Resource Harvesting in Cognitive Wireless Computing Networks with Mobile Clouds and Virtualized Distributed Data Centers: Performance Limits

Maria Kangas, Savo Glisic, *Senior Member, IEEE,* Yuguang Fang, *Fellow, IEEE,* and Pan Li, *Member, IEEE,*

***Abstract*—We consider a virtualized data center (VDC) consisting of a set of servers hosting a number of mobile terminals forming a mobile cloud, and study the problem of optimal resource allocation in the presence of time varying workloads and uncertain channels. The channel uncertainty may be either due to fading and/or uncertain link availability and reliability in cognitive wireless networks. The servers are processing certain applications delegated to them by the terminals, for either energy saving or due to the lack of necessary software at the terminal to process the applications. The control problem is to dynamically adjust resources according to channel and workload fluctuations in order to maximize the long-term average throughput and to minimize the energy cost of the overall system while maintaining network stability. We develop a unified VDC model for both cognitive and conventional wireless networks, carry out a unified stability analysis and characterize the joint stability region for the unified VDC model. We also propose a new dynamic policy that supports every point in the network stability region, outperforms previously proposed network stabilizing policies without using the information of arrival statistics and mitigates the mutual impact of primary and secondary service providers on each other.**

***Index Terms*—Lyapunov drift, network stability, dynamic programming, stability analysis, value iteration algorithm.**

I. INTRODUCTION

Due to the high cost of cloud service data centers, there is a growing interest in improving the energy efficiency of today's data centers and cloud computing facilities [1]. Unfortunately, resources inside the data centers often operate at low utilization due to the inefficient resource allocation [2]. For example, a single idle server can draw as much as 65% of the peak power value if not turned off [3]. In current systems, servers are also under-utilized most of the time, as applications' resource demands are easily over-estimated in order to handle even the most demanding workloads. As a result, applications hold resources that they hardly need at all, since large workloads may be rare. Ideally, unused resources should be released for other applications to use.

Data center virtualization has been shown to offer great benefits in reducing the total power consumption and increasing reliability allowing multiple heterogeneous applications to share resources and run simultaneously on a single server [4]. Virtualization increases server utilization by enabling consolidation of multiple applications on the same server and the sharing of resources among these applications. By using this technique, it is possible to control the data center so that the virtual machines (VMs) occupy only the necessary resources to serve their applications. However, achieving right balance between consolidation and resource utilization of each application is a critical issue for applications with time-varying demands. Workload adaptive resource allocation is important to create high performance data centers. In addition, in order to handle multiple resource competing applications with time varying demands, implementing efficient power allocation, scheduling and routing algorithms is important.

In this paper, we consider a virtualized cloud service data center in the presence of workload fluctuations and uncertain channels. The cloud consists of a set of terminals with queues and the data center is composed of a subset of more powerful servers, which are distributed across the network. The channel uncertainty is due to fading in conventional wireless networks (CWNs) and/or uncertain link availability and reliability both in primary service provider (PSP) and secondary service provider (SSP) cognitive networks (CNs). The statistics of these uncertainties in a SSP cognitive network are studied in [5]. In order to increase the energy efficiency of cognitive networks, the concepts of SSP and PSP cognitive networks have been recently introduced in [6]. In this concept, SSP provides channel state information for secondary users (SUs) so that the complexity is allocated to the network rather than to the terminals. In this way, a wide range of terminals can operate as SUs and terminals do not need to have cognitive capabilities. The goal of this work is to maximize a joint utility of the long-term application processing throughput of the terminals and to minimize the average total power usage in the overall system while keeping the network stable. We believe that our results can be used as a performance benchmark for comparing various solutions of different practical resource allocation schemes in the VDCs.

The remainder of this paper is organized as follows. The

M. Kangas and S. Glisic are with the Department of Electrical and Computer Engineering, University of Oulu, Oulu, Finland, 90570.
E-mails: maria.kangas@ee.oulu.fi, savo.glisic@ee.oulu.fi
Y. Fang is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA.
E-mail: fang@ece.ufl.edu
P. Li is with the Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH 44106, USA.
E-mail: lipan@case.edu

The work of Y. Fang and P. Li was partially supported by US National Science Foundation under grant CNS-1343356/CNS-1343220 and CNS-1147813/CNS-1147851. The work of M. Kangas and S. Glisic was partially supported by the Finnish Academy project COCAHANE # 257162.

related work is presented in Section II. Section III describes the system model and Section IV presents the optimization problem formulation. In Section V, we reformulate the problem as a Markov decision process (MDP), and propose a new optimal dynamic control policy. The joint network stability regions for both SSP and PSP cognitive networks and also for CWNs are derived in Section VI. The complexity of the dynamic policy is analyzed in Section VII. In Section VIII, the unified stability analysis for both PSP and SSP cognitive networks and CWN is introduced and our dynamic transmission policy is shown to outperform other existing network stabilizing policies. The simulations are conducted to validate the theoretical analysis of this paper and presented in Section IX.

II. RELATED WORK

Dynamic resource allocation in VDCs has been a hot topic among the researchers [2], [7], [8], [9]. In [2], [7] and [8] feedback-driven resource control systems are designed to automatically adapt to dynamic workload changes and to meet service level objectives of applications within the shared virtualized infrastructure. Such techniques use feedback control loop, where the goal is to allocate resources to meet its performance target. However, since feedback techniques require information about the target performance level, they cannot be used when the goal is to maximize the utility. In [9], the authors propose a dynamic live placement scheme for applications in cloud computing environments called EnaCloud, where an energy-aware heuristic algorithm is proposed to minimize the number of running VMs. Much of the previous work on resource allocation in the VDCs is based on proactive workload adaptive resource provisioning and steady state queuing models [10], [11], [12], [13]. The work in [10] defines a dynamic resource provisioning problem for virtualized server systems as a sequential optimization problem which is solved using a lookahead control [11]. Such a technique is quite useful when control actions have deadlines to meet, but requires estimates of future workloads. In [13], dynamic resource provisioning in a virtualized service environment is based on the estimate of the power usage behavior of the hosted applications. Three online workload adaptive resource control mechanisms based on steady state queueing analysis, feedback control theory and the combination of these two are proposed in [12]. This approach requires the implementation of the statistical models for the workload, and resource allocation decisions are then made to meet such a predicted resource demand. When predictions are accurate, proactive resource allocation does provide very good performance [14]. In practice, however, predictions may be inaccurate and expensive since they require workload data analysis and storage space. Lyapunov optimization has been used to guarantee network stability optimal cross-layer control policies for wireless networks [15]. The work in [16] uses Lyapunov optimization to design an online control, routing and resource allocation algorithm for a VDC. While this algorithm adjusts to workload fluctuations, it does not take into account the possible channel variations between the terminals and the servers. By considering the changing user demands, control decisions based on both the channel variations and the workload, have been shown to be effective in providing higher throughput and smaller delay in the presence of time varying channels and resource demands [17], [18].

In this paper, we maximize the long-term application processing throughput of the terminals and minimize the average total power usage in the overall system while guaranteeing the network stability. Different from the previous work, our control problem is formulated as a Markov decision process (MDP) and solved using dynamic programming and value iteration algorithm (VIA) [19], [20] for both PSP and SSP cognitive networks as well as for CWNs. The resulting dynamic control policy is shown to support every point on the network stability region without requiring the information of arrival statistics and proved to be stable using the Lyapunov drift theory. In [17] and [21], a randomized stationary policy and a frame based algorithm were used to analyse the stability of a dynamic algorithm. It is shown in [17], [21] that the performance of their dynamic algorithm is fixed amount worse than the performance of the randomized stationary and the frame based algorithms. In this paper, we prove that the performance of our dynamic policy is better than the performance of the stationary policy and propose a new unified stability analysis for both PSP and SSP cognitive networks as well as for CWNs. In addition, we show that the frame based policy proposed in [17], [21] cannot guarantee network stability. Different from the works that use steady state queuing and channel models, our approach makes use of both the queue length state information (QSI) and the channel state information (CSI) to dynamically adjust the available resources to meet the demand and to increase reliability and resource utilization of the data center. Our approach also differs from the previous works in the sense that the requests can be processed either at the terminals or at the virtual machines of the servers depending on CSI, QSI and computational intensity of the request. Resource harvesting in this paper refers to the possibility of opportunistic utilization of the network resources by a terminal. These resources include:

- Spectrum, when using cognitive links.
- Power of the data center.
- Necessary software in the data center that is not available at the terminals.

The contributions of this paper can be summarized as follows: 1. A comprehensive unified model of the virtualized data center (computing could) for both PSP and SSP cognitive networks as well as for CWNs is developed. 2. The model decouples performance analysis of PSP and SSP cognitive networks although their operations are interdependent. 3. The mutual impact of PSP and SSP cognitive networks is mitigated by appropriate adaptation of the access control parameters in the network. 4. New optimal dynamic control policy is introduced. 5. Unified stability region for PSP and SSP cognitive networks and CWNs are characterized. 6. Unified stability analysis for both PSP and SSP cognitive networks as well as for CWNs is presented. 7. Using the Lyapunov drift theory, it is shown that our dynamic policy supports every point in the network stability region without requiring information of

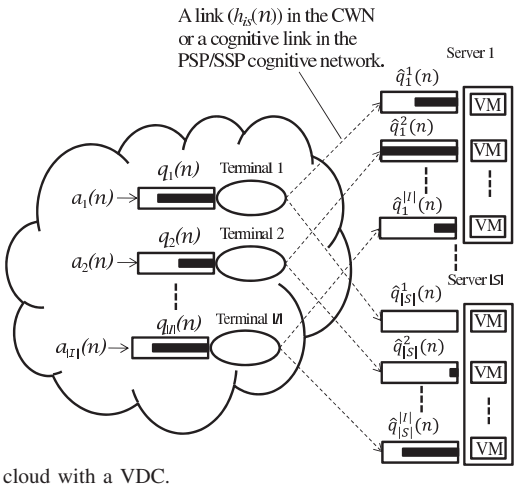


Fig. 1. A cloud with a VDC.

arrival statistics and that the performance of our policy is better than the performance of the stationary randomized policy propose in [17], [21]. 8. We show that the frame based policy described in [17], [21] cannot guarantee system stability.

III. SYSTEM MODEL

We consider a network composed of a VDC and a number of mobile terminals with queues belonging to different clusters of mobile clouds. We use $\mathcal{I}$ to denote the set of terminals within a cloud and the VDC is composed of a set of servers $\mathcal{S}$ hosting the cloud, as illustrated in Fig. 1. The VDC may be either centralized or distributed across the network as in the network with caching [22]. However, on purpose, we do not want to limit our work on a specific network architecture. Our analysis is valid for any data center with partitioning (virtualization) of the processing resources (centralized or distributed) and any conventional or PSP/SSP cognitive network characterized by the primary user (PU) return probability and secondary user (SU) channel sampling quality. By definition, mobile cloud is a set/cluster of terminals that share a certain pool of resources [23]. In our case, the terminals share the resources located at the data center.

Let $|\mathcal{S}|$ denote the number of servers within the data center and $|\mathcal{I}|$ represent the number of terminals within the cloud. Each server s is transformed into $|\mathcal{I}|$ VMs, each capable of serving a terminal. For simplicity, we assume that each mobile terminal can request service only from one server at a time, but the hosting server can change in time. By dividing the time into frames with index n, we define the following parameter for each terminal i and server s:

$$b_{is}(n) = \begin{cases} 1; & \text{If terminal } i \text{ is served on a VM of server } s \\ & \text{in frame } n. \\ 0; & \text{Otherwise.} \end{cases}$$

Let $B_i(n) = [b_{i1}(n), ..., b_{i|S|}(n)]$ denote the vector of these parameters in frame n.

Application requests arrive to each terminal i according to a process $a_i(n)$ at the beginning of each frame n. The arrival processes $a_i(n)$ are stationary and ergodic with average rates λ_i requests/frame. The external arrivals $a_i(n)$ are bounded in their second moments every frame and $\mathbb{E}\{[a_i(n)]^2\} \leq (a_i^{\max})^2$ for all $i \in \mathcal{I}$. However, we do not assume any knowledge of the statistics of $a_i(n)$. We let $A(n) = [a_1(n), ..., a_{|\mathcal{I}|}(n)]$ denote the vector of these arrivals. For analysis purpose, we assume that the application requests are placed into infinite length transmission buffers $q_i(n)$, that are later defined in Subsection III-C.

A. Channel Model

We use $|h_{is}(n)|^2$ to represent the channel gain between terminal i and server s. A block fading model is assumed so that the channel values remain fixed during a frame and may change from frame to frame according to a Markov chain. Let $H_i(n) = [|h_{i1}(n)|^2, |h_{i2}(n)|^2, ..., |h_{i|\mathcal{S}|}(n)|^2] \in \mathcal{H}_i$ denote the vector of channel gain processes at terminal i in frame n. The channel process $H_i(n)$ is stationary and ergodic and takes values on a finite state space $\mathcal{H}_i$. Since the servers can have different locations, it is possible that the channels between terminal i and different servers are different.

If the channel is used within the CWN, the channel gain vector is given by $H(n)$ in every frame n. Let π_{H_i} represent the steady state probability for the channel state H_i in the CWN. The channel processes are channel convergent with steady state probabilities π_{H_i}.

If the channel is used within the cognitive network, the equivalent channel gain process $H_i^e(n)$ will have the following form:

$$H_i^e(n) = \begin{cases} H_i(n); & \text{With probability } p_H^{\mathrm{P}} \text{ for PSP CN or} \\ & \text{with probability } p_H^{\mathrm{S}} \text{ for SSP CN.} \\ 0; & \text{With probability } p_0^{\mathrm{P}} \text{ for PSP CN or} \\ & \text{with probability } p_0^{\mathrm{S}} \text{ for SSP CN.} \end{cases}$$

For the PSP cognitive network, $p_H^{\mathrm{P}} = (1 - p_1^{\mathrm{S}}) + p_1^{\mathrm{S}} p_{\mathrm{pd}}$ and $p_0^{\mathrm{P}} = p_1^{\mathrm{S}}(1 - p_{\mathrm{pd}})$. We assume that PU transmits a preamble prior to message transmission to clear the channel in case that SU is using it (with probability p_1^{S}). Secondary user detects correctly that preamble and clears the channel with probability p_{pd}. Let p_1^{P} represent the probability that a PU is active and p_{id} is the probability that a SU detects the idling channel. The derivation of the probability 1 - p_1^{P} is given in [5]. In the SSP cognitive network, p_H^{S} is then given as $p_H^{\mathrm{S}} = (1 - p_1^{\mathrm{P}}) p_{\mathrm{id}}$, and the probability that the channel cannot be used is $p_0^{\mathrm{S}} = (1 - p_1^{\mathrm{P}})(1 - p_{\mathrm{id}}) + p_1^{\mathrm{P}}$. In other words, SU gets the channel $H_i(n)$, if the PU is not active and the SU detects the idling channel. The channel is not used, if PU is not active but the SU fails to detect the idling channel or the PU is active. Let $\pi_{H_i}^e$ denote the steady state probability for channel state H_i^e in PSP/SSP cognitive networks given as

$$\pi_{H_i}^e = \begin{cases} p_H^{\mathrm{P}} \pi_{H_i} / p_H^{\mathrm{S}} \pi_{H_i}; & \text{When } H_i^e = H_i. \\ 1 - p_H^{\mathrm{P}} / 1 - p_H^{\mathrm{S}}; & \text{When } H_i^e = 0. \end{cases}$$

We use $I(n)$ to denote the channel availability indicator at the beginning of a frame n. For the SSP cognitive network, $I(n)$ is defined as

$$I(n) = \begin{cases} 1; & \text{If } H_i^e(n) = H_i(n). \\ 0; & \text{If } H_i^e(n) = 0. \end{cases}$$

The probability that $I(n) = 1$ is $p[I(n) = 1] = p_H^{\mathrm{S}}$ and the probability that $I(n) = 0$ is $p[I(n) = 0] = p_0^{\mathrm{S}}$. For the PSP cognitive network, $I(n)$ is given as

$$I(n) = \begin{cases} 1; & \text{If } H_i^e(n) = H_i(n). \\ 0; & \text{If } H_i^e(n) = 0. \end{cases}$$

The probabilities for the PSP cognitive network are $p[I(n) = 1] = p_H^{\mathrm{P}}$ and $p[I(n) = 0] = p_0^{\mathrm{P}}$.

In addition, for the given channel in the SSP cognitive network, we define a channel corruption indicator $I_{\mathrm{r}}(n)$ during a frame n. In the SSP cognitive network, $I_{\mathrm{r}}(n)$ is given as

$$I_{\mathrm{r}}(n) = \begin{cases} 0; & \text{If PU returns to the channel.} \\ 1; & \text{Otherwise.} \end{cases}$$

The probabilities $p[I_{\mathrm{r}}(n) = 1] = 1 - p_{\mathrm{return}}^{\mathrm{P}}$ and $p[I_{\mathrm{r}}(n) = 0] = p_{\mathrm{return}}^{\mathrm{P}}$. The PU return probability $p_{\mathrm{return}}^{\mathrm{P}}$ is discussed in [5]. The channel corruption indicator $I_{\mathrm{r}}(n)$ in the PSP cognitive network is

$$I_{\mathrm{r}}(n) = \begin{cases} 0; & \text{If SU returns to the channel and does not} \\ & \text{detect the presence of PU (collision).} \\ 1; & \text{Otherwise.} \end{cases}$$

For the PSP cognitive network, the probabilities are given as $p[I_{\mathrm{r}}(n) = 1] = (1 - p_{\mathrm{return}}^{\mathrm{S}}) + p_{\mathrm{return}}^{\mathrm{S}} p_{\mathrm{sd}}$ and $p[I_{\mathrm{r}}(n) = 0] = p_{\mathrm{return}}^{\mathrm{S}}(1 - p_{\mathrm{sd}})$, where $p_{\mathrm{return}}^{\mathrm{S}}$ is the probability of SU returning to the channel and p_{sd} is the probability that SU correctly detects the presence of PU.

Additional modification of the channel model includes the option what we refer to as "partial cognitive networks" (PC networks), where the network operator's overall resources include both cognitive and conventional (purchased) links [6]. Given π_{H_i}, $\pi_{H_i}^e$, H_i^e and $H_i \in \mathcal{H}_i$, deriving the channel model for the PC network is straightforward.

B. Power consumption

Depending on the current workloads, current channel states, available energy and needed software, the application requests can be processed either at the terminal or delegated to be performed at one of the servers hosting the terminal. Let $\mu_{is}(n)$ denote the number of requests delivered from terminal i to be processed at the hosting server s in frame n. We use $\mu_i(n)$ to represent the number of requests processed at terminal i in frame n, when there is a channel available between terminal i and server s, i.e., $H_i(n) \in \mathcal{H}_i$. In addition, let $\mu_{i0}(n)$ denote the number of requests that can be processed at terminal i only, when there is no channel available between terminal i and server s in frame n, i.e., $I(n) = 0$. When $I(n) = 0$, more applications might be processed at terminal i only and $\mu_{i0}(n) \geq \mu_i(n)$.

We use $P_i^{\mathrm{tot}}(n) = P_i(n) + P_{is}(n)$ to represent the total power consumption of terminal i in frame n, where $P_i(n)$ is the power required to process application requests at terminal i and $P_{is}(n)$ is the power required to deliver requests to be processed at server s. Let α_i and α_{is} denote non-negative parameters. In the CWN, we have

$$P_i(n) = \mu_i(n)\alpha_i, \tag{1}$$

$$P_{is}(n) = \frac{\mu_{is}(n)\alpha_{is}}{|h_{is}(n)|^2}. \tag{2}$$

In the PSP/SSP congnitive networks, $P_i(n)$ and $P_{is}(n)$ are given as

$$P_i(n) = I(n)\mu_i(n)\alpha_i + [1 - I(n)]\mu_{i0}(n)\alpha_i, \tag{3}$$

$$P_{is}(n) = \frac{I(n)\mu_{is}(n)\alpha_{is}}{|h_{is}(n)|^2}. \tag{4}$$

Let $P^{\max}$ denote the maximum power available at terminal i in frame n.

Each server s has a set of resources that are allocated to the VMs hosted on it by its resource controller. These resources can include, for example, the data center power and the necessary software at the data center that is not available at the terminals. Both of these resources can be easily added into the system model as described later in Section IV. However, in this paper, we only focus on the CPU frequency and power constraints. All servers are assumed to have identical CPU resources. In our model, CPUs run at finite number of operating frequencies $f_{\min} < f < ... < f_{\max}$. At each utilization level f, the power consumption at server s is estimated as $\hat{P}_s(f) = \hat{P}_{\min} + \theta(f - f_{\min})^2$ [16]. Available techniques such as dynamic frequency scaling (DFS), dynamic voltage scaling (DVS) and combination of the two can be used to change the current CPU frequency that affects the CPU power consumption [24], [25]. The maximum power at server s is given as $\hat{P}^{\max} = \hat{P}_{\min} + \theta(f_{\max} - f_{\min})^2$. At utilization level f, the maximum supportable service rate $\hat{\mu}_s(f)$ at server s is given as [16]

$$\hat{\mu}_s(f) = \frac{\hat{P}_s(f)}{\hat{\alpha}_s} = \frac{\hat{P}_{\min} + \theta(f - f_{\min})^2}{\hat{\alpha}_s}, \tag{5}$$

where $\hat{\alpha}_s$ represents a non-negative parameter. The VM's resource allocation can be changed dynamically online without disrupting the running applications within the VMs [26]. The resources for each VM are adapted to the changing workloads during its lifetime. In virtualized server environment the virtual machine monitor (VMM) at any physical machine handles resource multiplexing and isolation between VMs [26].

C. Queueing Model

Every frame n in the CWN, $\mu_i(n) + \mu_{is}(n)$ application requests are removed from the buffer of terminal i. Let $q_i(n)$ denote the queue length at terminal i and $Q(n) = [q_1(n), q_2(n), ..., q_{|I|}(n)]$ represent the vector of queue lengths at terminals in frame n. The queuing dynamics in the CWN are then given as

$$q_i(n+1) = q_i(n) + a_i(n) - [\mu_i(n) + \mu_{is}(n)]. \tag{6}$$

For the PSP/SSP cognitive networks, the corresponding equation is

$$q_i(n+1) = q_i(n) + a_i(n) - I(n)[\mu_i(n) + I_{\mathrm{r}}(n)\mu_{is}(n)] + [1 - I(n)]\mu_{i0}(n). \tag{7}$$

At each server s, the delegated requests can be stored into a buffer reserved for terminal i at server s before the

requests are processed at server s. We use $\hat{q}_s^i(n)$ to denote the queue length of terminal i at server s, $\widehat{\mathbf{Q}}(n) = [\hat{q}_1^1(n), \hat{q}_2^1(n), ..., \hat{q}_{|\mathcal{S}|}^1(n); ...; \hat{q}_1^{|\mathcal{I}|}(n), \hat{q}_2^{|\mathcal{I}|}(n), ...\hat{q}_{|\mathcal{S}|}^{|\mathcal{I}|}(n)]$ denotes the $|\mathcal{I}| \times |\mathcal{S}|$ matrix of the queue lengths at each server s and $\widehat{Q}_i(n) = [\hat{q}_1^i(n), \hat{q}_2^i(n), ..., \hat{q}_{|\mathcal{S}|}^i(n)]$ represents the ith row of $\widehat{\mathbf{Q}}(n)$. Let $\hat{\mu}_s^i(n)$ represent the service rate [requests/frame] server s provides to terminal i in frame n. The queueing dynamics for the application requests of terminal i at server s for both PSP and SSP cognitive networks is given as

$$\hat{q}_s^i(n+1) = \hat{q}_s^i(n) + I(n)I_{\mathrm{r}}(n)\mu_{is}(n) - \hat{\mu}_s^i(n). \tag{8}$$

For the CWN, $\hat{q}_s^i(n+1)$ is written as

$$\hat{q}_s^i(n+1) = \hat{q}_s^i(n) + \mu_{is}(n) - \hat{\mu}_s^i(n). \tag{9}$$

Finally, let $\hat{\mu}_s(n) = \sum_{i\in\mathcal{I}} \hat{\mu}_s^i(n)$ represent the total service rate at server s, and $\hat{q}_s(n) = \sum_{i\in\mathcal{I}} \hat{q}_s^i(n)$ denote the sum of queue lengths at server s.

IV. UNIFIED PROBLEM FORMULATION

In order to derive a unified optimization problem for both CWN and PSP/SSP cognitive wireless networks, one should note that the service rates for the PSP/SSP cognitive networks can be derived from the service rates of the CWN. When the number of requests transmitted from terminal i to server s and the number of requests processes at terminal i in the CWN are given by $\mu_{is}(n)$ and $\mu_i(n)$, respectively, the corresponding service rates for PSP and PSP cognitive networks are defined as

$$\mu_{is}(n)^* = \mu_{is}(n)p[I(n)=1]p[I_{\mathrm{r}}(n)=1] \tag{10}$$

$$\acute{\mu}_i(n)^* = \mu_i(n)p[I(n)=1] + \mu_{i0}(n)p[I(n)=0] = \mu_i(n)^* + \mu_{i0}(n)^*, \tag{11}$$

where $\mu_i(n)^* = \mu_i(n)p[I(n)=1]$ and $\mu_{i0}(n)^* = \mu_{i0}(n)p[I(n)=0]$.

Given (10) and (11), the unified power consumption and queueing dynamics for both PSP and SSP cognitive networks as well as for CWN are

$$P_i(n) = \alpha_i \acute{\mu}_i(n)^*, \tag{12}$$

$$P_{is}(n) = \frac{\mu_{is}(n)p[I(n)=1]\alpha_{is}}{|h_{is}(n)|^2}, \tag{13}$$

$$q_i(n+1) = q_i(n) + a_i(n) - [\acute{\mu}_i(n)^* + \mu_{is}(n)^*] \tag{14}$$

for each terminal i and

$$\hat{q}_s^i(n+1) = \hat{q}_s^i(n) + \mu_{is}(n)^* - \hat{\mu}_s^i(n) \tag{15}$$

for each terminal i at server s.

A specific control action at terminal i is a decision on how many applications are processed at the terminal, how many requests are forwarded to server s, and which specific server s is hosting the terminal i. We let $\mathcal{U}(n)$ denote the set of control actions available at the terminals in frame n, and $U_i(n) = \{\acute{\mu}_i(n)^*, \mu_{is}(n)^*, b_{is}(n)\} \in \mathcal{U}(n)$ represents a specific control action at terminal i in frame n. In addition, we use $U(n) = [U_1(n), U_2(n), ..., U_{|\mathcal{I}|}(n)]$ to represent the vector of control actions in frame n.

The control action at each server s includes selecting the CPU frequency, that affects the power consumption $\hat{P}_s(n)$, as well as CPU resource distribution among different VMs that host the terminals running on that server. This allocation is subject to the available control options at each server s. For example, the controller may allocate different fractions of CPU to the VMs in that frame. We use $\hat{\mathcal{U}}(n)$ to denote the set of all control actions available at server s. Let $\hat{U}_s(n) = \{\hat{\mu}_s(n)\} \in \hat{\mathcal{U}}(n)$ denote a particular control action taken at server s in frame n under any policy and $\hat{P}_s(n)$ is the corresponding power consumption. The vector of control actions at the data center is given as $\hat{U}(n) = [\hat{U}_1(n), \hat{U}_2(n), ..., \hat{U}_{|\mathcal{S}|}(n)]$.

Let $X(n) = \{Q(n) + A(n), \widehat{\mathbf{Q}}(n), \mathbf{H}(n)\}$ represent the state of the system in frame n with countable state space $\mathcal{X}$, where $\mathbf{H}(n) = [|h_{11}(n)|^2, |h_{12}(n)|^2, ..., |h_{1|\mathcal{S}|}(n)|^2; \ |h_{21}(n)|^2, |h_{22}(n)|^2, ..., |h_{2|\mathcal{S}|}(n)|^2; ...; |h_{|\mathcal{I}|1}(n)|^2, |h_{|\mathcal{I}|2}(n)|^2, ..., |h_{|\mathcal{I}||\mathcal{S}|}(n)|^2]$ denote $|\mathcal{I}| \times |\mathcal{S}|$ channel gain matrix in frame n. We use $D_X(n) = \{U(n), \hat{U}(n)\}$ to denote the control input, i.e., the action, in fame n, when the state of the system is $X(n)$. At the beginning of each fame n, the network controller decides upon the value of $D_X(n)$ depending on the current state of the system $X(n)$. The control input $D_X(n)$ takes values in a general state space $\mathcal{D}_X(n)$, which represents all the feasible control options in state $X(n)$. Starting from state X, let $\pi = \{D_X(1), D_X(2), ...\}$ denote the policy, i.e., the sequence of actions. We use Π to denote the space of all such policies and $\pi \in \Pi$.

It is important to note that the availability of the software resources could be added here to the system model by simply introducing a binary variable

$$\varphi_i(n) = \begin{cases} 1; & \text{If terminal } i \text{ has the necessary} \\ & \text{software to process the applications.} \\ 0; & \text{Otherwise.} \end{cases}$$

and rewriting the state as $X(n) = \{Q(n) + A(n), \widehat{\mathbf{Q}}(n), \mathbf{H}(n), \varphi(n)\}$, where $\varphi(n) = [\varphi_1(n), ..., \varphi_{|\mathcal{I}|}(n)]$ is the vector of variables $\varphi_i(n)$. If $\varphi_i(n) = 0$, application requests cannot be processed at terminal i in frame n.

Let δ_i represent a non-negative weight used as a normalizing parameter. The goal is to map from the current $X(n)$ to an optimal sequence of $D_X(n)$, that solves the following optimization problem:

$$\underset{\pi\in\Pi}{\text{maximize}} \quad \lim_{n\to\infty} \frac{1}{n} \sum_{\eta=0}^{n-1} \sum_{i\in\mathcal{I}} \mathbb{E}_X^{\pi} \Big\{ \acute{\mu}_i(\eta)^* + \sum_{s\in\mathcal{S}} b_{is}(\eta)\mu_{is}(\eta)^* - \delta_i \frac{P_i^{\mathrm{tot}}(\eta)}{P^{\max}} \Big\} - \lim_{n\to\infty} \frac{1}{n} \sum_{\eta=0}^{n-1} \sum_{s\in\mathcal{S}} \mathbb{E}_X^{\pi} \{\hat{P}_s(\eta)\} \tag{16}$$

subject to

$$\lambda_i \in \Lambda_{\mathrm{T}},$$

$$q_i(\eta) \quad \text{and} \quad \hat{q}_s^i(\eta) \quad \text{stay stable},$$

$$P_i^{\mathrm{tot}}(\eta) \le P^{\max} \quad \text{and} \quad \hat{P}_s(\eta) \le \hat{P}^{\max}.$$

The constraints are valid for all $i \in \mathcal{I}$ and $s \in \mathcal{S}$ and Λ_{T} represents network stability region presented later in Section

VI. The objective in (16) is a constrained dynamic optimization problem and it maximizes the joint utility of the sum throughput of the applications processed at the terminals and minimizes the overall power usage both at the terminals and at the data center. It allows the design of resource allocation policies that adjust to workload and channel variations. For example, if the current workload is small, then this objective encourages scaling down the instantaneous capacity in the servers in order to achieve energy savings. Similarly if the current workload is large, the objective encourages scaling up the instantaneous capacity by higher power consumption. In addition, (16) encourages to delay some parts of input traffic by scheduling more packets in good channel states, and less in poor conditions in order to achieve the maximum long-term throughput with minimum power consumption.

V. OPTIMAL CONTROL POLICY

In this section, we propose a dynamic control policy that solves the constrained dynamic optimization problem in (16). Every frame n, the policy uses the current QSI and CSI to define resource allocation decisions $U_i(n)$ and $\hat{U}_s(n)$ for each terminal i and server s. However, in order to calculate the control actions at terminal i, we do not need information about the input statistics or QSI and CSI of other terminals. Similarly, in order to calculate the control actions at server s, we do not need information about the input statistics or QSI of other servers. As calculating the optimal control actions requires information of the current QSI and CSI only and do not rely on the statistics governing future arrivals, one should note that (16) can be solved separately for each terminal i and server s.

A. Resource allocation at the terminals

Let $X_i(n) = \{q_i(n) + a_i(n),\ \widehat{Q}_i(n),\ H_i(n)\}$ represent the state of terminal i in frame n with countable state space $\mathcal{X}_i$. Let $y_i(n) = q_i(n) + a_i(n)$ and rewrite $X_i(n)$ as $X_i(n) = \{y_i(n),\ \widehat{Q}_i(n),\ H_i(n)\}$. In addition, we use $U_{X_i}(n) = \{\acute{\mu}_i(n)^*, \mu_{is}(n)^*, B_i(n)\}$ to denote the control input, i.e., action, at terminal i in frame n in state $X_i(n)$. The control input $U_{X_i}(n)$ takes values in a general state space $\mathcal{U}_{X_i}(n)$, which represents all the feasible resource allocation options available in state $X_i(n)$ in frame n. By feasible options we mean the set of control actions that satisfy the power and the queue constraints, as we cannot transmit more application requests than there are in the queue. Let $\pi_i = \{U_{X_i}(0), U_{X_i}(1), ...\}$ denote the policy, i.e., the sequence of actions, at terminal i, and Π_i represent the space of all such policies.

For each terminal i, the goal of this paper is to map from the current QSI and CSI to an optimal policy $\pi_i^\star \in \Pi_i$ that stabilizes the system and solves the following optimization problem:

$$\begin{aligned} \underset{\pi_i \in \Pi_i}{\text{maximize}} \quad & \lim_{n\to\infty} \frac{1}{n} \sum_{\eta=0}^{n-1} \mathbb{E}_{X_i}^{\pi_i} \{T_i(\eta) + S_i(\eta)\} \\ \text{subject to} \quad & \lim_{n\to\infty} \frac{1}{n} \sum_{\eta=0}^{n-1} \mathbb{E}_{X_i}^{\pi_i} \left\{ \frac{P_i^{\text{tot}}(\eta)}{P^{\max}} \right\} \leq 1. \end{aligned} \tag{17}$$

In (17),

$$T_i(\eta) = [y_i(\eta) - \sum_{s\in\mathcal{S}} b_{is}(\eta)\hat{q}_s^i(\eta)] \frac{\sum_{s\in\mathcal{S}} b_{is}(\eta)\mu_{is}(\eta)^*}{\mu_{is}^{\max}}, \tag{18}$$

$$S_i(\eta) = y_i(\eta)[\acute{\mu}_i(\eta)^* + \sum_{s\in\mathcal{S}} b_{is}(\eta)\mu_{is}(\eta)^*] \tag{19}$$

and the maximum number of application requests that can be delivered from terminal i to server s in one frame is

$$\mu_{is}^{\max} = \max_{\{s\in\mathcal{S}, H_i\in\mathcal{H}_i\}} \frac{P^{\max}|h_{is}|^2}{\alpha_{is}}. \tag{20}$$

One should note that, based on the definition of H_i^{e} for PSP/SSP cognitive networks in Section III-A, $\mu_{is}^{\max}$ gets the same value for both the PSP/SSP cognitive network and the CWN. Equation (17) maximizes the long-term average throughput of the terminals while keeping the energy cost and queues low. For example, high power computationally intensive application requests at the terminal can be delegated to the hosting server in order to achieve energy savings at the terminal. If the backlog value at the terminal i is larger than the backlog of terminal i at server s, the objective in (17) encourages the terminal to delegate its requests to be processed at the servers.

1) Formulation as a Markov Decision Process: We first convert the constrained dynamic optimization problem in (17) into an unconstrained problem (UP) and then find the optimal policy for this UP [19], [20], [27], [28].

The set of feasible actions U_{X_i} in each state $X_i = \{y_i, \widehat{Q}_i, H_i\}$ is the set of all $\{\acute{\mu}_i^*, \mu_{is}^*, B_i\}$ that satisfy the power and the queue constraints as we cannot transmit more packets than there are in the queue, i.e., $\acute{\mu}_i^* + \mu_{is}^* \leq y_i$ and $P_i^{\text{tot}} \leq P^{\max}$. After taking an action $U_{X_i} = \{\acute{\mu}_i^*, \mu_{is}^*, B_i\}$, the following state is given as $Z_i = \{q_i, \widehat{Y}_i, H_i\}$, where $\widehat{Y}_i = [\hat{y}_1^i, ..., \hat{y}_{|\mathcal{S}|}^i]$ and $\hat{y}_s^i = \hat{q}_s^i + b_{is}\mu_{is}^*$. Based on (6) and (9), we get this by noting that $y_i - (\acute{\mu}_i^* + \mu_{is}^*) = q_i$ and $\hat{q}_s^i + \mu_{is}^* = \hat{y}_s^i$. It is important to note that for each state $X_i = \{y_i, \widehat{Q}_i, H_i\}$ with equal $\widehat{Q}_i$ and H_i, where $q_i \in \{0, 1, ..., y_i\}$, $a_i \in \{0, 1, ..., y_i\}$ and $q_i + a_i = y_i$, the set of feasible actions and following states are the same. Thus, state $Z_i = \{q_i, \widehat{Y}_i, H_i\}$ is *equivalent* to a state $X_i = \{y_i, \widehat{Q}_i, H_i\}$, if the channels are the same and both q_i and a_i take values with the set $\{0, 1, ..., y_i\}$ so that $q_i + a_i = y_i$ and $\hat{q}_s^i$ takes values with the set $\{0, 1, ..., \hat{y}_s^i\}$ so that $\hat{y}_s^i = \hat{q}_s^i + b_{is}\mu_{is}^*$ for each server s. When $a_i = 0$ and $b_{is}\mu_{is}^* = 0$ for all $s \in \mathcal{S}$, we have $y_i = q_i$ and $\widehat{Q}_i = \widehat{Y}_i$. Then, $X_i = \{y_i, \widehat{Q}_i, H_i\} = \{q_i, \widehat{Y}_i, H_i\} = Z_i$. For example, let us consider a system with a terminal and 2 servers. In state $X_i = \{y_i, \widehat{Q}_i, H_i\}$, we let $y_i = 3$ and $\widehat{Q}_i = [\hat{q}_1^i, \hat{q}_2^i] = [1, 2]$. Then, $q_i = \{0, 1, ..., 3\}$, $a_i = \{0, 1, ..., 3\}$, $q_i + a_i = 3$ and $[\hat{y}_1^i, \hat{y}_2^i] = [1, 2]$. When $a_i = 0$ and $b_{is}\mu_{is}^* = 0$, $y_i = q_i = 3$ and $\hat{q}_s^i = \hat{y}_s^i$. Now we have $X_i = Z_i$. This property is important when calculating the optimal value functions in (28), as $W^l(X_i) = W^l(Z_i)$, if X_i is *equivalent* to Z_i. We let $p(Z_i|X_i, U_{X_i})$ to denote the transition probability from state X_i to state Z_i with action U_{X_i}.

For a policy π_i, define the reward D_i and cost functions E_i as

$$D_i = \lim_{n\to\infty} \frac{1}{n} \sum_{\eta=0}^{n-1} \mathbb{E}_{X_i}^{\pi_i} \{T_i(\eta) + S_i(\eta)]\} \tag{21}$$

and

$$E_i = \lim_{n\to\infty} \frac{1}{n} \sum_{\eta=0}^{n-1} \mathbb{E}_{X_i}^{\pi_i} \left\{ \frac{P_i^{\text{tot}}(\eta)}{P^{\max}(\eta)} \right\}. \quad (22)$$

Let Π_i^E denote the set of all admissible control policies $\pi_i \in \Pi_i$, which satisfy the constraint $E_i(\eta) \leq 1$ in every frame η. Then, (17) can be restated as a constrained optimization problem given as

$$\text{maximize } D_i; \text{ subject to } \pi_i \in \Pi_i^E. \quad (23)$$

The problem (23) can be converted into a family of unconstrained optimization problems through a Lagrangian relaxation [29]. The corresponding Lagrangian function for any policy $\pi_i \in \Pi_i$ and for every $\beta_i \geq 0$ can now be defined as

$$J_\beta^{\pi_i}(X_i) = \lim_{n\to\infty} \frac{1}{n} \sum_{\eta=0}^{n-1} \mathbb{E}_{X_i}^{\pi_i} \{T_i(\eta) + S_i(\eta) - \beta_i E_i(\eta)\}. \quad (24)$$

Given $\beta_i \geq 0$, the unconstrained optimization problem is defined as

$$\text{maximize } J_\beta^{\pi_i}(X_i) \text{ subject to } \pi_i \in \Pi_i. \quad (25)$$

An optimal policy for unconstrained problem is also optimal for the original constrained control problem when β_i is appropriately chosen [27], [29].

The problem given in (25) is a standard MDP with the maximum average reward criterion. For each initial state $X_i \in \mathcal{X}_i$, define a corresponding discounted reward MDP with value function

$$W_\alpha(X_i) = \underset{\pi_i \in \Pi_i}{\text{maximize}} \quad \sum_{n=0}^{\infty} \mathbb{E}_{X_i}^{\pi_i} \{\alpha^n R[U_{X_i}(n), X_i(n)]\} \quad (26)$$

where the discount factor $\alpha \in (0,1)$, and a reward from taking an action $U_{X_i}(\eta)$ in state $X_i(\eta)$ is defined as

$$R[U_{X_i}(n), X_i(n)] = T_i(n) + S_i(n) - \beta_i E_i(n). \quad (27)$$

$W_\alpha(X_i)$ is defined as the optimal total expected discounted utility for discount factor α [30]. One way to solve (26) is to use value iteration algorithm (VIA) [27], [30], [31].

VIA is the standard dynamic programming approach to recursively compute the discount optimal sequence $\pi_i^\star$ for (26) [27], [31]. For notational simplicity, we suppress the subscript α. The solution to (26), i.e., the optimal value functions $W^\star(X_i)$ for each initial state X_i and the corresponding discount optimal sequences $\pi_i^\star \in \Pi_i$ can be solved with the following iterative algorithm:

$$W^{l+1}(X_i) = \max_{U_{X_i} \in \mathcal{U}_{X_i}} \{R(U_{X_i}, X_i) + \alpha \sum_{Z_i \in \mathcal{Z}_i} p(Z_i|X_i, U_{X_i}) W^l(Z_i)\}. \quad (28)$$

In (28), $\mathcal{Z}_i \subset \mathcal{X}_i$ is the set of feasible states that follow state X_i by taking an action U_{X_i}, and l denotes the iteration index. For each initial state X_i, define the optimal action in each state X_i as

$$\arg \max_{U_{X_i} \in \mathcal{U}_{X_i}} \left\{ R(U_{X_i}, X_i) + \alpha \sum_{Z_i \in \mathcal{Z}_i} p(Z_i|X_i, U_{X_i}) W^\star(Z_i) \right\}. \quad (29)$$

B. Resource allocation at the servers

Let $\hat{X}_s(n) = [\hat{y}_s^1(n), ..., \hat{y}_s^{|I|}(n)]$ represent the vector of queue lengths at server s in frame n with countable state space $\hat{\mathcal{X}}_s$. Let $\hat{U}_{\hat{X}_s}(n) = \{[\hat{\mu}_s^1(n), ..., \hat{\mu}_s^{|\mathcal{I}|}(n)]\}$ denote the particular control action in state $\hat{X}_s(n)$, and $\hat{\mathcal{U}}_{\hat{X}_s}(n)$ is the set of feasible resource allocation options in each state $\hat{X}_s(n)$. In addition, we use $\hat{\pi}_s = \{\hat{U}_{\hat{X}_s}(1), \hat{U}_{\hat{X}_s}(2), ...\}$ to denote the sequence of control actions at server s and $\hat{\Pi}_s$ represents the set of all such policies.

For each terminal s, map from the current queue and channel states to an optimal sequence of actions that stabilizes the system and solves the following optimization problem:

$$\underset{\hat{\pi}_s \in \hat{\Pi}_s}{\text{maximize}} \quad \lim_{n\to\infty} \frac{1}{n} \sum_{\eta=0}^{n-1} \sum_{i \in \mathcal{I}} \mathbb{E}_{\hat{X}_s}^{\hat{\pi}_s} \{\hat{y}_s^i(\eta) \hat{\mu}_s^i(\eta)\}$$

$$\text{subject to } \widehat{P}_{\min} \leq \lim_{n\to\infty} \frac{1}{n} \sum_{\eta=0}^{n-1} \mathbb{E}_{\hat{X}_s}^{\hat{\pi}_s} \{\widehat{P}_s(\eta)\} \leq \widehat{P}_{\max}. \quad (30)$$

The objective encourages allocating bigger fractions of CPU to the VMs of the terminals with the biggest backlog values at the server. If the current backlog value of terminal i at server s is inside the instantaneous capacity region, then this objective also encourages allocating less CPU to the VMs of the terminals with low backlog values and/or run CPU at slower speeds to achieve energy savings at the server.

1) Formulation as a Markov Decision Process: The set of feasible actions in each state $\hat{X}_s = [\hat{y}_s^1, ..., \hat{y}_s^{|I|}]$ is the set of all $\{[\hat{\mu}_s^1, ..., \hat{\mu}_s^{|\mathcal{I}|}]\}$ that satisfy $\hat{\mu}_s^i \leq \hat{y}_s^i$ and $\hat{P}_s \leq \hat{P}^{\max}$. After taking an action $\hat{U}_{\hat{X}_s}$, the following state is given as $\hat{Z}_s = \{[\hat{q}_s^1, ..., \hat{q}_s^{|I|}]\}$. State $\hat{Z}_s$ that is equivalent to a state $\hat{X}_s$, where $\hat{q}_s^i \in \{0, 1, ..., \hat{y}_s^i\}$, $b_{is}\mu_{is}^* \in \{0, 1, ..., \hat{y}_s^i\}$ and $\hat{q}_s^i + b_{is}\mu_{is}^* = \hat{y}_s^i$, as described in Subsection V-A1. Let $p(\hat{Z}_s|\hat{X}_s, \hat{U}_{\hat{X}_s})$ denote the transmission probability from state $\hat{X}_s$ to state $\hat{Z}_s$ with action $\hat{U}_{\hat{X}_s}$. Just as in Subsection V-A1, (30) can now be solved by converting it into a MDP and by finding the optimal policy for this MDP using the VIA.

VI. ACHIEVABLE RATES

The network capacity/stability region is defined as the set of all arrival rates $\lambda = [\lambda_1, ..., \lambda_{|\mathcal{I}|}]$ that the network can stably support, considering all possible resource allocation policies that we can have for the system. In this Section, we characterize the fundamental throughput limitations and present the unified capacity/stability region of the system given in Fig. 1 for both SSP and PSP cognitive networks as well as for the CWN. For precise definition of stability for single queues and for queueing networks, we refer readers to [17]. As the optimization can be solved separately for each terminal i and server s, the supportable arrival rate regions can also be derived separately for the two cases.

A. Unified Arrival Rate Region at the Terminals

Let g_i denote the long-term average number of application requests that can be supported at each terminal i in the CWN. We use c_i to denote the long-term average number of

application requests processed at terminal i, c_{is} represents the long-term average number of application requests delivered from terminal i to server s and $g_i = c_i + \sum_{s\in\mathcal{S}} c_{is}$.

Given c_i and c_{is} for the CWN, the long-term average number of application requests processed at terminal i and the long-term average number of application requests delivered from terminal i to server s for the cognitive wireless networks are respectively given as

$$c_{is}^* = c_{is}p(I=1)p(I_{\rm r}=1) \tag{31}$$

$$\acute{c}_i^* = c_i p(I=1) + c_{i0}p(I=0) = c_i^* + c_{i0}^*, \tag{32}$$

where $c_i^* = c_i p(I=1)$ and $c_{i0}^* = c_{i0}p(I=0)$. Here c_{i0}^* represents the long-term average number of requests processed at terminal i, when there is no channel available between terminal i and server s, i.e., $H_i^e = 0$. Let $g_i^* = \acute{c}_i^* + \sum_{s\in\mathcal{S}} c_{is}^*$ denote the long-term average number of application requests that can be supported at terminal i in PSP/SSP cognitive networks.

Due to the time varying channel conditions between terminal i and the servers, g_i^* must be averaged over all possible channel states. Moreover, for the given channel states, g_i^* is not fixed and depends on control policy $\pi_i \in \Pi_i$ for choosing the control actions. Thus, numerical calculation of all supportable rates g_i^* is computationally very challenging.

However, based on (1) and (2), the supportable arrival rate region at the terminals can also be defined by considering only the set of policies, where each terminal transmits at full power in each frame n. Let $\mathcal{O}_{H_i} \subset \mathcal{U}_{X_i}$ represent the set of possible options to allocate the total power $P^{\max}$ at each terminal i in channel state H_i. In addition, we use $O_{H_i} \in \mathcal{O}_{H_i}$ to denote a total power allocation action at terminal i, when the system is in channel state H_i. The long-term average transmission rate of terminal i for the full power policies is given by $g_{\max_i}^*$. The set of all full power long-term average transmission rates $g_{\max_i}^*$ that a terminal can be configured to support is now given as

$$\Gamma^* = \sum_{H_i\in\mathcal{H}_i} \pi_H \text{Conv}\{\mu_i(O_{H_i},H_i)^* + \sum_{s\in\mathcal{S}} b_{is}\mu_{is}(O_{H_i},H_i)^* | O_{H_i}\in\mathcal{O}_{H_i}\} + p(I=0)\mu_{i0}^{\max}, \tag{33}$$

where

$$\mu_{i0}^{\max} = P^{\max}/\alpha_i \tag{34}$$

is the maximum number of requests that can be processed at terminal i, when there is no channel available between terminal i and server s. For the PSP and SSP cognitive networks, $p(I=0) = p_0^{\rm P}$ and $p(I=0) = p_0^{\rm S}$, respectively. In the CWN, $p(I=0) = 0$. In (33), addition and scalar multiplication of sets are used, and $\text{Conv}\{\mathcal{B}\}$ represents the convex hull of the set $\mathcal{B}$ that is defined as the set of all convex combinations $p_1v_1 + p_2v_2 + ... + p_jv_j$ of elements $v_j \in \mathcal{V}$, where p_js are probabilities summing to 1.

The throughput region Γ^* can be viewed as the set of all long-term full power average service rates $g_{\max_i}^*$ that the terminal can be configured to support. Thus, the unified supportable rate region $\Lambda_{\rm T}$ at the terminals for both the PSP and SSP cognitive networks as well as for the CWN is the set

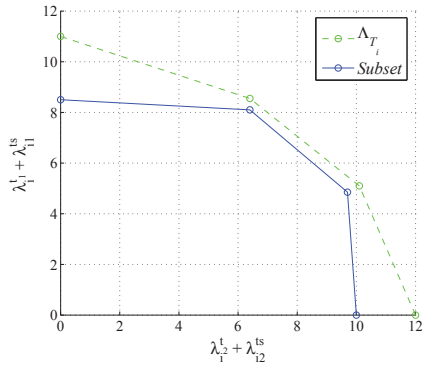


Fig. 2. The rate region $\lambda_i^{t_2} + \lambda_{i2}^{ts}$ vs. $\lambda_i^{t_1} + \lambda_{i1}^{ts}$ and the subregion λ_{i2}^{ts} vs. λ_{i1}^{ts}.

of all average arrival rates vectors $\lambda = [\lambda_1, \lambda_2, ..., \lambda_{|\mathcal{I}|}]$ for which there exists a control policy π_i that satisfies

$$\lambda_i \le \lim_{n\to\infty} \frac{1}{n}\sum_{\eta=1}^{n-1} \mathbb{E}_{X_i}^{\pi_i}\{\mu_i(\eta)^* + \sum_{s\in\mathcal{S}} b_{is}(\eta)\mu_{is}(\eta)^*\} + p(I=0)\mu_{i0}^{\max} \le g_{\max_i}^* \tag{35}$$

for some $g_{\max_i}^* \in \Gamma^*$, as rates below each point in Γ^* can likewise be supported. Specifically, λ is in the region $\Lambda_{\rm T}$ if there exists a average service rate vector g_i^* such that there exists a control process which supports the rates λ.

For the CWN, we write λ_i as $\lambda_i = \lambda_i^{\rm t} + \sum_{s\in\mathcal{S}} \lambda_{is}^{\rm ts}$, where $\lambda_i^{\rm t}$ denotes the average number of supported input requests at terminal i that are processed at terminal i, and $\lambda_{is}^{\rm ts}$ represents the average number of supported input requests at terminal i that are forwarded from terminal i to server s. In addition, let $\lambda_i^{{\rm t}_s}$ denote the average number of supportable input requests processed at terminal i, when $b_{is} = 1$, and $\lambda_i^{\rm t} = \sum_{s\in\mathcal{S}} \lambda_i^{{\rm t}_s}$. In order to avoid multidimensional illustration of the results, we fix $|\mathcal{I}| = |\mathcal{S}| = 2$. For the channel model given in Section IX, the supportable rate region $\lambda_i^{t_2} + \lambda_{i2}^{\rm ts}$ vs. $\lambda_i^{t_1} + \lambda_{i1}^{\rm ts}$ is plotted as a dashed line in Fig. 2 and denoted as $\Lambda_{{\rm T}_i}$. For comparison, the subset of the region $\Lambda_{{\rm T}_i}$ in Fig. 2, illustrates the supportable arrival rate region for the channels between terminal i and servers, i.e., $\lambda_i^{t_1} = \lambda_i^{t_2} = 0$.

Let $\lambda_i^{\max}$ denote the maximum average number of requests that can be supported at terminal i in the CWN. It can be seen in Fig. 2, that $\lambda_i^{\max} = 8 + 7 = 15$. We have $\lambda_i^{\max} = \lambda_{\max_i}^{\rm t} + \sum_{s\in\mathcal{S}} \lambda_{\max_{is}}^{\rm ts}$, where $\lambda_{\max_i}^{\rm t}$ denote the maximum number of supported input requests at the terminal i processed at terminal i and $\lambda_{\max_{is}}^{\rm ts}$ represents the maximum number of supported input requests at terminal i forwarded from terminal i to server s. In Fig. 2, it can be seen that $\lambda_{\max_i}^{\rm t} = 0.5$ and $\sum_{s\in\mathcal{S}} \lambda_{\max_{is}}^{\rm ts} = 8 + 6.5 = 14.5$. Given $\lambda_i^{\max}$, the maximum supportable arrival rate at terminal i for the PSP and SSP cognitive networks is given as

$$\lambda_{\max_i}^{\rm cn} = \lambda_{\max_i}^{\rm t} p(I=1) + \sum_{s\in\mathcal{S}} \lambda_{\max_{is}}^{\rm ts} p(I=1)p(I_{\rm r}=1) + p(I=0)\mu_{i0}^{\max}. \tag{36}$$

For the channel model of the CWN given in Section IX, the unified supportable arrival rate region at terminals $(\Lambda_{\rm T})$ for

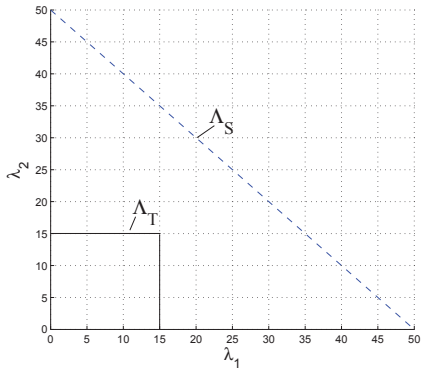


Fig. 3. The unified supportable arrival rate region at the terminals (Λ_T) and at the server (Λ_S).

both the PSP and SSP cognitive networks as well as for the CWN is now illustrated in Fig. 3.

B. Unified Arrival Rate Region at Servers

Let $\hat{g}_s^i$ denote the long-term average number of application requests of terminal i processed at server s, and $\hat{g}_s = \sum_{i\in\mathcal{I}} \hat{g}_s^i$ is the long-term average supportable rate at server s. The long-term average number of application requests $\hat{g}_s$ is not fixed and depends on control policy for choosing the actions.

Let Λ_S represent the supportable arrival rate region at server s. In order to calculate Λ_S, we consider only the set of policies that consume the whole $\hat{P}^{\max}$ at server s in each frame n. We use $\hat{\mathcal{O}}_s$ to represent the set of possible full power allocation options at server s, and $\hat{O}_s \in \hat{\mathcal{O}}_s$ denotes a full power allocation action at server s. One should note that $\hat{\mathcal{O}}_s \subset \hat{\mathcal{U}}_s$. Let $\hat{g}_s^{\max}$ denote the long-term full power average number of requests processed at server s. The set of full power average number of requests that can be supported at server s is

$$\hat{\Gamma} = \text{Conv}\{\hat{\mu}_s^1(\hat{O}_s)+\hat{\mu}_s^2(\hat{O}_s)+...,+\hat{\mu}_s^{|\mathcal{I}|}(\hat{O}_s)|\hat{O}_s \in \hat{\mathcal{O}}_s\}. \quad (37)$$

Specifically, the throughput region $\hat{\Gamma}$ can be viewed as the set of all full power long-term average service rates $\hat{g}_s^{\max}$ that a server can be configured to support. Thus, the supportable arrival rate region Λ_S at server s is the set of all average arrival rates $\sum_{i\in\mathcal{I}} \lambda_{is}^{\text{ts}}$ for which there exists a control policy $\hat{\pi}_s$ that satisfies

$$\sum_{i\in\mathcal{I}} \lambda_{is}^{\text{ts}} \leq \lim_{n\to\infty} \frac{1}{n} \sum_{\eta=1}^{n-1} \mathbb{E}\{\hat{\mu}_s(\eta)\} \leq \hat{g}_s \leq \hat{g}_s^{\max} \quad (38)$$

for some $\hat{g}_s^{\max} \in \hat{\Gamma}$ as rates below each point in $\hat{\Gamma}$ can likewise be supported.

For comparison, the supportable arrival rate region at server s, Λ_S, is illustrated in Fig. 3 together with Λ_T. Since Λ_T is a subset of Λ_S, it is clear that server s can support all arrival rates λ inside Λ_T. Thus, the network stability region Λ is equal to Λ_T. Stability region is unique for each network and it should not be mixed up with the stability region of a specific resource allocation policy. The stability region of a resource allocation policy is a closure of the set of arrival rate vectors λ that the policy can stably support and it is a subset of the network capacity region [17].

VII. COMPLEXITY ANALYSIS

In this section, we analyse the complexity of the dynamic control policy proposed in Section V. The complexity of solving MDPs using VIA has been also considered, for example, in [27] and [32]. However, unlike in [27], we would like to emphasize that our policy does not require any knowledge of the statistics of $a_i(n)$ which significantly decreases the computational complexity of the VIA.

In order to calculate the optimal policy in (29), we first need to calculate the rewards in (27) and then the optimal value functions in (28). It is easy to see that the complexity of calculating the optimal control policy depends not only on the sizes of $\mathcal{X}_i$ and $\hat{\mathcal{X}}_s$ but also on the number of feasible control options in each state $X_i \in \mathcal{X}_i$ and $\hat{X}_s \in \hat{\mathcal{X}}_s$. We start with defining the cardinality of $\mathcal{X}_i$ and $\hat{\mathcal{X}}_s$.

Let $|\mathcal{X}_i|$ and $|\hat{\mathcal{X}}_s|$ denote the number of states in $\mathcal{X}_i$ and $\hat{\mathcal{X}}_s$, respectively. In addition, let $|\mathcal{H}_i|$ denote the number of channel states in state space $\mathcal{H}_i$. For arrival rates inside Λ_T, we have $\limsup_{n\to\infty} y_i(\eta) = y_i^{\max}$ and $\limsup_{n\to\infty} \hat{y}_s^i(\eta) = \hat{y}_s^{\max}$ for all $i \in \mathcal{I}$ and $s \in \mathcal{S}$. The total number of states at terminal i is

$$|\mathcal{X}_i| = (y_i^{\max}+1)|\mathcal{H}_i|(\hat{y}_s^{\max}+1)^{|\mathcal{S}|} \quad (39)$$

and the total number of states at servers s

$$|\hat{\mathcal{X}}_s| = (\hat{y}_s^{\max}+1)^{|\mathcal{I}|}. \quad (40)$$

The rewards in (27) need to be calculated for each action $U_{X_i} \in \mathcal{U}_{X_i}$ in each state $X_i \in \mathcal{X}_i$. Let $|\mathcal{U}_{X_i}|$ and $|\hat{\mathcal{U}}_{\hat{X}_s}|$ denote the number of feasible control actions in each state $X_i \in \mathcal{X}_i$ and $\hat{X}_s \in \hat{\mathcal{X}}_s$, respectively. In addition, we use $\mu_{X_i}^{\max}$ to represent the maximum number of application requests that can be removed from the buffer of terminal i with power $P^{\max}$ in state X_i. The number of feasible actions in state X_i is then given as

$$|\mathcal{U}_{X_i}| = (|\mathcal{S}|+1)\min\{y_i, \mu_{X_i}^{\max}\}+1, \quad (41)$$

and the number of feasible actions in state $\hat{X}_s$ as

$$|\hat{\mathcal{U}}_{\hat{X}_s}| = |\mathcal{I}|\min\{\sum_i \hat{y}_s^i, \hat{\mu}_s^{\max}\}+1, \quad (42)$$

where $\hat{\mu}_s^{\max} = \hat{P}^{\max}/\alpha_s$. The total number of calculated rewards at terminal i and server s are now given as $\sum_{|\mathcal{X}_i|} |\mathcal{U}_{X_i}|$ and $|\hat{\mathcal{X}}_s||\hat{\mathcal{U}}_{\hat{X}_s}|$, respectively.

After calculating all the rewards, we get the optimal value functions $W^\star$ by calculating the value function in (28) l times for each state $X_i \in \mathcal{X}_i$ until the convergence happens. Thus, in order to get the optimal value functions, the value functions need to be calculated in total of $l|\mathcal{X}_i|$ times for terminal i and $l|\hat{\mathcal{X}}_s|$ times for server s. Given the optimal value functions, the optimal actions for each state $X_i \in \mathcal{X}_i$ ($\hat{X}_s \in \hat{\mathcal{X}}_s$) can now be calculated from (29).

It is important to note that if the dynamic policy required the knowledge of the arrival rate statistics, we could not calculate the optimal actions separately for each terminal i and server s. Then, the total number of network states would be given as

$$(y_i^{\max}+1)^{|\mathcal{I}|}|\mathcal{H}_i|(\hat{y}_s^{\max}+1)^{|\mathcal{S}|}(\hat{y}_s^{\max}+1)^{|\mathcal{I}||\mathcal{S}|}(a_i^{\max}+1)^{|\mathcal{I}|}. \quad (43)$$

When compared to (39) and (40), the number of states in (43) is considerably higher.

VIII. STABILIZING CONTROL POLICIES

In this section, we compare the performance of our dynamic policy with the performance of the randomized stationary policy presented in [17], [21]. We show that the performance of our dynamic policy is better than the performance of the stationary policy and prove that the frame based policy, that was argued to provide performance better than the stationary policy in [17], [21] cannot guarantee network stability.

A. Lyapunov Drift

Our stability analysis relies on Lyapunov drift that specifies a sufficient condition for the stability of a system with queues. This method is used to prove the stability of different policies in several publications, such as [15], [17], [21], [33], [34] and [35].

1) Lyapunov drift at terminal i: The maximum service rate at terminal i is given as

$$\mu^*_{\max_i} = \max\{\mu_i^{\max}, \mu_{i0}^{\max}\}, \tag{44}$$

where $\mu^*_{\max_i} = \max_{\{s\in\mathcal{S}, H_i\in\mathcal{H}_i\}} \mu_i(P_i) + \mu_{is}(P_{is}, h_{is})$ and $\mu_{i0}^{\max}$ is given in (34). Such a value exists because the arrival rates are bounded [15], [17], [21].

Consider the *K*-step dynamics of unfinished work at terminal i:

$$q_i(K) = q_i(0) + \sum_{n=0}^{K-1} a_i(n) - \sum_{n=0}^{K-1}[\acute{\mu}_i(n)^* + \sum_{s\in\mathcal{S}} b_{is}(n)\mu_{is}(n)^*]. \tag{45}$$

We can write (45) as

$$q_i(K) = y_i(0) + \sum_{n=1}^{K-1} a_i(n) - \sum_{n=0}^{K-1}[\acute{\mu}_i(n)^* + \sum_{s\in\mathcal{S}} b_{is}(n)\mu_{is}(n)^*], \tag{46}$$

where $y_i(0) = q_i(0) + a_i(0)$. By adding $a_i(K)$ on both sides of (46), we get

$$y_i(K) = y_i(0) + \sum_{n=1}^{K} a_i(n) - \sum_{n=0}^{K-1}[\acute{\mu}_i(n)^* + \sum_{s\in\mathcal{S}} b_{is}(n)\mu_{is}(n)^*], \tag{47}$$

where $y_i(K) = q_i(K) + a_i(K)$. Inserting $y_i = y_i(0)$, $\acute{\mu}_i^* + \mu_{is}^* = \frac{1}{K}\sum_{n=0}^{K-1}\acute{\mu}_i(n)^* + \sum_{s\in\mathcal{S}} b_{is}(n)\mu_{is}(n)^*$ and $a_i = \frac{1}{K}\sum_{n=1}^{K} a_i(n)$ into (47), we have

$$y_i(K) = y_i + Ka_i - K(\acute{\mu}_i^* + \mu_{is}^*). \tag{48}$$

Squaring both sides of (45), defining the Lyapunov function as $L(y_{\mathrm{T}}) = y_i^2$ and taking conditional expectations of the inequality given $y_{\mathrm{T}}(0)$, the *K*-step Lyapynov drift is given as:

$$\mathbb{E}\{L[y_{\mathrm{T}}(K)] - L[y_{\mathrm{T}}(0)]\big|y_{\mathrm{T}}(0)\} \le K^2M - 2Ky_i(0)\frac{1}{K}\times \Big[\sum_{n=0}^{K-1}\mathbb{E}\{\acute{\mu}_i(n)^* + \sum_{s\in\mathcal{S}} b_{is}(n)\mu_{is}(n)^*\big|y_{\mathrm{T}}(0)\} - \sum_{n=1}^{K}\mathbb{E}\{a_i(n)\big|y_{\mathrm{T}}(0)\}\Big]. \tag{49}$$

The above equation represents Lyapunov drift for any resource allocation policy that we can have for the system and

$$M = (\mu^*_{\max_i} + a_i^{\max})^2. \tag{50}$$

Since $y_i(K) = q_i(K) + a_i(K)$, where $q_i(K)$ is given in (45), the policy that minimizes $\frac{1}{K+1}\sum_{n=0}^{K}\mathbb{E}\{y_i(n)\}$ also minimizes $\frac{1}{K+1}\sum_{n=0}^{K}\mathbb{E}\{q_i(n)\}$.

We now define K as the number of frames required to reach the steady state behavior so that

$$\sum_{n=0}^{K}\mathbb{E}\{y_i(n)\} = \sum_{n=0}^{K+k}\mathbb{E}\{y_i(n)\}, \quad \forall \quad i\in\mathcal{I}, \tag{51}$$

where $k = \{1, ..., \infty\}$.

2) Lyapunov drift at server s: The maximum service rate of terminal i at server s is

$$\hat{\mu}_s^{\max_i} \triangleq \max_{\{i\in\mathcal{I}\}} \hat{\mu}_s^i(\hat{P}^{\max}). \tag{52}$$

The *K*-step dynamics of unfinished work at server s are given by

$$\hat{q}_s^i(K) = \hat{q}_s(0) + \sum_{n=0}^{K-1}\mu_{is}(n)^* - \sum_{n=0}^{K-1}\hat{\mu}_s^i(n), \tag{53}$$

that can be written as

$$\hat{y}_s^i(K) = \hat{y}_s^i(0) + \sum_{n=1}^{K}\mu_{is}(n)^* - \sum_{n=0}^{K-1}\hat{\mu}_s^i(n). \tag{54}$$

By defining Lyapunov function as $L(\hat{y}_{\mathrm{ST}}) = (\hat{y}_s^i)^2$, the *K*-step Lyapunov drift is then given as

$$\mathbb{E}\{L[\hat{y}_{\mathrm{ST}}(K)] - L[\hat{y}_{\mathrm{ST}}(0)]\big|\hat{y}_{\mathrm{ST}}(0)\} \le K^2\hat{M} - 2K\hat{y}_s^i(0)\frac{1}{K}\times \Big[\sum_{n=0}^{K-1}\mathbb{E}\{\hat{\mu}_s^i(n)\big|\hat{y}_{\mathrm{ST}}(0)\} - \sum_{n=1}^{K}\mathbb{E}\{\mu_{is}(n)^*\big|\hat{y}_{\mathrm{ST}}(0)\}\Big], \tag{55}$$

where $\hat{M}$ is given as

$$\hat{M} = (\hat{\mu}_s^{\max_i} + \mu_{is}^{\max})^2 \tag{56}$$

and $\mu_{is}^{\max}$ is defined in (20). Equation (55) represents the Lyapunov drift for any resource allocation policy yielding service rate $\hat{\mu}_s^i$ at server s. Since $\hat{y}_s^i(K) = \hat{q}_s^i(K) + \mu_{is}(K)^*$, the policy that minimizes $\max_{i\in\mathcal{I}}\Big\{\frac{1}{K+1}\sum_{n=0}^{K}\mathbb{E}\{\hat{y}_s^i(n)\}\Big\}$, also minimizes $\max_{i\in\mathcal{I}}\Big\{\frac{1}{K+1}\sum_{n=1}^{K}\mathbb{E}\{\hat{q}_s^i(n)\}\Big\}$.

B. Randomized Stationary Policy

In order to support every point in the network stability region described in Section VI, it is sufficient to consider only the class of stationary, randomized policies that take control decisions based on the current channel states only and does not consider current workloads. The randomized stationary policy was presented in [17] and it can be implemented only if the channel steady state probabilities and both the external arrival rates λ and the internal arrival rates c_{is}^* are known in advance. In this paper, the stationary policy will be used to analyze the performance of our dynamic control policy. The details on the

stability analysis and the implementation of a stationary policy can be found in [17], [21].

The average arrival rates of each terminal i and the average arrival rates of each terminal i at servers s are assumed to be within Λ, so that $\lambda_i+\theta \in \Lambda$ and $\lambda_{is}^{\text{ts}}+\theta \in \Lambda$. Then, $\lambda_i \leq g_i^*-\theta$ and $\lambda_{is}^{\text{ts}} \leq \hat{g}_s^i - \theta$. For the stationary policy, we can now have [17], [21]

$$\frac{1}{K}\sum_{n=0}^{K-1}\mathbb{E}\{\acute{\mu}_i(n)^* + \sum_{s\in\mathcal{S}} b_{is}(n)\mu_{is}(n)^*\} - \frac{1}{K}\sum_{n=1}^{K}\mathbb{E}\{a_i(n)\} \geq \frac{2\theta}{3} \tag{57}$$

for each terminal i and

$$\frac{1}{K}\sum_{n=0}^{K-1}\mathbb{E}\{\hat{\mu}_s^i(n)\} - \frac{1}{K}\sum_{n=1}^{K}\mathbb{E}\{\mu_{is}(n)^*\} \geq \frac{2\theta}{3}. \tag{58}$$

for each terminal i at server s. Inserting (57) and (58) into right hand side of (49) and (55), respectively, the queuing bounds for the stationary policy are given as

$$\limsup_{n\to\infty}\frac{1}{n+1}\sum_{\eta=0}^{n}\mathbb{E}\{q_i(\eta)\} \leq \limsup_{n\to\infty}\frac{1}{n+1}\sum_{\eta=0}^{n}\mathbb{E}\{y_i(\eta)\} \leq \frac{3KM}{4\theta} \tag{59}$$

for all $i \in \mathcal{I}$ and

$$\limsup_{n\to\infty}\frac{1}{n+1}\sum_{\eta=0}^{n}\mathbb{E}\{\hat{q}_s^i(\eta)\} \leq \limsup_{n\to\infty}\frac{1}{n+1}\sum_{\eta=0}^{n}\mathbb{E}\{\hat{y}_s^i(\eta)\} \leq \frac{3K\hat{M}}{4\theta} \tag{60}$$

for all $i \in \mathcal{I}$ and $s \in \mathcal{S}$.

C. Frame Based Policy

Frame based policy works like the dynamic policy, but updates the backlog information every K frames. Given (17) and (30), the frame based policy is then designed to maximize

$$\frac{1}{K}\sum_{n=0}^{K-1}\mathbb{E}\Big\{[y_i(0)-\hat{q}_s^i(0)]\frac{\sum_{s\in\mathcal{S}} b_{is}(n)\mu_{is}(n)^*}{\mu_{is}^{\max}} + y_i(0)[\acute{\mu}_i(n)^* + \sum_{s\in\mathcal{S}} b_{is}(n)\mu_{is}(n)^*]\Big\} \tag{61}$$

at each terminal i and

$$\frac{1}{K}\sum_{n=0}^{K-1}\sum_{i\in\mathcal{I}}\mathbb{E}\{\hat{y}_s^i(0)\hat{\mu}_s^i(n)\} \tag{62}$$

at each servers s.

It was argued in [17], [21] that since the frame based policy maximizes

$$\frac{1}{K}\sum_{n=0}^{K-1} y_i(0)\mathbb{E}\{\acute{\mu}_i(n)^* + \sum_{s\in\mathcal{S}} b_{is}(n)\mu_{is}(n)^*\} \tag{63}$$

and (62), the frame based policy is stable and its performance is better than the performance of any other network stabilizing policy.

Theorem 1: Frame based policy is not the best policy and it cannot guarantee network stability.

Proof: In order for a system to be stable, all the queues both at the terminals and servers must be stable [17]. It is easy to see that by maximizing the sum in (63), the frame based policy maximizes the right hand side of the Lyapunov drift in (49). However, when there are shared resources in the network, by maximizing only the sum in (62), the frame based policy cannot guarantee that the right hand side of (55) is maximized for each virtual queue of terminal i at server s.

Let us consider a simple example of a system with two terminals and a server s. By assuming that for the best network stabilizing policy we have

$$\sum_{n=1}^{K}\sum_{i=1}^{2}\mu_{is}(n)^* - \sum_{n=0}^{K-1}\sum_{i=1}^{2}\hat{\mu}_s^i(n) = 40, \tag{64}$$

$$\sum_{n=1}^{K}\mu_{1s}(n)^* - \sum_{n=0}^{K-1}\hat{\mu}_s^1(n) = 0{+}10{+}5{+}8{-}10{+}2{+}8{-}4{+}1 = 20 \tag{65}$$

and

$$\sum_{n=1}^{K}\mu_{2s}(n)^* - \sum_{n=0}^{K-1}\hat{\mu}_s^2(n) = 0{+}10{+}5{+}8{-}10{+}2{+}8{-}4{+}1 = 20. \tag{66}$$

It can be seen that if $\sum_{\eta=1}^{n}\mu_{is}(\eta)^* - \sum_{\eta=0}^{n-1}\hat{\mu}_s^i(\eta) = 23$, $\hat{y}_s^i(\eta+1)-\hat{y}_s^i(\eta) = \mu_{is}(\eta+1)^* - \hat{\mu}_s^i(\eta)$ is negative preventing (65) and (66) to get bigger than 23. However, since the frame based policy is designed to maximize only the sum in (62), it can allocate the resources so that

$$\sum_{n=1}^{K}\mu_{1s}(n)^* - \sum_{n=0}^{K-1}\hat{\mu}_s^1(n) = 0{+}10{+}5{+}8{-}0{+}2{+}8{-}0{+}1 = 34 \tag{67}$$

and

$$\sum_{n=1}^{K}\mu_{2s}(n)^* - \sum_{n=0}^{K-1}\hat{\mu}_s^2(n) = 0{+}10{+}5{+}8{-}20{+}2{+}8{-}8{+}1 = 6. \tag{68}$$

The frame based policy maximizes the sum $\sum_{n=1}^{K}\sum_{i=1}^{2}\mu_{is}(n)^* - \sum_{n=0}^{K-1}\sum_{i=1}^{2}\hat{\mu}_s^i(n) = 34+6 = 40$, but it cannot guarantee that the right hand side of (55) is maximized for each virtual queue of terminal i at server s. The frame based policy provides very small delay for terminal 2, but prevents terminal 1 to reach its steady state and stability.

■

D. Dynamic Control Policy

In this section, we show that our dynamic control policy offers performance better than the stationary policy and provides bounds on average delays at each terminal i and server s without requiring information of arrival statistics.

Theorem 2: Dynamic policy supports every point on the network stability region without requiring information of arrival

statistics. The performance of the dynamic policy is better than the performance of the randomized stationary algorithm.

Proof: Dynamic control policy is designed to maximize (17) at each terminal i and (30) at each servers s. Inserting $y_i(n) = y_i(0) + \sum_{\eta=1}^{n} a_i(\eta) - \sum_{\eta=0}^{n-1}[\acute{\mu}_i(\eta)^* + \sum_{s\in\mathcal{S}} b_{is}(\eta)\mu_{is}(\eta)^*]$ into (19) in (17) and $\hat{y}_s^i(n) = \hat{y}_s^i(0) + \sum_{\eta=1}^{n}\mu_{is}(\eta)^* - \sum_{\eta=0}^{n-1}\hat{\mu}_s^i(\eta)$ into (30), we see that the dynamic policy maximizes

$$\frac{1}{K}\sum_{n=0}^{K-1}\mathbb{E}\Big\{[y_i(n)-\hat{q}_s^i(n)]\frac{\sum_{s\in\mathcal{S}} b_{is}(n)\mu_{is}(n)^*}{\mu_{is}^{\max}}+ y_i(0)[\acute{\mu}_i(n)^*+\sum_{s\in\mathcal{S}} b_{is}(n)\mu_{is}(n)^*]+[\acute{\mu}_i(n)^*+\sum_{s\in\mathcal{S}} b_{is}(n)\mu_{is}(n)^*]\big[\sum_{\eta=1}^{n} a_i(\eta)-\sum_{\eta=0}^{n-1}[\acute{\mu}_i(\eta)^*+\sum_{s\in\mathcal{S}} b_{is}(\eta)\mu_{is}(\eta)^*]\big]\Big\} \tag{69}$$

at each terminal i and

$$\frac{1}{K}\sum_{n=0}^{K-1}\sum_{i\in\mathcal{I}}\mathbb{E}\Big\{\hat{y}_s^i(0)\hat{\mu}_s^i(n)+\hat{\mu}_s^i(n)\big[\sum_{\eta=1}^{n}\mu_{is}(\eta)^*-\sum_{\eta=0}^{n-1}\hat{\mu}_s^i(\eta)\big]\Big\} \tag{70}$$

at each servers s.

It can be seen in (69) that dynamic policy maximizes the right hand side of the Lyapunov drift in (49). In addition, it is easy to see in (70) that dynamic policy is designed to maximize the right hand side of (55) for each user i at server s. The dynamic policy allocates more CPU to a terminal with the longest queue and thus minimizes $\max_{i\in\mathcal{I}}\left\{\frac{1}{K}\sum_{n=0}^{K-1}\mathbb{E}\{\hat{q}_s^i(n)\}\right\}$ at server s. By comparing (62) and (70) it can also be seen that, unlike the frame based policy, our dynamic policy maximizes $\frac{1}{K}\sum_{n=0}^{K-1}\sum_{i\in\mathcal{I}}\mathbb{E}\{\hat{\mu}_s^i(n)\}$ so that $\frac{1}{K}\sum_{n=0}^{K-1}\mathbb{E}\{\hat{\mu}_s^i(n)\} - \sum_{n=1}^{K}\mathbb{E}\{\mu_{is}(n)^*\}$ on the right hand side of (55) is maximized for each virtual queue of terminal i at server s. Thus, the dynamic policy stabilizes the network and its performance is better that the performance of the stationary policy. The queueing bounds for the dynamic policy can now be given as in (59) and (60). ■

IX. PERFORMANCE EVALUATION

For illustration purposes, we have evaluated the performance of the dynamic control policy via simulations. The performance of the optimal dynamic transmission policy is illustrated in the presence of time varying workloads and uncertain channels both for CN and PC network as well as for CWN. It is shown that by adapting to the changes in network conditions, our control policy mitigates the effect of PSP and SSP cognitive networks on each other. The simulations support our stability analysis presented in Sections VI and VIII, and are implemented using Matlab.

A. Experiment Setup

Due to the complexity of the problem, we set $|\mathcal{I}| = |\mathcal{S}| = 2$. Although the simulations are run only for a small system, we would like to point out that the stability has been proven analytically for any size of the system in Section VIII. The channel process is generated according to a Markov chain and state transition matrix for the channel between terminal i and the hosting servers in the CWN is given as

$$\mathbf{T} = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \\ T_{41} & T_{42} & T_{43} & T_{44} \end{bmatrix} = \begin{bmatrix} 0.3 & 0.5 & 0.2 & 0 \\ 0.1 & 0.6 & 0.2 & 0.1 \\ 0.1 & 0.3 & 0.5 & 0.1 \\ 0 & 0.1 & 0.25 & 0.65 \end{bmatrix}, \tag{71}$$

where T_{kl} is the probability of transitioning from channel state k to l, and the corresponding stationary probabilities $p\{H_i = (|h_{11}|^2, |h_{12}|^2)\}$ are given as $p\{H_i = (10,\ 10)\} = 0.1$, $p\{H_i = (10,\ 20)\} = 0.4$, $p\{H_i = (20,\ 10)\} = 0.3$, $p\{H_i = (20,\ 20)\} = 0.2$.

For the SSP cognitive network, the probability that the channels between terminal i and the servers are available for communication is $p_H^{\mathrm{S}} = 0.9$ or $p_H^{\mathrm{S}} = 0.7$. The stationary probabilities are then given as $p\{H_i = (10,\ 10)\} = 0.09$, $p\{H_i = (10,\ 20)\} = 0.36$, $p\{H_i = (20,\ 10)\} = 0.27$, $p\{H_i = (20,\ 20)\} = 0.18$, $p\{H_i = (0,\ 0)\} = 0.1$ or $p\{H_i = (10,\ 10)\} = 0.07$, $p\{H_i = (10,\ 20)\} = 0.28$, $p\{H_i = (20,\ 10)\} = 0.21$, $p\{H_i = (20,\ 20)\} = 0.14$, $p\{H_i = (0,\ 0)\} = 0.3$. The probability that PU returns to the given channel is $p_{\text{return}}^{\mathrm{P}} = 0.05$.

In the PC network, where the overall resources include both cognitive and conventional links, we assume that the channel between terminal i and server 1 is cognitive and the channel between terminal i and server 2 is a non-cognitive channel. The probability that the channel between terminal i and server 2 is available for communication is $p_H^{\mathrm{S}} = 0.9$ or $p_H^{\mathrm{S}} = 0.7$. The stationary probabilities are given as $p\{H_i = (10,\ 10)\} = 0.09$, $p\{H_i = (10,\ 20)\} = 0.36$, $p\{H_i = (20,\ 10)\} = 0.27$, $p\{H_i = (20,\ 20)\} = 0.18$, $p\{H_i = (0,\ 10)\} = 0.05$, $p\{H_i = (0,\ 20)\} = 0.05$ or $p\{H_i = (10,\ 10)\} = 0.07$, $p\{H_i = (10,\ 20)\} = 0.28$, $p\{H_i = (20,\ 10)\} = 0.21$, $p\{H_i = (20,\ 20)\} = 0.14$, $p\{H_i = (0,\ 10)\} = 0.15$, $p\{H_i = (0,\ 20)\} = 0.15$. The probability that PU returns to the given channel between terminal i and server 2 is $p_{\text{return}}^{\mathrm{P}} = 0.05$.

For a Poisson process, the second moment of arrivals in each frame is finite [17]. Thus, each terminal is assumed to receive requests from applications according to a Poisson process at an average rate of λ_i. In the simulations, λ_i takes values between 1 to 8 requests/frame, and $\lambda_1 = \lambda_2$. The maximum available power at each terminal is $P_{\max} = 4W$. We use $\alpha_i = 0.6$ in (1), the discount factor α = 0.7 in (26) and $\alpha_{is} = 100$ in (2). The Lagrangian multiplier is fixed to $\beta_i = 1$. The long-term average sum power, sum delay, and sum throughput are calculated over $N = 20000$ frames.

Each CPU is assumed to follow a quadratic power-frequency relationship. Specifically, CPU is assumed to have a discrete set of frequency options in the interval [1.6GHz, ..., 2.6GHz] at increments of 0.2 GHz and the corresponding power consumption (in watts) at frequency f is given by $\hat{P}_{\min} + \theta(f -$

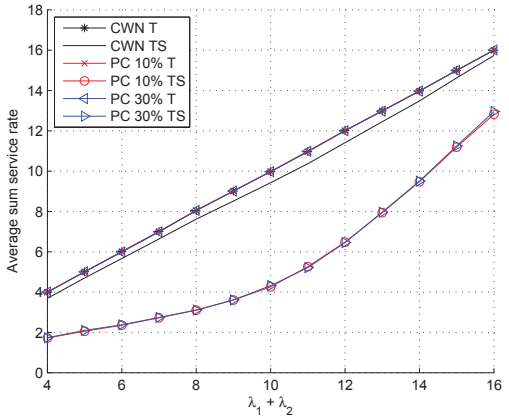
$1.6GHz)^2$ where $\hat{P}_{\min} = 10W$ and $\theta = 10W/(GHz)^2$. Thus, the CPU power consumption at the highest frequency is $20W$. At each utilization level f, the maximum supportable service rate $\mu_{is}^{\max}(f)$ is given in (5), where $\hat{\alpha}_s = 0.4$. Thus, on average, a server running at the minimum (maximum) speed can process 25 (50) requests/frame.

B. Numerical Results and Discussions

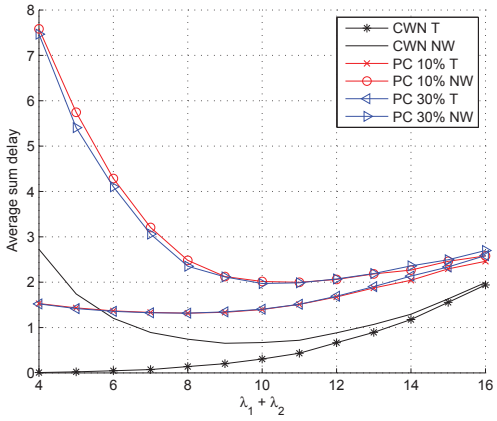
In the figures we have used the following notations: 'CWN' - conventional wireless network, 'CN' - cognitive network, 'PC' - partial cognitive network, 'T'- terminals, 'S'- servers, 'TS'- transmission from terminals to servers and 'NW' - entire network. In addition, '10%' and '30%' represent the probabilities that the channel between terminal i and server 1 is not available for communication.

The average sum service rates at the terminals (T) and the average sum rates from terminals to servers (TS) are plotted as a function of $\lambda_1 + \lambda_2$ for both the CWN and the PC network in Fig. 4(a). It can be seen in the figure, that the average sum service rates at the terminals both in the CWN and the PC network equal $\lambda_1 + \lambda_2$. In the CWN, almost all application requests are forwarded to be processed at the servers. In the PC network, the effect of PSP and SSP cognitive networks on each other is mitigated by processing considerably more requests at the terminals. If the channel between the terminal and server 1 is not available for communication, and if the channel between the terminal and server 2 is bad, the more requests are processed at the terminal, especially when the arrival rates are low. However, it also can be seen in Fig. 4(a), that the number of requests forwarded to the servers gets higher with the increase of $\lambda_1 + \lambda_2$. This is due to the smaller processing capabilities at the terminals than at the servers.

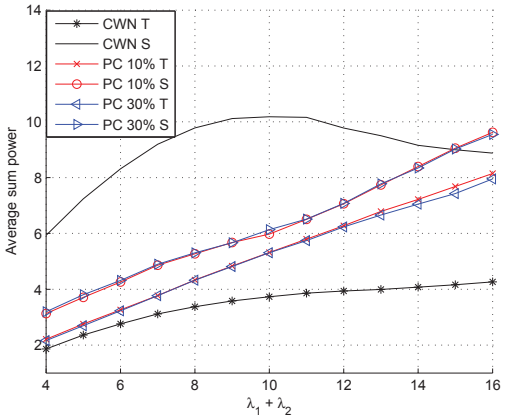
The average sum delays at the terminals (T) and the average sum delays over the entire network (NW) are plotted as a function of $\lambda_1 + \lambda_2$ for both CWN and PC network in Fig. 4(b). It can be seen, that for the given system parameters the processing delay at the servers decreases as $\lambda_1 + \lambda_2$ increases, when $\lambda_1 + \lambda_2 < 9$. This is because, at low arrival rates, the queues at the servers are short. Thus, in order to maximize (30), it is more advantageous to delay some of the requests in order to achieve energy savings at the server. When $\lambda_1 + \lambda_2$ is large, there is no much processing delay at the servers, because high arrival rates from the terminals encourage servers to empty their queues by increasing their processing capabilities. Due to the uncertain availability and reliability of the channel between the terminals and server 1 in the PC network, the delay at server 1 is longer in the PC network than in the CWN. Thus, also the overall network delay in the PC network is longer than that of the CWN. It can also be seen, that the overall network delay in the PC 30% network is a bit shorter than in the PC 10% network. This is due to the fact that, even if the channel between the terminals and server 1 is not available for communication, the channel between the terminals and server 2 is. In addition, the probability that the transmission over the given channel between terminal i and server 1 fails is smaller in the PC 30% network than in the PC 10% network, since $p(I = 1)p(I_{\mathrm{r}} = 0) = 0.7 \times 0.5 = 0.35$ and $p(I = 1)p(I_{\mathrm{r}} = 0) = 0.9 \times 0.5 = 0.45$.



(a) Average sum rates vs. $\lambda_1 + \lambda_2$.



(b) Average sum delays vs. $\lambda_1 + \lambda_2$.



(c) Average sum powers vs. $\lambda_1 + \lambda_2$.

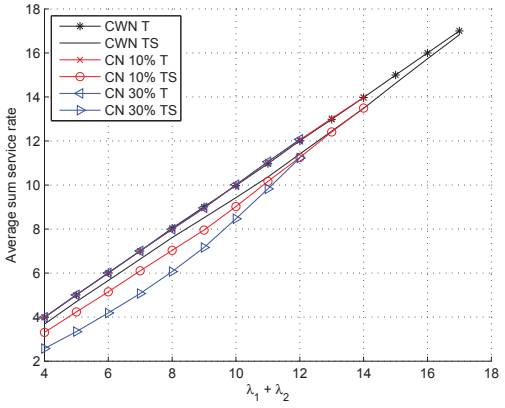
Fig. 4. Average sum rates, average sum delays and average sum powers of the optimal policy as a function of $\lambda_1 + \lambda_2$ for both CWN and PC network.

The average sum power consumptions both at terminals (T) and servers (S) are plotted as a function of $\lambda_1 + \lambda_2$ for CWN and PC network in Fig. 4(c). As most of the requests are processed at the servers in the CWN, the power consumption at the servers is significantly higher than the power consumption at the terminals. Due to the uncertain availability and reliability of the channel between the terminals and server 1 in the PC

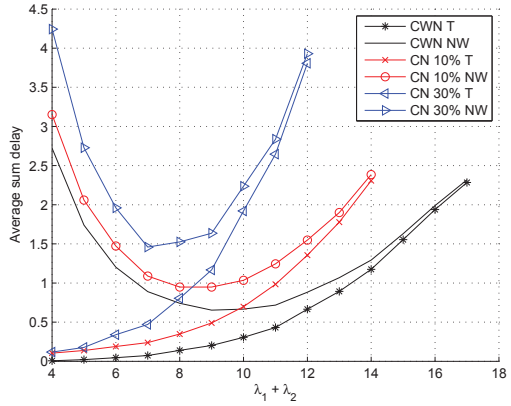
network, terminals consume more power in the PC network than in the CWN. If the channel between terminal i and server 1 is not available for communication, or if the channel between terminal i and server 2 is bad, it is more advantageous in terms of saving the transmission power to process more requests at the terminal. For the given range of $\lambda_1 + \lambda_2$, the power consumption at the servers in the CWN is smaller when $\lambda_1 + \lambda_2 \geq 13$ than when $7 < \lambda_1 + \lambda_2 < 13$. As mentioned earlier in this paper, the server consumes at least $\hat{P}_{\text{min}}$ even to process only a small amount of data. Thus, the active servers do not necessarily always process the maximum number of requests that could be processed with the used power, when $7 < \lambda_1 + \lambda_2 < 13$. If $\lambda_1 + \lambda_2$ is large, the used power can be better utilized in every frame, and more data can be processed with the lower power consumption. It can also be seen, that the average sum power in the PC 30% network is very close to the average sum power in the PC 10% network. This is because the channel between the terminal and server 2 is non-cognitive and the probability that the transmission over the given channel between terminal i and server 1 fails is smaller in the PC 30% network than in the PC 10% network, i.e., $p(I = 1)p(I_{\text{r}} = 0) = 0.7 \times 0.5 = 0.35$ and $p(I = 1)p(I_{\text{r}} = 0) = 0.9 \times 0.5 = 0.45$. In addition, due to the uncertain link availability and reliability between server 1 and the terminals, server 1 does not receive as many requests as server 2. However, as severs consumes at least $\hat{P}_{\text{min}}$ to process any amount of data, server 1 consumes almost equal amount of power as server 2. For the given arrival rates there is not enough requests to fully exploit the available power at server 1 and that is why the sum power consumption at the servers increases for all $\lambda_1 + \lambda_2$.

The average sum service rates at the terminals (T) and the average sum rates from terminals to servers (TS) are plotted as a function of $\lambda_1 + \lambda_2$ for both CWN and SSP cognitive network (CN) in Fig. 5(a). It can be seen that the average sum service rates at the terminals equal $\lambda_1 + \lambda_2$ for both networks supporting our stability analysis in Sections VI and VIII. However, due to the different network stability regions, the maximum supportable arrival rates in cognitive wireless networks is smaller than in the CWN. It can be seen, that the probability to process requests at the terminals is slightly higher in the CN than in the CWN, when arrival rates are low. This is due to the uncertain channel availability and reliability between the terminals and the servers. However, for high arrival rates, most of the requests are processed at the server only also in cognitive wireless network. For high arrival rates, it is more beneficial in terms of decreasing the transmission power and the delay to forward the application requests to the servers.

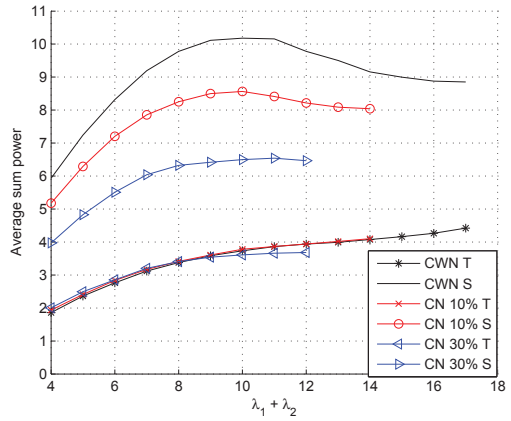
The average sum delays at the terminals (T) and average sum delays over the entire network (NW) are plotted as a function of $\lambda_1 + \lambda_2$ for both CWN and CN in Fig. 5(b). Due to the uncertain channel availability and reliability between the terminals and the servers, the delay in the CN is significantly longer than in the CWN. It can also be seen, that the processing delay at the servers decreases as $\lambda_1 + \lambda_2$ increases, when $\lambda_1 + \lambda_2$ is small. This is because, at low arrival rates, the queues at the servers are short. Thus, it is more advantageous to delay some of the requests in order to achieve energy savings at the server. When $\lambda_1 + \lambda_2$ is large, there is not much processing delay at the servers, because high arrival rates from the terminals encourage servers to empty their queues by increasing the capability to process the requests.



(a) Average sum rates vs. $\lambda_1 + \lambda_2$.



(b) Average sum delays vs. $\lambda_1 + \lambda_2$.



(c) Average sum powers vs. $\lambda_1 + \lambda_2$.

Fig. 5. Average sum rates, average sum delays and average sum powers of the optimal policy as a function of $\lambda_1 + \lambda_2$ for both CWN and SSP cognitive network.

The average sum power consumptions both at the terminals (T) and the servers (S) are plotted as a function of $\lambda_1 + \lambda_2$ for CWN and CN in Fig. 5(c). It can be seen in the figure, that

in the cognitive network our policy consumes approximately 10% or 30% less power at the servers than the policy consumes in the CWN. That is due to the uncertain channel availability between the terminal and the servers in the cognitive wireless network. It can also be seen, that the power consumption at the terminals in the cognitive network is slightly smaller or equal to power consumption in the CWN. The delay in the CN is significantly longer than in the CWN, since the terminals delay its requests waiting for the available channels or better channel conditions. Thus, the average power consumption at the terminals in the cognitive network is slightly smaller than in the CWN.

X. Conclusion

In this paper, we have considered a virtualized data center (computing cloud) consisting of a set of servers hosting a number of mobile terminals (a mobile cloud) and studied the problem of optimal resource allocation in the presence of time varying workloads and uncertain channels. The channel uncertainty is either due to fading and/or uncertain link availability and reliability in PSP/SSP cognitive networks. We have designed an optimal resource allocation policy that maximizes jointly utility of the long-term average throughput and minimizes the energy consumption, both at terminals and servers, while maintaining network stability. We have characterized the unified network stability region for both SSP and PSP cognitive networks as well as for the CWN, and presented a new unified stability analysis for the three networks. Under this model we have provided a new dynamic resource allocation policy that is shown to support every point on the network stability region without requiring information of arrival statistics. Performance evaluation has been carried out in order to compare the performance of optimal dynamic policy in the CWN with the performance of dynamic policy in the SSP/PSP cognitive wireless networks, and to validate the theoretical analysis of the paper. The results have shown that by adapting to the changes in network conditions, our dynamic policy can mitigate the impact of PSP and SSP cognitive networks on each other. We believe that the presented approach can be used as a performance benchmark and lays the foundation for future solutions of different simplified resource allocation schemes in VDC computing clouds.

References

[1] H. T. Dinh, C. Lee, D. Niyato and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Communications and Mobile Computing (WCMC)*, pp. 1587-1611, 2013.

[2] P. Padala, K-Y. Hou, K. G Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal and A. Merchant, "Adaptive control of virtualized resources in utility computing environments", in *Proceedings of EuroSys*, 2007.

[3] A. Greenberg, J. Hammilton, D. A. Maltz and P. Patel, "The cost of a cloud: Research problems in data center networks", *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, Jan. 2009.

[4] A. Wolke, M. Bichler and T. Setzer, "Planning vs. dynamic control: Resource allocation in corporate clouds", *IEEE Transactions on Cloud Computing*, no. 99, pp. 1-14, 2015.

[5] S. Glisic, B. Lorenzo, I. Kovacevic and Y. Fang, "Modeling dynamics of complex wireless networks", in *HPCS*, Helsinki, Finland, July 2013.

[6] H. Yue, M. Pan, Y. Fang and Savo Glisic, "Spectrum and energy efficient relay station placement in cognitive radio networks", *IEEE J. Select. Areas Commun.*, vol. 31, no. 5, May 2013.

[7] P. Padala, K-Y. Hou, K. G Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal and A. Merchant, "Automatic control of multiple virtualized resources", in *Proceedings of EuroSys*, 2009.

[8] X. Liu, X. Zhu, P. Padala, Z. Wang and S. Singhal, "Optimal multivariate control for differentiated service on a shared hosting platform", in *Proceedings of CDC*, Dec. 2007.

[9] B. Li, J. Li, J. Huai, T. Wo, Q. Li and L. Zhong, "EnaCloud: An energy-saving application live placement approach for cloud computing environments", in *IEEE International Conference on Cloud Computing*, Tampa, USA, 2009.

[10] D. Kusic and N. Kandasamy, "Power and performance management of virtualized computing environments via lookahead control", in *Proceedings of ICAC*, June 2008.

[11] S. Abdelwahed, N. Kandasamy, S. Singhal and Z. Wang, "Predictive control for dynamic resource allocation in enterprise data centers", in *Proceedings of IEEE RTAS*, 2004.

[12] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang and N. Gautam, "Managing server energy and operational cost in hosting centers", in *Proceedings of SIGMERICS*, June 2005.

[13] S. Govindan, J. Choi, B. Urgaongar, A. Sivasubramanian and A. Baldini, "Statistical profiling-based techniques for effective power provisioning in data centers", in *Proceedings of EuroSys*, Apr 2009.

[14] W. Xu, X. Zhu and S. Neema, "Online control for self-management in computing systems", in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2006.

[15] L. Georgiadis, M. J. Neely and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*, Foundations and Trends in Networking, Now Publisher, 2006.

[16] R. Urgaonkar, U. L. Kozat, K. Igarashi and M. J. Neely, "Dynamic resource allocation and power management in virtualized data centers", in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS)*, 2010.

[17] M. J. Neely, *Dynamic power allocation and routing for satellite and wireless networks with time varying channels*, Ph.D dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 2003.

[18] B. E. Collins and R. L. Cruz, "Transmission policies for time varying channels with average delay constraints", in *Proc. of Allerton Conf. on Commun., Control, and Comp.*, Monticello, IL, USA, 1999.

[19] D. Bertsekas, *Dynamic Programming and Optimal Control*, Vol. 1, 3rd ed., Athena Scientific, 2005.

[20] D. Bertsekas, *Dynamic Programming and Optimal Control*, Vol. 2, 3rd ed., Athena Scientific, 2007.

[21] M. J. Neely, E. Modiano and C. E. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks", *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89-103, Jan. 2005.

[22] C. Yang, Z. Chen, Y. Yao, B. Xia and H. Liu, "Energy efficiency in wireless cooperative caching networks", in *Proc. IEEE ICC*, 2014.

[23] R. Kaewpuang, D. Niytao, P. Wang and E. Hossain, "A framework for cooperative resource management in mobile cloud computing", *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2685-2700, Dec. 2013.

[24] J. Kephart, H. Chan, R. Das, D. Levine, G. Tesauro, F. Rawson and C. Lefurgy, "Coordinating multiple autonomic managers to achieve specified power-performance tradeoff", in *Proc. International Conf. on Autonomic Computing*, June 2007.

[25] D. Kusic and N. Kandasamy", "Control for dynamic resource provisioning in enterprise computing systems", in *Proc. International Conf. on Autonomic Computing*, 2006.

[26] E. Kalyvianaki, *Resource provisioning for virtualized server applications*, Computer Laboratory, University of Cambridge, Cambridge, United Kingdom, Tech. Rep., Nov. 2009.

[27] M. Goyal, A. Kumar and V. Sharma, "Optimal cross-layer scheduling of transmissions over a fading multiaccess channel", *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3518-3537, Aug. 2008.

[28] R. A. Berry and R. B. Gallager, "Communication over fading channels with delay constraints", *IEEE Trans. Inf. Theory*, vol. 50, no. 1, pp. 125-144, Jan. 2002.

[29] D. J. Ma, A. M. Makowski and A. Shwartz, "Estimation andoptimal control for constrained Markov chains", in *IEEE Conference on Desicion and Control*, 1986.

[30] M. Goyal and A. Kumar and V. Sharma, "Power constrained and delay optimal policies for scheduling transmissions over a fading channel", in *IEEE Infocom*, 2003.

[31] R. Bellman, *Dynamic Programmin*, Princeton University Press, Princeton, NJ, 1957.

[32] M. L. Littman, T. L. Dean and L. Pack Kaelbling", "On the complexity of solving Markov Decision Problems", in *Proc. of the 11th International Conference on Uncertainty in Artificial Intelligence*, 1995.
[33] E. Yeh and R. Berry, "Throughput optimal control of cooperative relay neworks", *IEEE Trans. Inf. Theory*, vol. 53, no. 10, pp. 3827-3833, Oct. 2007.
[34] H. Halabian, I.Lambaris and C. Lung", "Network capacity region of multi-queue multi-server queuing system with time varying connectivities", in *ISIT*, 2010.
[35] J. Jose, L. Ying and S. Wishwanath", "On the stability region of amplify-and-forward cooperative relay networks", in *ITW*, Oct. 2009.

PLACE PHOTO HERE

Maria Kangas is currently a doctoral researcher and a project manager in the area of wireless networks at Centre for Wireless Communications (CWC) at University of Oulu, Finland. She received her M.Sc. degree in Telecommunication Engineering from University of Oulu, Finland, in 2007. She was a visiting Phd student at the Rice University, Houston, Texas during 1.11.2008-1.5.2009 and 1.1.2010-1.5.2010. Her research interests include dynamic programming, network stability, heterogeneous networks, radio resource management, network optimization theory, network topology control, opportunistic communications, social networks and complex networks.

PLACE PHOTO HERE

Savo Glisic (M′90-SM′94) is a Professor of Telecommunications at University of Oulu, Finland, head of the networking research group, and Director of Globalcomm Institute for Telecommunications. He was Visiting Scientist at Cranfield Institute of Technology, Cranfield, U.K.(1976-1977) and University of California, San Diego (1986-1987). He has been active in the field wireless communications for 30 years and has published a number of papers and books. The incoming book "Advanced wireless Networks: 5G/6G, 3E, John Wiley and Sons, 2015" covers the enabling technologies for the definition of 5G/6G systems. He is also running an extensive doctoral program in the field of wireless networking. His research interest is in the area of network optimization theory, network topology control and graph theory, cognitive networks and game theory, radio resource management, QoS and queuing theory, networks information theory, protocol design, advanced routing and network coding, relaying, cellular, WLAN, ad hoc, sensor, active and bio inspired networks with emphasis on genetic algorithms. Dr. Glisic has served as the Technical Program Chairman of the third IEEE ISSSTA′94, the eighth IEEE PIMRC′97, and IEEE ICC′01. He was Director of IEEE ComSoc MD programs.

PLACE PHOTO HERE

Yuguang "Michael" Fang (S′92-M′97-SM′99-F′08) received a Ph.D. degree in Systems Engineering from Case Western Reserve University in January 1994 and a Ph.D degree in Electrical Engineering from Boston University in May 1997. He was an assistant professor in the Department of Electrical and Computer Engineering at New Jersey Institute of Technology from July 1998 to May 2000. He then joined the Department of Electrical and Computer Engineering at University of Florida in May 2000 as an assistant professor, got an early promotion to an associate professor with tenure in August 2003 and to a full professor in August 2005. He holds a University of Florida Research Foundation (UFRF) Professorship from 2006 to 2009, a Changjiang Scholar Chair Professorship with Xidian University, Xian, China, from 2008 to 2011, and a Guest Chair Professorship with Tsinghua University, China, from 2009 to 2012. He has published over 250 papers in refereed professional journals and conferences. Dr. Fang received the National Science Foundation Faculty Early Career Award in 2001 and the Office of Naval Research Young Investigator Award in 2002, and is the recipient of the Best Paper Award in IEEE International Conference on Network Protocols (ICNP) in 2006 and the recipient of the IEEE TCGN Best Paper Award in the IEEE High-Speed Networks Symposium, IEEE Globecom in 2002. Dr. Fang is also active in professional activities. He is a Fellow of IEEE and a member of ACM. He is currently serving as the Editor-in-Chief for IEEE Wireless Communications and serves/served on several editorial boards of technical journals including IEEE Transactions on Communications, IEEE Transactions on Wireless Communications, IEEE Wireless Communications Magazine and ACM Wireless Networks. He was an editor for IEEE Transactions on Mobile Computing and currently serves on its Steering Committee. He has been actively participating in professional conference organizations such as serving as the Steering Committee Co-Chair for QShine from 2004 to 2008, the Technical Program Vice-Chair for IEEE INFOCOM 2005, Technical Program Symposium Co-Chair for IEEE Globecom 2004, and a member of Technical Program Committee for IEEE INFOCOM (1998, 2000, 2003-2010).

PLACE PHOTO HERE

Pan Li received the B.E. degree in Electrical Engineering from Huazhong University of Science and Technology, Wuhan, China, in 2005, and the Ph.D. degree in Electrical and Computer Engineering from University of Florida, Gainesville, in 2009, respectively. Since Fall 2015, he has been with the Department of Electrical Engineering and Computer Science at Case Western Reserve University. He was an Assistant Professor in the Department of Electrical and Computer Engineering at Mississippi State University between August 2009 and August 2015. His research interests include network science and economics, energy systems, security and privacy, and big data. He has been serving as an Editor for IEEE Journal on Selected Areas in Communications – Cognitive Radio Series and IEEE Communications Surveys and Tutorials, a Feature Editor for IEEE Wireless Communications, and a Technical Program Committee (TPC) Co-Chair for Ad-hoc, Mesh, Machine-to-Machine and Sensor Networks Track, IEE VTC 2014, Physical Layer Track, Wireless Communications Symposium, WTS 2014, and Wireless Networking Symposium, IEEE ICC 2013. He received the NSF CAREER Award in 2012 and is a member of the IEEE and the ACM.