- Every domain can be served by any server. The physical meaning of this statement is that the users are in line-of-sight with all the available smart walls, since scheduling makes sense in the case where users need to share commonly accessible resources. If a user is close only to one wall, then the assignment is trivial. Therefore, we inherently assume a physical setup where a set of users are within a large rectangular room, such as a waiting area in an airport.

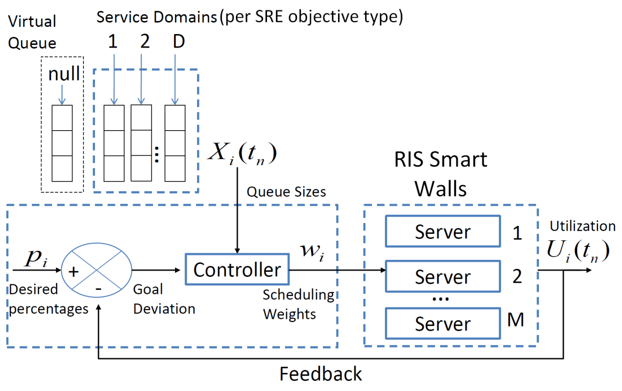


Fig. 2 – Overview of the proposed SRE resource slicing model, comprising a set of SLA-specific service domains representing the SRE wireless service types, and a set of smart wall servers. A controller enforces the SLAs using the reported smart wall utilization times as feedback.

Every domain i has a finite, aggregate arrival rate of requests, λ_i, which follows an unknown distribution with bounded inter-arrival times. Regarding the service process, we assume a non-i.i.d model, where the service time distribution can vary depending on the domain identifier, i. The service time of a request belonging to domain i, is described by a random variable S_i that follows some general distribution with mean $E[S_i]$ and finite variance, that we assume to be bounded. Therefore, there exists some finite S_i^{max} such that:

$$0 < S_i \leq S_i^{max} < \infty \qquad (1)$$

We note that the bounded inter-arrival times and the bounded service times are the only requirements for the proposed mathematical framework. We define the RIS wall utilization, $U_i^{\pi}(t)$, of domain i up to time t, under a request-to-server assignment policy π as:

$$U_i^{\pi}(t) \triangleq \frac{total\ service\ time\ up\ to\ t}{M \cdot t} \qquad (2)$$

When the following limit exists, the system is said to have reached a *steady state*, and the *allotted percent of RIS wall capacity* to domain i is then defined as:

$$\widetilde{U_i}(\pi) \triangleq \lim_{t \to \infty} U_i^{\pi}(t) \qquad (3)$$

The maximum achievable utilization for domain i, obtained when all of its requests can be served in the cluster for $t \to \infty$ is:

$$\tilde{p_i} = \frac{\lambda_i \cdot E[S_i]}{M} < 1 \qquad (4)$$

This can be thought of as the utilization obtained if there was a single domain in the system, utilizing all the resources. We note that servicing all the requests does not mean that the queue will be empty from one point and on, but rather that the probability of noticing queue sizes bigger than zero is negligible (convergence in probability). The following analysis will prove that all policies that fall under a negative-drift class of (Deficit

Table 1 – Summary of Notation

SLA	Service Level Agreement
SLO	Service Level Objective
$\mathcal{D}$	The set of service domains, representing SRE service types.
i	The domain index, $i \in \{1, \dots, D\}$.
$\mathcal{M}$	The set of smart wall servers.
m	The server index, $m \in \{1, \dots, M\}$.
λ_i	The arrival rate of requests at domain i.
S_i	The service times for domain i requests.
π	A request-to-server assignment policy.
$U_i^{\pi}(t)$	Smart wall utilization of domain i up to time t.
$\widetilde{U_i}(\pi)$	The allotted percent, for $t \to \infty$ (steady state).
$\tilde{p_i}$	The maximum achievable utilization.
p_i	The utilization goal (%) for domain i.
p_i^*	The actually achieved service (%).

Weighted Round-Robin) DWRR policies, reach a steady state whenever $\tilde{p_i} < p_i$ in non-work conserving mode of operation.
Here note that the physical meaning of non-work conserving operation in the SRE is that a user receives a message to wait until his request can been processed. This is necessary for ensuring that there is always the capability to assuredly serve all types of domain requests. To the contrary, work conservation means that the user is serviced without introducing artificial delays.
Furthermore, let l_n denote the idle time between round n and round $n+1$. Idle time can occur because there are no available requests in the system, or the system operates in non-work conserving mode and there are no available requests for domains for which their utilization is below the goal. We assume only systems for which

$$l_n < l < \infty, a.s., \qquad (5)$$

where $l \in \Re_+$ denotes the maximum observable idle time (or maximum observable inter-arrival time).
Finally, we denote as p_i the steady state percentage utilization goal, and the corresponding goal vector as $\mathbf{p} = (p_i)$, where $\sum_i^{|D|} p_i = 1$. The goal vector simply defines the requested percentage share of the aggregate wall server time per domain, e.g. $\mathbf{p} = (20\%, 30\%, 50\%)$, *assuming steady state*. The employed notation is summarized in Table 1.

3.3 Objectives

Our objective in this study, is to define control policies π that fulfill a number of criteria. We investigate policies that uphold the SLO for every domain i, under the assumptions of stochastic arrivals and unknown service time distribution. No workload prediction/estimation is assumed.
The objectives of this study are: a) the analysis of the conditions for the existence of a steady state (i.e. that

the limit in eq. *(3)* exists). b) The analysis of the feasibility space of the class of policies defined and the analysis of the redistribution mechanism of the residual smart wall service time required in order to achieve any custom but allowed steady state p_i^*. It will be shown that the redistribution mechanism greatly affects the actual percentage achieved.
Taking into account the redistribution process, the SLOs can be expressed as equations of the form:

$$\widetilde{U_i}(\pi) = \begin{cases} \tilde{p_i} & \text{, if } \tilde{p_i} \leq p_i \\ p_i^* & \text{, if } \tilde{p_i} > p_i, p_i^* \in [p_i, \tilde{p_i}] \end{cases} \tag{6}$$

An exemplary SLA can be described as follows: "*without any knowledge of statistics regarding the arrival and the service process in a set of smart wall servers, guarantee* 20% *of smart wall utilization to requests of class A,* 30% *of smart wall utilization to requests of class B and* 50% *of smart wall utilization to requests of class C in the long run. Also assume that no statistical knowledge is available regarding the correlation between the request size and the service time*".
Note that in the above definition there is no distinction between environments where mixed workloads are present and environments where the system is always saturated, meaning that there are always available requests by all the domains in the controller queues. In the case of dynamic arrivals, it is up to the policy to decide how to redistribute service in the case where a high priority domain must be served but has no available requests.
The study examines a class of Dynamic Weighted Round-Robin policies (DWRR), Π and show that every $\pi \in \Pi$ achieves steady state in the non-work conserving mode and upholds the domain SLOs, since the limit in eq. *(3)* exists and the differentiation objective in eq. *(6)* is satisfied. We also study the role of the redistribution mechanism in convergence of the policies in the work-conserving mode. In addition, we present a thorough analysis on the feasibility space of the policies, deducing the achievable service percentages, p_i^*, for every domain.

4. THE PROPOSED, SRE DOMAIN-SERVING POLICIES

4.1 Class Π of Dynamic Weighted Round Robin policies

We define the class Π of Dynamic Weighted Round Robin policies, which operates at rounds, with no loss of generality. A number of requests (weight), w_i^n, is serviced in round n, for every domain i. In order to enable the work conserving property, we decompose Π into two algorithmic components that we denote as **f** and **h**, respectively. Any policy $\pi \in \Pi$ obeys the following rules regarding **f**, while any arbitrary **h** algorithm can be used to enable the work conserving property and support the redistribution mechanism.
At the beginning of round n, at time instant $t_n \in \Re_+$, let $X_i(t_n)$ denote the queue size for domain i. We define:

$$\text{Class } \Pi = \begin{cases} \text{If } \exists i : U_i(t_n) < p_i, X_i(t_n) > 0 \rightarrow \textit{Algor. } \mathbf{f} \\ \text{If } \forall i : U_i(t_n) < p_i, X_i(t_n) = 0 \rightarrow \textit{Algor. } \mathbf{h} \end{cases} \tag{7}$$

The *algorithm* **f** *(negative drift)*, calculates the weights w_i^n as follows:

$$w_i^n = \begin{cases} 0, \text{if } U_i(t_n) > p_i \text{ or } X_i(t_n) = 0. \\ \min\{k(t_n, \pi), X_i(t_n)\}, \text{ if } U_i(t_n) \leq p_i, X_i(t_n) > 0 \end{cases} \tag{8}$$

In other words, w_i^n job requests are scheduled for domain i, where $k(t_n, \pi) \leq K$ and K is an arbitrary finite positive integer.
The class of *algorithms* **h** *(work conserving)* is used whenever $X_i(t_n) = 0, \forall i : U_i(t_n) < p_i$, in order to enable work conservation. In case where there are no available requests from all under-served ("suffering") domains, the class of policies Π can utilize any arbitrary scheduling decision, as long as there are available requests to do so. For example, the algorithm can choose one domain j at random from the domains for which $X_j(t_n) > 0$ and serve it for a finite number of times.
This algorithmic decomposition defines a class of policies for two reasons. Firstly, regarding the general operation of the policies (Algorithm **f**) there is no restriction on the number of requests served per round, as long as over-satisfied domains are blocked when there are no available requests from a suffering domain. The only requirement is imposed by the artificial limit K, used to avoid the case of assigning an infinite number of requests to some domain. Secondly, there is no limitation, on the operation of the **h**-algorithm, used for enforcing work conservation via redistribution of residual service slots. In general, the **f**-algorithm of the scheduling class is used to guarantee the objective defined in eq. *(6)* and define the feasibility space, while the **h**-algorithm defines the actual state achieved within the feasibility space.

5. THEORETICAL ANALYSIS

Under any policy $\pi \in \Pi$ (either work-conserving or non work conserving), the wall server utilization random variable evolves in time according to the following recursive formula:

$$U_i(t_{n+1}) = a^n \cdot U_i(t_n) + b_i^n \tag{9}$$

where $a^n = \frac{t_n}{t_{n+1}}$, $b_i^n = \frac{S_i^n}{t_{n+1}}$ and S_i^n denotes the aggregate service time that domain i received during round n, expressed as a sum of service time random variables. This recursion is aligned with the classical feedback ARX models [41], where in this case both terms a^n, b_i^n are stochastic and thus utilization is also a random variable. In fact, this is the reason that stochastic analysis tools are required to effectively control the system when the arrival and service process are unknown. Term a^n defines the way the history affects the system and b^n is related to the control input. Note that because of the bounded service-time and idle time assumptions, any round duration is also bounded. In addition, for any domain, the increase (or decrease) in utilization is also bounded according to the following lemma.

Lemma 5.1 (Utilization difference) *The utilization difference between the beginning and the end of each round for every domain is bounded according to:*

$$|U_i(t_{n+1}) - U_i(t_n)| \leq \frac{l + S_i^{max} + \sum_{j \in \mathcal{D}} S_j^{max}}{t_{n+1}} \qquad (10)$$

Proof 5.2 (Proof of lemma 5.1) *By definition, S_i^n denotes the service time received for domain i within the time interval $[t_n, t_{n+1}]$. Then using eq. (9):*

$$U_i(t) = \frac{t_n}{t_{n+1}} \cdot U_i(t_n) + \frac{S_i^n}{t_{n+1}} \qquad (11)$$

and, hence,

$$|U_i(t_{n+1}) - U_i(t_n)| = \left| \left(\frac{t_{n+1} - t_n}{t_{n+1}} \right) \cdot U_i(t_n) + \frac{S_i^n}{t_{n+1}} \right|. \qquad (12)$$

In the inequality (10), we additionally included the term l (an upper bound for all l_n), since idle time may be experienced due to lack of available requests from all the domains. In this case, it holds that

$$t_{n+1} - t_n = l_n + \sum_{i : w_i^n \geq 1} S_i^n, \qquad (13)$$

where l_n denotes the sum of "null requests" duration in all the servers, between round n and $n+1$. Via eq. (13) and by employing the upper bounds for the right-hand quantities of eq. (12), we can derive inequality (10):

$$|U_i(t_n + 1) - U_i(t_n)| = \left| \left(\frac{t_{n+1} - t_n}{t_{n+1}} \right) U_i(t_n) - \frac{S_i^n}{t_{n+1}} \right| \qquad (14)$$

$$\leq \left| \frac{t_{n+1} - t_n}{t_{n+1}} \right| \cdot U_i(t_n) + \left| \frac{K \cdot S_i^{max}}{t} \right| \qquad (15)$$

$$\leq \left| \frac{l + \sum_{j \in \mathcal{D}} K \cdot S_j^{max}}{t_{n+1}} \right| + \left| \frac{K \cdot S_i^{max}}{t_{n+1}} \right| \qquad (16)$$

$$= \frac{l + K \cdot S_i^{max} + \sum_{j \in \mathcal{D}} K \cdot S_j^{max}}{t_{n+1}} \qquad (17)$$

A standard way of proving the convergence of a work-conserving policy, is to show that eq. *(9)* converges in the general case. This essentially means, that the limit of eq. *(3)* exists and that the system will reach a steady state. The form of eq. *(9)* may predispose for convergence, since the control action has waning effect on utilization, as time progress ($b_i^n \to 0$ and $a^n \to 1$ for $t \to \infty$).

The following theoretical results can be deduced regarding the analysis of negative drift DWRR policies, under unknown service time and arrival processes statistics:

1. In the case of stochastic arrivals, when operating in non-work conserving mode, steady state exists for any policy $\pi \in \Pi$. Any policy converges with probability 1 to the minimum between the goal percentage and the maximum utilization service.
2. Assuming steady state, the feasibility space for any policy $\pi \in \Pi$, in the case of stochastic arrivals and all modes of operation, can be clearly defined.
3. In the case of stochastic arrivals, under work conserving mode of operation:

   [a.]

   (a) any policy $\pi \in \Pi : \tilde{p}_i \leq p_i$ convergences to $\tilde{p}_i$.

   (b) for any policy $\pi \in \Pi : \tilde{p}_i > p_i$ convergence point depends on the service redistribution algorithm $\mathbf{h}$.

In order to prove 1) we introduce a novel approach for proving convergence for the non-work conserving mode of operation, via the concept of null-domains. For point 2) we examine the feasibility space of the DWRR policies using a series of lemmas. We prove 3a) and we provide detailed analysis regarding point 3b).

5.1 Convergence and Feasibility Space Analysis

Before we proceed with the analysis, note that although the utilization evolution in transient state heavily depends on the time evolution and the past control actions (due to the effect of term a^n in eq. *(9)*), time evolution has no actual effect on convergence. It is the combination of the selected $\pi = (\mathbf{f}, \mathbf{h})$ policies that is responsible for the long-run behavior and when convergence can be achieved in the Cauchy sense [33]. See also [40] for a proof of the case of non-work conserving policies. For the case of work conserving policies under saturation, because of the use of the virtual queue operation, similar convergence properties also hold. According to the analysis presented in the following, we exploit results from stochastic analysis in order to prove convergence.

In the general case of unknown arrival process, $\tilde{p}_i$ (the maximum utilization percentage achieved in steady state according to eq. *(4)*, may be higher, equal or lower than the percentage goal requested. A number of lemmas and theorems establishes the feasibility ("capacity") region formally, for every policy $\pi \in \Pi$. For the ease of presentation, in subsection 5.1.1 we present the proof of theorem 5.3, and proceed to study theorem 5.5 in subsection 5.1.3.

5.1.1 Non-Work Conserving Mode: Convergence w.p. 1

As the following theorem presents, under non-work conserving mode, every negative drift DWRR policy reaches steady state. In addition, the capacity region is a vector.

Theorem 5.3 (Non-work conserving) *In non-work conserving mode of operation, under any policy $\pi \in \Pi$, the limit in eq. (3) exists $\forall i \in \mathcal{D}$. Moreover the objective defined in eq. (6) is satisfied. Thus:*

$$\widetilde{U_i}(\pi) = \min\{\tilde{p_i}, p_i\} \qquad (18)$$

In the non-work conserving mode, the policies operate according to the negative drift rules of algorithm $\mathbf{f}$ (i.e., we have $\pi = \mathbf{f}$) and no service redistribution is allowed. If there are no available requests from all the domains that are below their goal, the system goes to idle state. Essentially, the class only requires to block over-satisfied domains at the control instances and serve one or more of the under-utilized domains. In addition, it imposes no restriction on the number of requests that are served during a round.

The proof of convergence is based on the following observation. Because of the negative drift given to the over-satisfied domains at every control instant, in transient state, the utilization trajectory for any domain oscillates around its goal percentage p_i. Since the utilization difference between any two successive rounds is bounded, the oscillation is continuously decreasing and there exists some point in time when the system converges in the Cauchy sense. In the case where $\tilde{p_i} < p_i$ we prove that the trajectory approaches $\tilde{p_i}$.

To proceed with the proof, we introduce the null-domain concept of Fig. 2. Essentially, based on the saturated case analysis presented in [40], we model the idle time as a null-domain, and treat the system state as a saturated case variation.

Initially, since $\underset{k\to\infty}{lim\,sup}\frac{l+S_i^{max}}{t_k} = 0$, we can write that:

$$\underset{k\to\infty}{lim\,sup}(p_i^*) + \underset{k\to\infty}{lim\,sup}\frac{l+S_i^{max}}{t_k} = p_i \qquad (19)$$

In addition, for any domain i and round k, the utilization is upper bounded by:

$$U_i(t_k) \leq p_i + \frac{S_i^{max}}{t_k}. \qquad (20)$$

This bound derives from eq. *(12)* and holds for the general case of stochastic arrivals and non-work work conserving mode of operation since all over-served domains are blocked by the policy (even if a number K of its requests were served during the last round). Then, we can write that:

$$\underset{k\to\infty}{lim\,sup}U_i(t_k) \overset{\text{eq. }(20),(19)}{\leq} \min\{\tilde{p_i}, p_i\} \text{ a.s} \qquad (21)$$

where $\tilde{p_i}$ is the maximum achievable utilization for domain i. This quantity is related with the maximum-overshoot in classical control theory [41].

Proof 5.4 ([Proof of Theorem 5.3) *We introduce the notion of a virtual queue holding null requests. The queuing structure of the controller now becomes one queue per domain plus one virtual queue for null requests (see Fig.2). A null request can be thought of as a series of* deactivate *commands. This is an artificial technicality, to model idle time in the system. Moreover, we assume that null requests always exist in the virtual queue.*

In order to examine the case of idle state, say that at the end of some round there are no available requests from any of the domains ($X_i(t_n) = 0, \forall i \in \mathcal{D}$). Then the scheduler selects a null request from the virtual queue and sends it to the requesting server. We assume that the server preemptively replaces null requests. If a null request is in service and some domain's request arrives to the system, the controller sends the request to the idle server. Thus the null requests can be treated essentially as an additional domain in the system.

In the general case, not all the domains converge to the goal percentage defined; some converge to the goal, others to the maximum achievable utilization. For the ease of presentation, we define the following partition of the domains set:

- *$\mathcal{D}_1$: includes all the domains for which $\tilde{p_i} < p_i$.*
- *$\mathcal{D}_2$: includes all the domains for which $\tilde{p_i} \geq p_i$. Then $\mathcal{D}_2 = \mathcal{D} - \mathcal{D}_1$.*
- *$\mathcal{D}_3$: a singleton set with the null requests domain.*
- *$\mathcal{F} : \mathcal{F} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$.*

Now consider that the idle domain receives all the service time that cannot be utilized due to lack of requests from the "suffering" domains in $\mathcal{D}_1$. Let $C(t_n)$ denote the utilization of the null domain, then we can write:

$$\underset{k\to\infty}{lim\,sup}C(t_n) = \sum_{i\in\mathcal{D}_1}(p_i - \tilde{p_i}) \qquad (22)$$

which is the upper bound for the null requests domain utilization. Then, the following is true:

$$\liminf_{k\to\infty} U_i(t_{k+1}) = \liminf_{k\to\infty}(1 - \sum_{j\in\mathcal{F}} U_j(t_{k+1})) \quad (23)$$

$$\geq 1 + \liminf_{k\to\infty}(-\sum_{j\in\mathcal{F}} U_j(t_{k+1}))$$

$$= 1 - \limsup_{k\to\infty} \sum_{j\in\mathcal{F}} U_j(t_{k+1})$$

$$\geq 1 - \sum_{j\in\mathcal{F}} \limsup_{k\to\infty} U_j(t_{k+1})$$

$$\geq 1 - [\sum_{j\in\mathcal{D}_1} \tilde{p_j} + \sum_{j\in\mathcal{D}_2} p_j + \limsup_{k\to\infty} C(t_n)]$$

Case 1: *$i \in \mathcal{D}_1$*

$$\liminf_{k\to\infty} U_i(t_{k+1}) \quad (24)$$

$$\geq 1 - [\sum_{j\in\mathcal{D}_1} \tilde{p_j} + \sum_{j\in\mathcal{D}_2} p_j + \sum_{j\in\mathcal{D}_1} (p_j - \tilde{p_j})]$$

$$= 1 - [(\sum_{j\in\mathcal{D}_1} \tilde{p_j}) - \tilde{p_i} + \sum_{j\in\mathcal{D}_2} p_j + \sum_{j\in\mathcal{D}_1} (p_j - \tilde{p_j})]$$

$$= 1 - [\sum_{\mathcal{D}_1\cup\mathcal{D}_2} p_j - \tilde{p_i}] = \tilde{p_i}$$

Case 2: *$i \in \mathcal{D}_2$*

$$\liminf_{k\to\infty} U_i(t_{k+1}) \quad (25)$$

$$\geq 1 - [\sum_{j\in\mathcal{D}_1} \tilde{p_j} + \sum_{j\in\mathcal{D}_2} p_j + \sum_{j\in\mathcal{D}_1} (p_j - \tilde{p_j})]$$

$$= 1 - [\sum_{j\in\mathcal{D}_1} p_j + \sum_{j\in\mathcal{D}_2} p_j] = p_i$$

According to to these equations,

$$\liminf_{k\to\infty} U_i(t_{k+1}) \geq \min\{p_i, \tilde{p_i}\} = \limsup_{k\to\infty} U_i(t_{k+1})$$

But, axiomatically, $\liminf_{k\to\infty} U_i(t_{k+1}) \leq \limsup_{k\to\infty} U_i(t_{k+1})$ and therefore we conclude that $\liminf_{k\to\infty} U_i(t_{k+1}) = \limsup_{k\to\infty} U_i(t_{k+1})$ and so the limit exists in any case.

We proceed to study the feasibility space for all DWRR policies in terms of achievable domain service vectors, with the focus being on work conserving modes of operation.

5.1.2 Work Conserving Mode

In contrast to the non-work conserving mode, under work conserving mode, the capacity region is a hyperplane, as stated by the following theorem. Note that we restrict our attention to the policies $\pi \in \Pi$ that reach steady state, since the convergence depends not only on the $\mathbf{f}$-algorithm, but to the $\mathbf{h}$-algorithm also. Thus convergence analysis is specific to every $\pi = (\mathbf{f}, \mathbf{h})$ pair.

Theorem 5.5 (Feasibility Region) *The feasibility space is a hyperplane defined by the intersection of hyperplanes defined by the following constraints:*

$$\sum_{i=1}^{|D|} \widetilde{U_i}(\pi) \leq 1, \widetilde{U_i}(\pi) \leq \tilde{p_i} \quad (26)$$

and

$$\widetilde{U_i}(\pi) = \tilde{p_i}, \mathit{if}\ \tilde{p_i} < p_i,\ \widetilde{U_i}(\pi) \geq p_i, \mathit{if}\ \tilde{p_i} \geq p_i \quad (27)$$

Proof 5.6 (Proof of Theorem 5.5) *The constraints defined in eq. (26) are irrelevant to the goal vector, while the constraints in eq. (27) derive from the relationships between the goal vector $\boldsymbol{p} = (p_i)$ and $\tilde{\boldsymbol{p}} = (\tilde{p_i})$. The first two constraints in eq. (26) state that the utilization achieved in steady state cannot exceed the physical maximum utilization of the system (which equals one) and the third constraint simply states that a domain cannot exceed the maximum utilization (Definition (2)).*
In contrast to the previous case of non-work conserving mode of operation, where the system may experience idle time even when there are available requests for service, in the work conserving mode of operation, service redistribution takes place and is up to policy $\mathbf{h}$ to achieve convergence. According to our presentation, the feasibility space of a class is actually defined by the $\mathbf{f}$-algorithm and the final performance depends on $\mathbf{h}$-algorithm selection. Constraints in eq. (27) are proven in the two lemmas that follow.

Lemma 5.7 (Domains away from the goal) *In the case where for some domain i, $\tilde{p_i} < p_i$ (the percentage goal p_i was ambitious), under any policy $\pi \in \Pi$, steady state exists and the achievable percentage $\widetilde{U_i}(\pi)$ equals $\tilde{p_i}$.*

The proof of this lemma is analogous to case 1, presented in the non-work conserving mode of operation and thus is omitted. The difference in the work conserving case, is that there is no "null" domain and we simply don't know how the extra service $C(t_n)$ is redistributed in domains that can utilize extra service.

Lemma 5.8 (Domains reaching their goals) *In the case where for some domain i, $\tilde{p_i} \geq p_i$ (the goal is feasible), under any policy $\pi \in \Pi$, $\widetilde{U_i} \geq p_i$ is true.*

We provide the proof of this lemma.

Proof 5.9 (Proof of lemma 5.8) *Assume that there exists some domain $i \in \mathcal{D}_2$ (includes all the domains for which $\tilde{p_i} \geq p_i$) and consider some very large round n_0, for which $U_i(t_n) < p_i$ is true $\forall n > n_0$. This means that the policy can serve less than it is expected, but also less than it could have received. According to the policy definition, this means that for any successive round $n > n_0$, the policy will set constantly $w_i^n = 1$ (independent in which other domains will also participate in all*

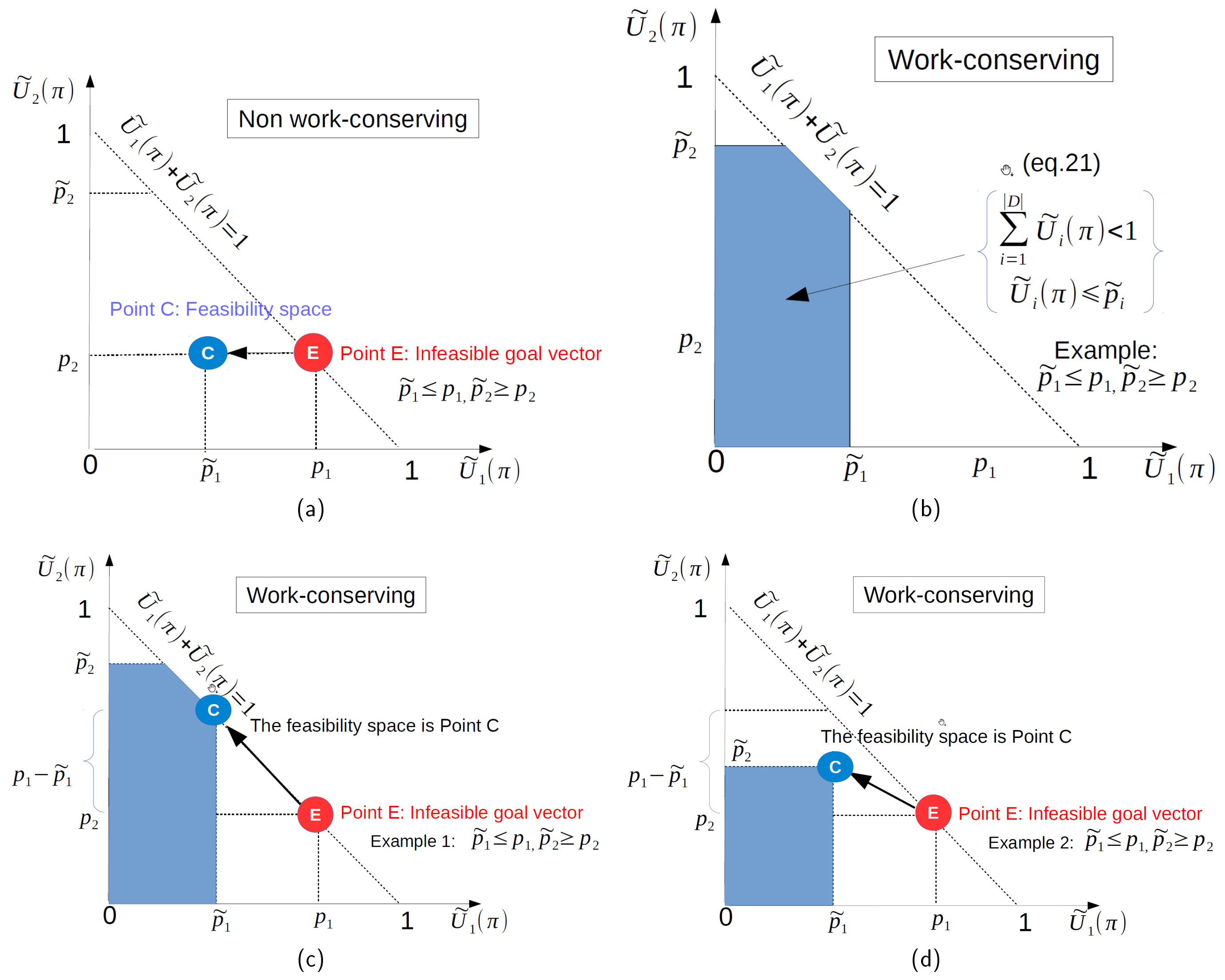

Non work-conserving
Work-conserving
$\tilde{U}_1(\pi)+\tilde{U}_2(\pi)=1$
Point C: Feasibility space
Point E: Infeasible goal vector
$\tilde{p}_1 \leq p_1, \tilde{p}_2 \geq p_2$
(eq.21)
$\sum_{i=1}^{|D|} \tilde{U}_i(\pi) < 1$
$\tilde{U}_i(\pi) \leq \tilde{p}_i$
Example:
The feasibility space is Point C
Example 1: $\tilde{p}_1 \leq p_1, \tilde{p}_2 \geq p_2$
Example 2: $\tilde{p}_1 \leq p_1, \tilde{p}_2 \geq p_2$
$p_1 - \tilde{p}_1$
(a)
(b)
(c)
(d)

Fig. 3 – Feasibility regions and scheduling dynamics (requested goal vector versus achieved goal vector) for work-conserving and non work-conserving operation.

or to a subset of the new rounds). The reason is that there will be unserved requests of this domain that can participate in the following rounds. This makes $l_n = 0$ *for all successive rounds. Under the same reasoning, all the domains that will participate in the following rounds and on will be served constantly. Assume that we define as* $\mathcal{B} \subseteq \mathcal{D}_2$ *as the set in which all these domains belong.*

But this means that all other domains that do not belong in this "special" set $\mathcal{B}$ *cannot exceed their targets (and receive the "lost" service from domains in* $\mathcal{B}$*), since whenever they do so they get blocked by the policy. Then, summing up the utilizations of all the domains, we get that for any* $t > t_{n_0}$*:*

$$\sum_{i\in\mathcal{B}(t)} U_i(t_n) < \sum_{i\in\mathcal{B}(t)} \tilde{p}_i < \sum_{i\in\mathcal{B}(t)} p_i$$

$$\sum_{i\notin\mathcal{B}(t)} U_i(t_n) \leq \sum_{i\notin\mathcal{B}(t)} p_i,$$

and thus:

$$\sum_{i=1}^{D} U_i(t_n) = \sum_{i\in\mathcal{B}} U_i(t_n) + \sum_{i\notin\mathcal{B}} U_i(t_n) < 1 \quad (28)$$

But this means that there must be idling, which is a contradiction. Thus, it must hold that for any domain $i \in \mathcal{D}_2 : \tilde{U}_i(\pi) \geq \tilde{p}_i$.

Note that in this case we cannot guarantee that the output utilization actually reaches steady state, or convergences to some point, since this depends on the selected

h-*algorithm. The above proofs, nevertheless, guarantees that the utilization trajectory will move above p_i independently of the selected* **h**-*algorithm. This concludes the proof of theorem 5.5.*

The following theorem is a direct consequence of lemmas 5.7 and 5.8 and states that all the policies that belong in class Π can be used to satisfy the objective in eq. *(6)*.

Theorem 5.10 *All the policies that belong to class Π can be used to satisfy the objective in eq. (6).*

For example, as shown in Fig. 3, in both cases of non work-conserving and work-conserving mode, in the case of two domains, the goal lies over the line that joins points (0,1) and (1,0) and the feasibility region is a single point. The point lies over the line in the case that the available load can support the maximum utilization sum that equals one. In the case where a very high goal was set for both domains, then the performance point would be a point that lies below the line that is defined by $\sum_{i\in\mathcal{D}} \widetilde{U_i}(\pi) = 1$.
For any policy $\pi \in \Pi$ under non work-conserving mode a domain i receives exactly the $\min\{p_i, \tilde{p}_i\}$ as can be seen in Fig.3a. The feasibility region, in the general case of stochastic arrivals, is defined by equations *(26)* and *(27)*. For example the shaded region in Fig. 3b is defined by the inequalities of eq. *(26)*. Under work-conserving operation a domain cannot receive less than the $\min\{p_i, \tilde{p}_i\}$ as we can see in examples presented in Figs. 3c and 3d where different relationship exists between $\tilde{p}_i$ and p_i. In all cases, the presented class Π of policies guarantees that the objective in eq. *(6)* is satisfied.

5.1.3 Details on the Load Redistribution Mechanism

In the work conserving mode, for policies $\pi \in \Pi$, $\pi = (\mathbf{f}, \mathbf{h})$ the normal operation is performed according to algorithm **f** and the service redistribution is performed according to algorithm **h**. While the operation of algorithm **f** guarantees the validity of the constraints in eq. *(27)* and the definition of the feasibility space, the algorithm **h** is responsible for the convergence properties and the exact points p_i^* achievable by the policy in effect.
The total service cycles that can be redistributed, are offered by domains $i \in \mathcal{D}_1$ to domains $j \in \mathcal{D}_2$ and (as analyzed in section 5.1.1) equals $\sum_{i\in\mathcal{D}_1}(p_i - \tilde{p}_i)$. Since we consider only work conserving policies, the service that is not utilized by some domains will be redistributed to other domains. According to the previous analysis, the actual performance is the following:

$$\widetilde{U_i} = \tilde{p}_i \text{ , if } i \in \mathcal{D}_1 \text{ (lemma 5.7)} \qquad (29)$$
$$\widetilde{U_i} = p_i + C_i \text{ , if } i \in \mathcal{D}_2 \text{ (lemma 5.8)}$$

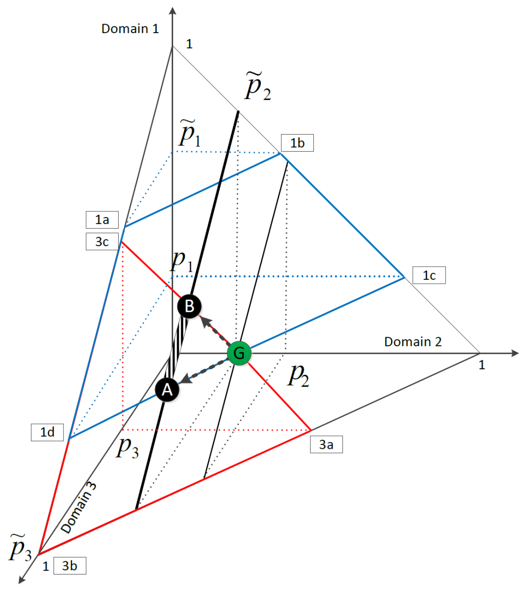


Fig. 4 – A three domains scenario, where $\tilde{v_2} < p_2$ and redistribution must take place. The final performance depends on algorithm **h** rules.

Note that that as the feasibility space was clearly defined, the total extra service that will be redistributed depends only on vectors $\mathbf{p} = (p_i)$ and $\tilde{\mathbf{p}} = (\tilde{p}_i)$ and is irrelevant of the actual weight (value $k(t_n)$) selected by algorithm **f** and algorithm **h**. It is the extra service C_i received by each domain, that depends on algorithm **h**. The independence between **f** and **h** algorithms can be expressed by the following lemma.

Lemma 5.11 (Service redistribution) *Under any policy in class Π and $\forall i \notin \tilde{\mathcal{B}}$, $C_i(t_n)$ redistribution depends only on the scheduling algorithm* **h** *and is irrelevant to algorithm's* **f** *$k(t_n)$ selection.*

Proof 5.12 (Proof of lemma 5.11) *In order to prove the above lemma, we will show that the service redistribution $C_i(t_n)$ does not depend on the number $k(t_n)$ selected at each control instant at the end of round t_n by algorithm* **f**. *Assume a system with two domains and two policies $\pi_1(\mathbf{f}_1, \mathbf{h}_1), \pi_2(\mathbf{f}_2, \mathbf{h}_2)$ such that $\mathbf{h}_1 = \mathbf{h}_2$. Both $\mathbf{h}_1, \mathbf{h}_2$ choose the other domain if it has requests, while $k_2(t_n) = k_1(t_n) + k$, where k is any integer that can be supported by the queueing dynamics. The same reasoning can be followed for D domains. Also assume that $\tilde{v_1} < p_1$ and $\tilde{v_2} > p_2$ so that service redistribution will occur, since domain 1 has not sufficient load to support the goal percentage. However, in both policies, lemma 5.1 guarantees that $\widetilde{U_{1+1}\pi_1} = \widetilde{U_{1+1}\pi_2} = \tilde{v_1}$, thus in both cases $C_2^{\pi_1} = C_2^{\pi_2} = p_1 - \tilde{v_1}$ and so $\widetilde{U_{2+1}\pi_1} = \widetilde{U_{2+1}\pi_2} = p_2 + \min\{C_2, \tilde{v_2} - p_2\}$. The fact that, under policy π_2, at every round, domain 1 was given more service cycles, plays no role in the long run. The minimum in this equality simply states that domain 2 cannot receive extra service that cannot be supported by its load.*

In Fig. 4, a visual representation of the feasibility space is given, in the case where three domains are competing for service using **f** rules. In this example, domain 3 can utilize all the capacity, $\tilde{v_3} = 1$, for domain 1, $\tilde{v_1} > p_1$, while for domain 2, $\tilde{v_2} < p_2$. According to the class of policies definition, theorem 5.5 and lemmas 5.7 and 5.8, load redistribution will take place while the feasibility space of the class of policies is defined as the intersection of the following hyperplanes: hyperplane 1 defined by points $(1a, 1b, 1c, 1d)$ that presents the utilization area domain 1 can exploit, hyperplane 2 defined by the line defined by restriction $\tilde{v_2}$ and hyperplane 3 (triangle) defined by points $(3a, 3b, 3c)$ that presents the utilization area domain 3 can exploit. This intersection is the line that joins points A and B in the figure.
Depending on the service redistribution algorithm, the long run performance and steady state (if it can be achieved), clearly depends on the selected **h**-algorithm. For example, if domain 1 utilizes all the extra service, the long run utilization point is point B (where domain 3 receives no extra service), while the long run utilization point is point A, where domain 3 receives all extra service. Intermediate points can be obtained, if the appropriate redistribution algorithm is selected. Nevertheless, as this study assumes the absence of statistical knowledge regarding the arrival and service process, it is not easy to find such a policy. Note that a second DWRR can be also deployed in order to precisely allocate this extra service as follows. After a long time has elapsed and $\mathcal{D}_1$ is not empty, set $\sum_{i\in\mathcal{D}_1} p_i - \tilde{p_i} > 0$. Then we can create a second DWRR list of weights with only $\mathcal{D}_2$ domains, define new goals regarding the extra service and redistribute service according to the policy's negative drift rules.

5.2 Concluding Remark

The following theoretical results can be collectively deduced regarding the analysis of negative drift DWRR policies, under unknown service time and arrival process statistics:

1. In the case of saturated arrivals, $\forall\pi \in \Pi$ converges with probability 1 (w.p. 1) to the goal percentage, p_i.
2. In the case of stochastic arrivals, when operating in non-work conserving mode, steady state exists $\forall\pi \in \Pi$ and $\widetilde{U_i}(\pi) = \min\{\tilde{p_i}, p_i\}$.
3. In the case of stochastic arrivals, under work conserving mode of operation and assuming steady state:

   [a.]

   (a) $\forall\pi \in \Pi : \tilde{p_i} \le p_i$, $\widetilde{U_i}(\pi)$ convergences to $\tilde{p_i}$.

   (b) $\forall\pi \in \Pi : \tilde{p_i} > p_i$, then $\widetilde{U_i}(\pi) \ge \tilde{p_i}$ and the precise convergence point depends on the service redistribution algorithm **h**.

6. EVALUATION

In this section, we present the evaluation of the DWRR policies using an extensive set of simulations. The goal is to verify the theoretical results obtained in previous sections, while also provide to the administrators interested to apply DWRR, a better representation of the actual DWRR performance. In the evaluation we add a comparison to the scheduler presented in [40], which assumes a system with stochastic arrivals, where a domain was selected at random when redistribution should be performed (we refer to this policy as *OBG* (serve Only Below Goal)). Here we focus on the stochastic arrivals case and present an indicative set of simulations regarding the following:

1. convergence of the policies in non-work conserving mode of operation and comparison with other schemes.
2. verification of the feasibility space analysis, by providing points outside the feasibility space and presenting the points where the system converges.
3. understanding of the **h**-policy effects on final performance, using the goals that fall outside the feasibility space, under different **h** algorithms.

A custom, JAVA-based simulator was build to perform the following experiments.

6.1 Convergence of the policies in work conserving mode of operation and comparison with other schemes

In both, saturated systems and systems with stochastic arrivals, DWRR can be used to satisfy the differentiation objectives. The only requirement to reach a predefined goal is that the goal is feasible. Figs. 5a and 5b are used to verify the DWRR performance in the following simulation scenarios, where the *OBG* is selected as the member policy from the DWRR class defined (in *OBG* when domain i is selected then $w_i = 1$ [40]).
In Fig.5a the goal vector defines equal service shares for three domains $\{33\%, 33\%, 33\%\}$ where in Fig. 5b the goal vector was set equal to $\{50\%, 30\%, 20\%\}$. In both scenarios, Poisson arrival rates where used for all domains, with mean rate vector equal to $\{0.5\%, 0.4\%, 0.3\%\}$ and exponential service time distribution with means $\{3.0\%, 2.0\%, 1.0\%\}$. Note that the reason for this selection was to use different workload characteristics per domain. As we can see in both cases, DWRR are able to differentiate according to the requested goal vector since the policies adapt in real time to workload variations and to the stochastics of the system.
In Fig. 5c a comparison is presented between DWRR (OBG) [40], WRR [42], Random scheduling and Credit Scheduling [8], with the workload and the goal the ones presented in Fig.5b,($\{50\%, 30\%, 20\%\}$). In this figure

the total absolute deviation is presented $\sum_{i=1}^{D} |U_i(t_n) - p_i|$. WRR is a static policy where the scheduling weights are defined according to the goal vector, so at each round, 5 requests where served by domain 1, 3 from domain 2 and 2 from domain 3 (if there were available); in Random scheduling a domain is selected at random; where we use a variation of the Credit scheduling policy to compare it with OBG. As we can see, WRR scheduling completely fails to align with the objective, since the service time per domain is different. In fact, in this example random scheduling performs even better. Adaptive policies like *OBG* and Credit can be used to satisfy the objective. DWRR may be more accurate when bursty traffic phenomena and ON/OFF periods are experienced. Nevertheless, this increased accuracy comes at the expense of increased overhead, when the application environment requires for I/O handling and preemptive operations.

In Fig. 5d the effects of increasing the domains in the system are presented. We present the case where we want to load balance the traffic between 2, 4 and 8 domains, respectively, and the goal is feasible. As we can see, increasing the number of domains results in increased convergence time. Additionally, increasing the number of domains, results in less scheduling opportunities per round per domain and, thus, increased convergence time.

According to the class of policies definition, algorithm **f** sets no restriction on the number of requests actually served per round when a "suffering" domain is selected. In Fig. 5e we can see the effect of setting the weight of a suffering domain equal to the queue size of this domain. (E.g., if a domain is below its goal and has 5 requests in queue at the control instant t_n, then set its scheduling weight equal to 5 for round n). The setup of the experiment is the same as in Fig. 5b. As we observe, this greatly affects the convergence period. The reason is that by giving increased gain at each control instant, we violently change the trajectory giving a different tendency to the system. Since the gain cannot be controlled by classical methods (proportional control, integrators etc. [41]), increasing the gain indirectly by large weights, leads to increased convergence times. In Fig. 5f we compare the *OBG* policy with the case where the maximum weight can be allocated according to policy **f** definition. Furthermore to stress the algorithms we create random ON-OFF [43] periods that essentially emulate bursts of requests. Evidently, the convergence properties of delayed decision making results in slow convergence.

Remark: From a practical perspective, a major concern when applying DWRR policies, is the increased overhead requirements, since the policy requires to track at the the end of every round the service per domain and update statistics accordingly, in order to make the control action for the next round. Selecting different weights is a way to track only the context switching instances (when we start to serve the other domains) and thus reduce overhead. Nevertheless, this comes at increased convergence periods.

6.2 Feasibility spaces

As already analyzed theoretically, whenever a goal is feasible, DWRR class of policies guarantees that in the long run the objective is satisfied. Under some statistical assumptions regarding the arrival process or when in non-work conserving operation, it will actually receive exactly its goal percentage. Nevertheless, if for some domain i the goal is outside the feasibility region, then under DWRR the maximum achievable percentage $\tilde{p_i}$ is achieved and the difference $p_i - \tilde{p_i}$ will be utilized by some other domain according to the rules set by policy **h**. We can see this behavior in Fig. 6a where the goal is on purpose set outside the feasibility region. In this experiment, a very high goal was set (the top bullet point in the figure) that defines the vector $(10\% - 20\% - 70\%)$. The arrival and service process where such that could not satisfy this objective. As expected theoretically, by theorem 5.5, the feasibility space is the dashed line defined by the intersection of hyperplanes set by equations *(26)* and *(27)*. Note that, the actual percentage achieved, depends on the redistribution algorithm **h** (in this case was *OBG* algorithm, that serves 1 request per domain when below goal, one domain at random when there are no available requests in the system from domains below the goal). DWRR scheduling guarantees that even when workload variations exists, ON/OFF periods or random disturbances, the feasibility space will be always given by eq. *(26)* and *(27)* and so a minimum service is guaranteed per class.

6.3 Load Redistribution Mechanism

In Fig. 6b we present the effect of using different redistribution policies, in the case of infeasible vector goal. The goal vector was set equal to $(70\% - 30\% - 20\%)$ and is denoted with the leftmost dot in the figure that nevertheless is outside the feasibility region. As in the previous case, the goal was set on purpose outside the feasibility region in order to perform redistribution of service. We compare DWRR in the case of four redistribution algorithms. Whenever an over-satisfied domain is served in order to be work conserving: a) choose one at random; b) choose the one with the minimum queue size; c) choose the one with the maximum queue size. As we can see, the feasibility space is the one defined according to theorem 5.5 and is irrelevant of the redistribution algorithm. The application of **h** is responsible for the actual percentage obtained.

An interesting observation can be made regarding the load redistribution mechanism. As hinted by Fig. 6b the "minQueue" redistribution algorithm exhibits a wider variation in terms of converged state placement over the feasibility space. To study this issue further, we study the form of the trail produced by "minQueue", "maxQueue" and "random" over 100 simulation runs with the same configuration as Fig. 6b. For every run **h**-algorithm, we log the average distances of the trail points from the line defined by the points $\emptyset$ and converged state. The results over the 100 runs form the boxplot of Fig. 7. In essence, the employed distance metric expresses the "wandering" rate of the system's trail towards convergence. Ideally, a perfect $\mathbf{f}/\mathbf{h}$ algorithm combination would yield a linear trail from the point $\emptyset$ to the final converged state. The "maxQueue" and "Random" **h**-algorithms yield approximately the same degree of divergence from linearity. However, the "minQueue" approach exhibits a considerably higher divergence. As shown in Fig. 6b, the "minQueue" approach is more exploratory in terms of convergence point within the feasibility space. By taking subtler scheduling decisions (choosing the smaller queues), "minQueue" is able to achieve a wider set of final converged states. On the other hand, "maxQueue" and "random" take more crude scheduling decisions, resulting into smaller divergence in terms of converged states within the feasibility space. In other words, the **h**-algorithms also regulate the feasibility space "exploration" rate of the system.

7. CONCLUSION

In this work, we introduced a model for performing resource slicing and scheduling of resources in smart radio environments (SREs). SREs incorporate special devices called reconfigurable intelligent surfaces (RIS) which can alter programmatically the way they interact with impinging electromagnetic waves. Thus, wireless propagation in an SRE become a programmable resource and, since the air is naturally shared among devices, a slicing model is required. Thus, the proposed model models sets of RIS units as servers, and organizes the types of SRE services as domains that can receive and enqueue requests. Scheduling can then be performed by policies that map requests to servers. The study showcased the use of this model by studying a class of negative drift Dynamic Weighted Round Robin policies, that is able to guarantee specific SRE shares between competing domains. We provided a general mathematical framework where the class of policies requires no statistical knowledge regarding the arrival and service process and can be used in multiple application scenarios. The following conclusions hold for the proposed class of negative drift DWRR policies. In the case of saturated arrivals, any DWRR policy converges with probability 1 to the goal percentage. In the case of stochastic arrivals, when operating in non-work conserving mode, any policy converges with probability 1 to the minimum between the goal percentage and the maximum utilization service. The feasibility space for any DWRR policy, in the case of stochastic arrivals and all modes of operation, can be clearly defined. Finally, in the case of stochastic arrivals, under work conserving mode of operation the differentiation objective is still satisfied, nevertheless the final performance depends on the redistribution mechanism.

ACKNOWLEDGMENT

This work was funded by projects WISAR (Foundation for Research and Technology–Synergy Grants 2022) and COLLABS (EU Horizon 2020, GA 871518).

REFERENCES

[1] C. Liaskos, S. Nie, A. Tsioliaridou, A. Pitsillides, S. Ioannidis, and I. Akyildiz, "A new wireless communication paradigm through software-controlled metasurfaces," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 162–169, 2018.

[2] M. Di Renzo, A. Zappone, M. Debbah, M.-S. Alouini, C. Yuen, J. de Rosny, and S. Tretyakov, "Smart radio environments empowered by reconfigurable intelligent surfaces: How it works, state of research, and road ahead," *arXiv preprint arXiv:2004.09352*, 2020.

[3] E. Basar, M. Di Renzo, J. De Rosny, M. Debbah, M.-S. Alouini, and R. Zhang, "Wireless communications through reconfigurable intelligent surfaces," *IEEE access*, vol. 7, pp. 116 753–116 773, 2019.

[4] C. Liaskos, A. Tsioliaridou, A. Pitsillides, S. Ioannidis, and I. Akyildiz, "Using any surface to realize a new paradigm for wireless communications," *Commun. ACM*, vol. 61, pp. 30–33, 2018.

[5] The VISORSURF project consortium, *The Internet of Materials.* CRC Press, ISBN 987-0-367-45738-9, 2021.

[6] C. Liaskos, A. Tsioliaridou, S. Nie, A. Pitsillides, S. Ioannidis, and I. Akyildiz, "On the network-layer modeling and configuration of programmable wireless environments," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1696–1713, 2019.

[7] D. P. Bovet and M. Cesati, *Understanding the Linux Kernel: from I/O Ports to Process Management.* " O'Reilly Media, Inc.", 2005.

[8] D. Ongaro, A. L. Cox, and S. Rixner, "Scheduling i/o in virtual machine monitors," in *Proc. of ACM SIGPLAN/SIGOPS'08*, pp. 1–10.

[9] E. Gelenbe and I. Mitrani, *Analysis and Synthesis of Computer Systems.* World Scientific, 2010, vol. 4.

[10] D. Schanzenbach and H. Casanova, "Accuracy and responsiveness of cpu sharing using xen's cap values," 2008.

[11] L. Cherkasova, D. Gupta, and A. Vahdat, "Comparison of the three cpu schedulers in xen," *Perf. Evaluation Review*, vol. 35, no. 2, p. 42, 2007.

[12] C. Liaskos, A. Tsioliaridou, S. Ioannidis, A. Pitsillides, and I. Akyildiz, "Next generation connected materials for intelligent energy propagation in multiphysics systems," *IEEE Communications Magazine*, 2021.

[13] C. Liaskos, G. Pyrialakos, A. Pitilakis *et al.*, "The internet of metamaterial things and their software enablers," *ITU Journal of Future and Emerging Technologies*, vol. 1, no. 1, pp. 1–23, November, 2020.

[14] O. Quevedo-Teruel, H. Chen, A. Díaz-Rubio, G. Gok, A. Grbic, G. Minatti, E. Martini, S. Maci, G. V. Eleftheriades, M. Chen *et al.*, "Roadmap on metasurfaces," *Journal of Optics*, vol. 21, no. 7, p. 073002, 2019.

[15] C. Liaskos, L. Mamatas, A. Pourdamghani, A. Tsioliaridou, S. Ioannidis, A. Pitsillides, S. Schmid, and I. F. Akyildiz, "Software-defined reconfigurable intelligent surfaces: From theory to end-to-end implementation," *Proceedings of the IEEE*, 2022.

[16] H. Wymeersch and B. Denis, "Beyond 5g wireless localization with reconfigurable intelligent surfaces," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[17] Ö. Özdogan, E. Björnson, and E. Larsson, "Intelligent reflecting surfaces: Physics, propagation, and pathloss modeling," *IEEE Wireless Commun. Lett.*, vol. 9, no. 5, pp. 581–585, 2019.

[18] C. Huang, A. Zappone, G. Alexandropoulos, M. Debbah, and C. Yuen, "Reconfigurable intelligent surfaces for energy efficiency in wireless communication," *IEEE Trans. Wireless Commun.*, vol. 18, no. 8, pp. 4157–4170, 2019.

[19] X. Tan, Z. Sun, D. Koutsonikolas, and J. M. Jornet, "Enabling indoor mobile millimeter-wave networks based on smart reflect-arrays," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 270–278.

[20] M. Di Renzo, A. Zappone, M. Debbah, M.-S. Alouini, C. Yuen, J. De Rosny, and S. Tretyakov, "Smart radio environments empowered by reconfigurable intelligent surfaces: How it works, state of research, and the road ahead," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 11, pp. 2450–2525, 2020.

[21] X. Yuan, Y.-J. A. Zhang, Y. Shi, W. Yan, and H. Liu, "Reconfigurable-intelligent-surface empowered wireless communications: Challenges and opportunities," *IEEE wireless communications*, vol. 28, no. 2, pp. 136–143, 2021.

[22] C. Liaskos, S. Nie, A. Tsioliaridou, A. Pitsillides, S. Ioannidis, and I. Akyildiz, "End-to-end wireless path deployment with intelligent surfaces using interpretable neural networks," *IEEE Transactions on Communications*, 2020.

[23] C. Liaskos, A. Tsioliaridou, S. Nie, A. Pitsillides, S. Ioannidis, and I. Akyildiz, "An interpretable neural network for configuring programmable wireless environments," in *IEEE Int. Workshop Signal Processing Adv. Wireless Commun. (SPAWC 2019)*, 2019, pp. 1–5.

[24] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat, "Enforcing performance isolation across virtual machines in xen," in *Proc. of ACM/IFIP/USENIX'06*, pp. 342–362.

[25] A. Gulati *et al.*, "Vmware distributed resource management," *VMware Technical Journal*, vol. 1, no. 1, pp. 45–64, 2012.

[26] R. McDougall and J. Anderson, "Virtualization performance: Perspectives and challenges ahead," *ACM SIGOPS Review*, vol. 44, no. 4, pp. 40–56, 2010.

[27] J. Smith *et al.*, "Stochastic-based robust dynamic resource allocation in a heterogeneous computing system," in *2009 International Conference on Parallel Processing*. IEEE, 2009, pp. 188–195.

[28] D. Guo and L. N. Bhuyan, "A qos aware multicore hash scheduler for network applications," in *Proc. IEEE INFOCOM'11*, pp. 1089–1097.

[29] T. F. Abdelzaher, J. A. Stankovic, C. Lu, R. Zhang, and Y. Lu, "Feedback performance control in software services," *IEEE Control Systems Magazine*, vol. 23, no. 3, pp. 74–90, 2003.

[30] J. Wei, X. Zhou, and C.-Z. Xu, "Robust processing rate allocation for proportional slowdown differentiation on internet servers," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 964–977, 2005.

[31] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements," in *International Workshop on Quality of Service*. Springer, 2003, pp. 381–398.

[32] K. Katsalis, L. Tassiulas, and Y. Viniotis, "Distributed resource allocation mechanism for soa service level agreements," in *Procedings of IEEE IFIP International Conference on New Technologies, Mobility and Security 2011*, pp. 1–6.

[33] K. J. Åström, "Theory and applications of adaptive control-a survey," *Automatica*, vol. 19, no. 5, pp. 471–486, 1983.

[34] C.-p. Li and M. J. Neely, "Delay and rate-optimal control in a multi-class priority queue with adjustable service rates," in *Proceedings IEEE INFOCOM'12*.

[35] P. P. Bhattacharya, L. Georgiadis, P. Tsoucas, and I. Viniotis, "Adaptive lexicographic optimization in multi-class m/gi/1 queues," *Mathematics of Operations Research*, vol. 18, no. 3, pp. 705–740, 1993.

[36] C.-p. Li, G. S. Paschos, L. Tassiulas, and E. Modiano, "Dynamic overload balancing in server farms," in *Proc. of 2014 IFIP Networking'14*, pp. 1–9.

[37] T. Abdelzaher and K. G. Shin, "End-host architecture for qos-adaptive communication," in *Proc. of IEEE RTSS'98*, pp. 121–130.

[38] T. F. Abdelzaher and K. G. Shin, "Qos provisioning with q-contracts in web and multimedia servers," in *Proc. IEEE RTSS'99*, pp. 44–53.

[39] K. Katsalis, G. S. Paschos, L. Tassiulas, and Y. Viniotis, "Dynamic cpu scheduling for qos provisioning," in *Proceedings of IEEE IM'13*, pp. 630–635.

[40] K. Katsalis, G. S. Paschos, Y. Viniotis, and L. Tassiulas, "Cpu provisioning algorithms for service differentiation in cloud-based environments," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 61–74, 2015.

[41] J. L. Hellerstein *et al.*, *Feedback Control of Computing Systems.* Wiley Online Library, 2004.

[42] J. Nagle, "On packet switches with infinite storage," *IEEE Transactions on Communications*, vol. 35, no. 4, pp. 435–438, 1987.

[43] D. G. Feitelson, *Workload Modeling for Computer Systems Performance Evaluation.* Cambridge University Press, 2015.

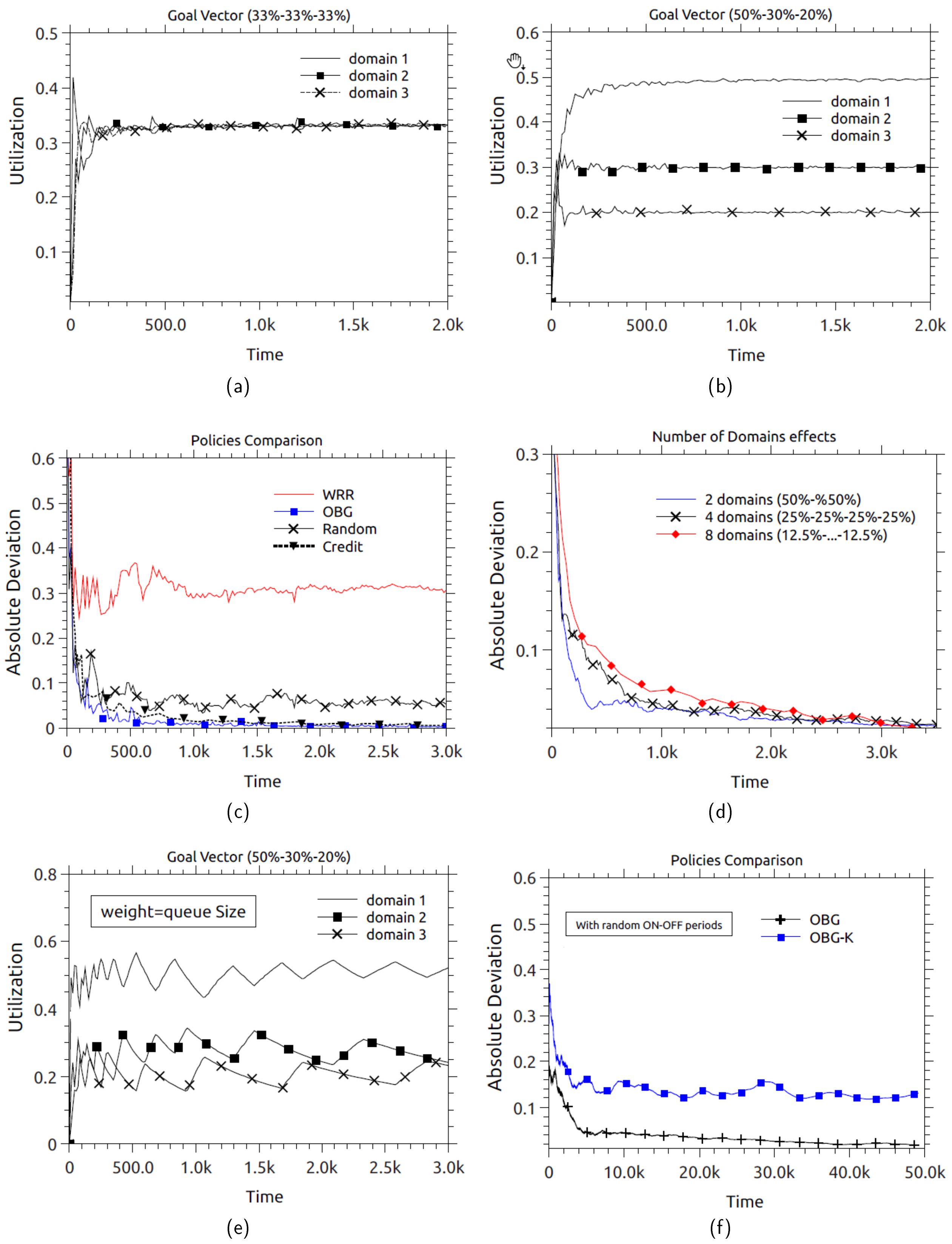
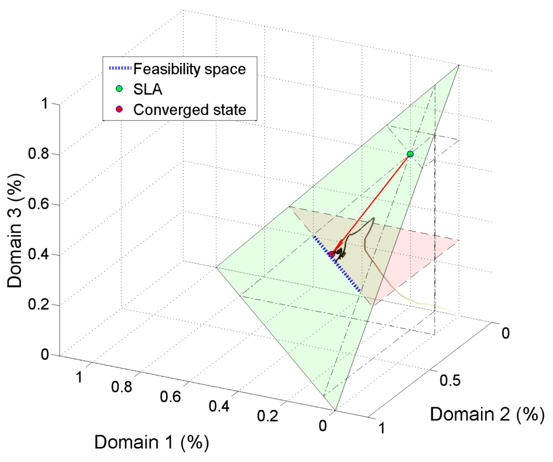
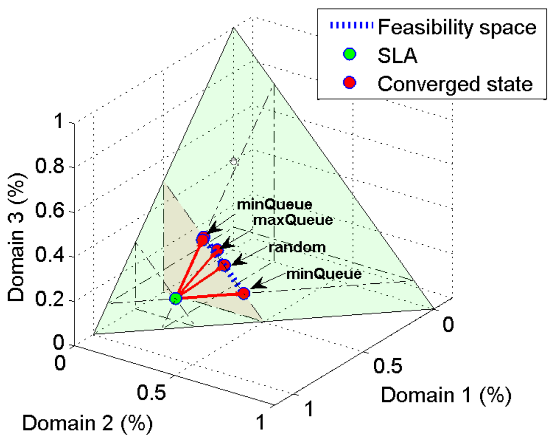


Fig. 5 – The examined aspects of DWRR Performance



(a)



(b)

Fig. 6 – DWRR Performance under various service redistribution policies.

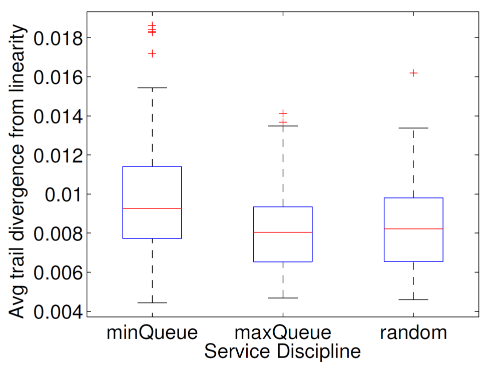


Fig. 7 – The selected **h**-algorithm affects the way the system converges to the final state.