Adaptive Service Rate and Vacation Length for Energy-efficient HeNB based on Queueing Analysis

Hong Zhang, Jun Cai, Senior Member, IEEE, Xiaolong Li, and Shiwei Huang

Abstract—In order to improve the energy efficiency of small cell networks, in this paper, we analyze the operating procedure of home base stations, i.e., HeNBs, in femtocells, by introducing an MAP/PH/1/k queueing model. In this analytical model, the HeNB’s power on/off is represented as the alternative service and vacation periods. Also, the hybrid access mode defined in 3GPP Release 9 is considered, which involves both high priority and low priority users. Based on the analytical results, an adaptive service rate and vacation length (ASV) method is proposed to maximize the HeNB’s energy efficiency while satisfying its QoS requirements such as blocking probability and user waiting time. Simulation results demonstrate that the proposed ASV method is more effective in improving energy efficiency compared to the counterpart.

Index Terms—Small cell networks, femtocells, energy efficiency, hybrid access, vacation queue

I. INTRODUCTION

Currently, Information and Communications Technologies (ICT) industry occupies a major share of human electricity consumption. Particularly, the resultant excessive carbon dioxide emissions have raised a global concern. Thus, minimizing energy usage and carbon footprint from ICT becomes crucial. Driven by demands of higher data rates and enhanced energy efficiency, femtocells (FCs) were introduced for future wireless communication networks [1]. In a FC, the base station (BS), also known as Home Evolved Node B (HeNB), is typically a consumer-deployed cellular access point, which can provide better link quality for home users with low transmit power and can offload traffic from macrocells. In 3GPP Release 9 [2], a novel hybrid access concept is introduced for the HeNB, which requires service provisioning to both open access users and closed subscriber group (CSG) users. With the hybrid access, mobile users belonging to the CSG receive preferential service over unsubscribed users from open access.

In traditional cellular networks, since the BSs in practice can barely operate in a consistently functioning condition when there are few users in their coverage, energy saving becomes available by switching off some underutilized BSs. In [3], Marsan et al. investigated the feasibility in reducing the number of active BSs by considering day-night behavior of mobile traffic. In [4], the effect of changing cell size, called cell breathe, was studied, and the optimal cell size was derived based on data rates and traffic demands. A similar proposal on dynamically adjusting cell-size to achieve energy saving and load balancing was introduced in [5], where a concept called cell zooming was introduced to change the cell size based on traffic variations. Wang et al. in [6] discussed the duty cycle of BS on/off and proposed an antenna selection criterion. In [7], the power saving ratio was derived by turning off some BSs during low traffic periods. A distance aware BS on/off scheme was proposed in [8], where BS on/off depended on distances between BSs and associated mobile users. In [9] and [10], Zhang et al. investigated a clustering BS-off scheme, and designed a virtual small networking (VSN) protocol to improve network energy saving. In [11], Han et al. discussed a scalable BS switching scheme to extend network coverage to include service areas of inactive BSs. However, since the traditional BSs in macrocells have wide coverages and serve a large number of users, it’s very complicated in practical implementation to switch on/off the macro-BSs by considering the challenges such as coverage holes and user associations.

In contrast to the traditional BSs in macrocells, FCs typically have small coverages with fewer users. Thus, it is imperative to introduce sleep mode and power control in order to save energy consumption at FCs. Saker et al. in [12] proposed optimal sleep and wake up schemes for macro-femto heterogeneous networks, where FCs worked in open access mode and could offload traffic from the macrocells. The basic idea was that when the macro-femto heterogeneous network was lightly loaded so that the macro-BS could handle all the traffic alone, the FCs were switched off. Vereecken et al. in [13] proposed heuristic BS on/off decisions based on the number of user connections to different BSs. However, the strategy assumed that both users’ and BSs’ locations were known. Estrada et al. in [14] studied optimal resource allocation in two-tier networks with hybrid access BSs. The authors claimed that effective spatial reuse between the macrocells and FCs could be achieved by joint power control in both tiers.

While unlike the previous research which mainly addressed the joint operations between macrocells and FCs, other studies in [15]–[17] focused on the operations of FCs only. In [15], Li et al. proposed a simple sleeping scheme, called fixed time sleeping, for saving FC energy. However, the fixed time sleeping scheme may not be well adapted to the traffic variations in practical networks. Ge et al. in [16] conducted
performance analysis for two-tier FC networks with open access users. Based on derived Markov chain models, the spectrum and energy efficiency were analyzed under different scenarios in terms of the number of FCs, the average number of users, and the number of open channels. Kim et al. in [17] studied the effects of the FCs’ sleeping ratio on the energy efficiency by using a stochastic geometry-based model and derived the optimal FC sleeping ratio to maximize the energy efficiency. However, neither [16] nor [17] addressed the issues of the effects of users’ priorities or the constraints on users’ delay and blocking rates.

Although existing studies considering the HeNB power on/off have successfully enhanced the system energy saving, these studies have not jointly considered the hybrid access and the HeNB on/off characteristics, which are essential for practical implementations. By considering user priorities, e.g., when the high priority user from CSG can interrupt the HeNB’s sleep periods, the HeNB on/off scheduling becomes more complicated in order to balance the energy saving and QoS provisioning in terms of the queue lengths, the user waiting time and blocking rates for different types of users. To address all these issues, in this paper, the HeNB on/off mechanism is first analyzed by formulating a queueing model. The formulated discrete time Markov chain models the hybrid access and transmitter power on/off mechanism in order to understand how the service of priority users can affect the HeNB on/off transitions. Then, we evaluate some key performance metrics such as user waiting time, blocking probability and system busy cycle. After that, we propose an adaptive service rate and vacation length (ASV) method, which can improve the HeNB energy efficiency while guaranteeing QoS provisioning.

The main contributions of this paper are summarized as follows.

- An MAP/PH/1/k queueing model is formulated to capture the behavior of the HeNB with multiple vacations, exhaustive service discipline, and non-preemptive priority. Key performance metrics resulting from queue length, user waiting time, blocking probability and system busy cycle are derived.
- In contrast to the existing literature [20]–[22], in this paper, the formulated MAP/PH/1/k queue captures user priority and vacations jointly in order to better match the HeNB characteristics in terms of hybrid access mode and transmitter power on/off mechanism. In addition, the model integrates the vacation termination policy with a preference to high priority users.
- An ASV method is proposed to address the problem of maximizing system energy efficiency by considering the HeNB on/off and user priorities under the QoS requirements. The problem is then solved by the Lagrange decomposition method, where the adaptive service rate is derived in the inner optimization, and the adaptive vacation length is derived in the outer optimization.
- A one-step look-ahead method is proposed in order to reduce the computational complexity of the ASV method. The one-step look-ahead method is based only on buffer size and the number of admissible service rates in the current stage, so that it is more computational effective compared to the ASV method.
- Extensive simulation results are provided to demonstrate the efficiency and effectiveness of the proposed adaptive service rate and vacation length strategy.

The rest of the paper is organized as follows. In Section II, the system model is described. The queueing model and the associated performance measures are derived in Section III. In Section IV, we formulate the problem of maximizing the HeNB energy efficiency and proposed the ASV and one-step look-ahead methods. Simulation results are demonstrated in Section V followed by conclusions in Section VI. Table I summarizes the important notations in this paper for ease of reference.
II. SYSTEM MODEL

Consider a macrocell-based cellular network with an overlaid HeNB deployment for covering hotspots or indoor areas. The HeNB enables hybrid access mode and transmit power on/off mechanism [2], [18]. The hybrid access mode consists of two user groups, i.e., high priority users (HP users) from the CSG and low priority users (LP users) from open access. These two user groups are different in terms of arrival rate, service rate and priority of service. Define \( R = \{ r_i \}, i = 0, 1, \ldots, N \) and \( N < \infty \), as a finite set of available service rates at the HeNB in ascending order, and \( P = \{ p_i \}, i = 0, 1, \ldots, N \), as the corresponding consumed power when the service rate is \( r_i \). In the transmit power on/off mechanism, there are two HeNB operation modes, i.e., transmitter power on (service mode) and off (vacation mode)\(^1\). At the beginning of each vacation period, the HeNB will choose a suitable vacation length from a finite set of available vacation lengths, \( V \). Let \( D_{0} \) denote the arrival phase of both types of users. The HeNB maintains a buffer of size \( k, k < \infty \), for both HP and LP users. The arriving users will join the queue if the buffer has the vacancy or be blocked otherwise. A complete buffer sharing policy is adopted, which means there is no buffer reservation for any type of users. The users in the same type are served according to the arrival order, i.e., first-come first-serve (FCFS). A non-preemptive discipline is adopted such that no service for an LP user is going to be started if there is an HP user in the system. However, if the service of an LP user has started, the current service of the LP user cannot be interrupted [22]. In addition, the queueing model of the HeNB implements exhaustive service and unaged vacation, which needs to satisfy the following disciplines.

- **Exhaustive service**: all users should be served including those waiting in the queue and those arrive after the server returns from vacation.
- **Ungated and multiple vacations**: the HeNB cannot take any vacation until the buffer becomes empty. When the buffer is still empty after a vacation, the HeNB starts another vacation immediately.
- **LP user arrival during vacation**: the HeNB has no obligation to terminate a vacation and start to serve the LP users that arrive during the vacation period. Nevertheless, the newly arrived LP users should be buffered in the queue.

III. QUEUEING ANALYSIS

In this section, we formulate the queueing model and derive the key performance measures including queue length, user waiting time, blocking probability and system busy cycle.

A. Queueing Model Formulation

The aforementioned system paradigm can be modeled as an MAP/PH/1/k queue with multiple vacations and non-preemptive priorities. Let \( l_{m} \in \{0, 1, \ldots, k\} \), \( m = 1, 2 \), indicate the number of type-\( m \) users in the system, and \( l_{1} + l_{2} \leq k \). Let \( z \) denote the arrival phase of both types of users, and \( \theta = \{ s_{m}, v \} \) denote the phase of server working mode, where \( s_{m} \) represents the phase of service for type-\( m \) users and \( v \) represents the phase of vacation. Note that since the server is in either service or vacation mode, only one of \( s \) and \( v \) can be referred at any time instant. Then, the behavior of the system can be described by a four-dimensional stochastic process as \( \Delta = (l_{1}, l_{2}, z, \theta) \) with \( 0 \leq l_{1} \leq k, 0 \leq l_{2} \leq k - l_{1} \).

We can classify the system state space \( \Delta \) into four categories: (i) the system is empty and on vacation, (ii) the system is in service and only the LP users exist in the system, (iii) type \( m \) users is \( \mu_{m}^{-1} = \beta_{m} (I - S_{m})^{-1} e \), where \( I \) denotes the identity matrix.

Each HeNB power off period may consist of multiple vacations. Each vacation follows a PH distribution denoted by \( (\delta, V) \) of order \( n_{e} \). The mean vacation length is \( v^{-1} = \delta (I - V)^{-1} e \). In addition, to better match the differential user priorities in the HeNB hybrid access mode, it is allowed that an HP user arrival can interrupt the server vacations, called ‘vacation termination’ policy. We use stochastic matrix \( Q \) of dimension \( n_{v} \times 1 \) to depict the vacation termination process.

The HeNB maintains a buffer of size \( k, k < \infty \), for both HP and LP users. The arriving users will join the queue if the buffer has the vacancy or be blocked otherwise. A complete buffer sharing policy is adopted, which means there is no buffer reservation for any type of users. The users in the same type are served according to the arrival order, i.e., first-come first-serve (FCFS). A non-preemptive discipline is adopted such that no service for an LP user is going to be started if there is an HP user in the system. However, if the service of an LP user has started, the current service of the LP user cannot be interrupted [22]. In addition, the queueing model of the HeNB implements exhaustive service and unaged vacation, which needs to satisfy the following disciplines.

- **Exhaustive service**: all users should be served including those waiting in the queue and those arrive after the server returns from vacation.
- **Ungated and multiple vacations**: the HeNB cannot take any vacation until the buffer becomes empty. When the buffer is still empty after a vacation, the HeNB starts another vacation immediately.
- **LP user arrival during vacation**: the HeNB has no obligation to terminate a vacation and start to serve the LP users that arrive during the vacation period. Nevertheless, the newly arrived LP users should be buffered in the queue.

1. Describing the HeNB non-operational mode has multiple terms in the literature, e.g., power off mode, dormant or sleep mode. In this paper, we adopt the term ‘vacation mode’ to better match the formulation of queueing model.
the system is in vacation with the LP users in the system, and iv) there are HP users in the system. In summary, the state space can be rewritten as

$$\Delta = \Delta_0^v \cup \Delta_0^{k} \cup \Delta_1 \cup \Delta_2$$

(1)

where $\Delta_0^v = (0, 0, z, v)$ means an empty system with arrival in phase $z$ and the server is on vacation in phase $v$. $\Delta_0^{k} = (0, l_2, z, v)$, $l_2 \in [1, k]$, indicates that the server is on vacation of phase $v$ with only the LP users waiting in the system. $\Delta_1 = (0, l_2, z, s_2)$, $l_2 \in [1, k]$, means there is no HP user and at least one LP user exists in the system with service in phase $s_2$. $\Delta_2 = (l_1, l_2, z, s_m)$, $l_1, l_2 \in [0, k-l_1]$, represents there are at least one HP user and $l_2$ LP users in the system with one user of type-$m$ in service and the server is in phase $s_m$. Note that $\Delta_2$ also includes the case when there are only HP users in the system with $l_2 = 0$ and $m = 1$. Note that according to the vacation termination policy, if an HP user joins in the queue when the system is on vacation, the server has to exit current vacation mode and the system state will transit from $\Delta_1^v$ to $\Delta_2$.

The state transitions of the constructed MAP/PH/1/k queue with user priorities are shown in Fig. 1, where the numbers in circles represent $(l_1, l_2)$, and $l = l_1 + l_2$ indicates the total number of users in the system. The state transition probability matrix $P$ of the modeled MAP/PH/1/k queue can be described as

$$P = \begin{bmatrix}
A_0 & B_0 & C_0 & A_1 & B_1 & C_1 & \cdots & A_2 & B_2 & C_2 & \cdots & A_{k-2} & B_{k-2} & C_{k-2} & A_k
\end{bmatrix}$$

(2)

where $A_n$, $n = 0, 1, \ldots, k$, $B_n$, $n = 1, 2, \ldots, k-1$, and $C_n$, $n = 0, 1, \ldots, k-1$, are submatrices. $A_n$ denotes the transitions under the condition that there are $l_1 = n$ HP users in the system. $B_n$ denotes the transitions when an HP user arrives (i.e., $l_1 = n \rightarrow l_1 = n+1$) in the system and there is no service completion, and $C_n$ denotes the transitions when the service of an HP user is completed (i.e., $l_1 = n \rightarrow l_1 = n-1$). All blank areas are zeros and only $A_n$ are square matrices. $A_n$, $B_n$, and $C_n$ involve the states in $\Delta_0^v \cup \Delta_1 \cup \Delta_1^v$, $\Delta_2$, and $\Delta_0^{k} \cup \Delta_1 \cup \Delta_2$, respectively.

We use submatrix $A_0$ as an example to show the derivation of the transition matrix. Note that $A_0$ denotes the probabilities that system states transform from $(0, l_2)$ to $(0, l_2')$ when there are no HP users involved in state transitions. The derivation of $A_0$ consists of following seven cases.

- **Case I.** Empty system with the server on vacation, i.e., $(l_1, l_2) = (0, 0)$
  In case I, there is no arrival from either class of users (the event probability is $D_0$). The server may be in the middle of vacation (with probability $V$), or at the end of one vacation so that the system starts another vacation due to empty buffer (with probability $\nu \delta$). Thus, the transition matrix in case I can be written as
  $$A_0^{00} = \begin{bmatrix} 0, & D_0 \otimes (\nu \delta + V) \end{bmatrix}$$

(3)

where $\otimes$ denotes Kronecker product, and 0 denotes that the probability of the server in service is 0.

- **Case II.** State transition from $(l_1, l_2) = (0, 0)$ to $(0, 1)$
  In case II, a new LP user arrives with a probability $D_0 1$. Due to the low priority of the new arrival, the server may keep staying in the vacation (with probability $V$) or finish the current vacation and be ready to serve the LP user in the next slot (with probability $\nu \beta_2$). We can represent the transition matrix in case II as
  $$A_0^{10} = \begin{bmatrix} D_0 \otimes (s_2 \delta) & 0 \end{bmatrix}$$

(4)

- **Case III.** State transition from $(l_1, l_2) = (0, 1)$ to $(0, 0)$
  In this case, there is no arrival of either HP or LP user (with probability $D_0$). The server completes service for one LP user and starts vacation (with probability $s_2 \delta$) since the buffer becomes empty after the service. In this case, the submatrix for transitions can be represented as
  $$A_0^{10} = \begin{bmatrix} 0 & D_0 \otimes (s_2 \delta) \end{bmatrix}$$

(5)

- **Case IV.** State transition from $(l_1, l_2) = (0, l_2')$ to $(0, l_2' + 1)$, $l_2' \in [1, k-1]$
  This case considers the events that there is one new arrival of the LP user and no service is finished. The unfinished service may result from i) incomplete service if the server is in service mode, ii) incomplete vacation, or iii) a vacation ends and service starts for the LP users. We can summarize the transition submatrix in case IV as
  $$A_0^{10} = \begin{bmatrix} D_0 \otimes (s_2 \delta) & 0 \\ D_0 \otimes (\nu \beta_2) & D_0 \otimes V \end{bmatrix}$$

(6)

- **Case V.** System state stays at $(l_1, l_2) = (0, l_2')$
  This case consists of a few sub-scenarios: i) no arrival with incomplete service (with probability $D_0 \otimes S_2$), ii) one arrival from the LP users with one complete service (with probability $D_0 1 \otimes (s_2 \beta_2)$), iii) no arrival with vacation completion and service starts for the LP users (with probability $D_0 \otimes (\nu \beta_2)$), or iv) no arrival with incomplete vacation (with probability $D_0 \otimes V$). In summary, the transition submatrix for case V is
  $$A_0^{10} = \begin{bmatrix} D_0 \otimes S_2 + D_0 1 \otimes (s_2 \beta_2) & 0 \\ D_0 \otimes (\nu \beta_2) & D_0 \otimes V \end{bmatrix}$$

(7)

- **Case VI.** State transition from $(l_1, l_2) = (0, l_2')$ to $(0, l_2' - 1)$, $l_2' \in [2, l_2]$
  In this case, the transitions mean a service for one LP user has been completed. Since there are LP users remaining in the buffer, the server will stay in the service mode. Obviously, there is no arrival in this case. The transition submatrix is
  $$A_0^{2} = \begin{bmatrix} D_0 \otimes (s_2 \beta_2) & 0 \\ 0 & 0 \end{bmatrix}$$

(8)

- **Case VII.** The system stays at the boundary state $(l_1, l_2) = (0, k)$
  Since the system reaches buffer limit, the system state can remain unchanged if there is no arrival or a new arrival is blocked. The server can be in the service for the LP
users, starting to serve the LP users after vacations, or in the middle of a vacation because there is no HP user in the system. By jointly considering all these scenarios, the transition submatrix becomes

\[
A_{01}^{11} = \begin{bmatrix}
D_{0,1} \otimes (s_2 \beta_2) + D \otimes S_2 & 0 \\
D \otimes (v \beta_2) & D \otimes V
\end{bmatrix}
\]

where \(s_2 = e - S_2 e\) and \(v = e - V e\).

In summary, the transition matrix \(A_0\) can be written as

\[
A_0 = \begin{bmatrix}
A_0^{00} & A_0^{01} \\
A_0^{10} & A_0^{11}
\end{bmatrix}
\]

where \(A_0\) is of \(D(k+1) \times (k+1)\), and \(D\) denotes the dimension with respect to the number of submatrices.

Note that vacation termination may be triggered when the HP user arrives. Therefore, to explain the effect of this termination policy, we consider the state transition \(B_0^{01}\) in submatrix \(B_0\) as an example. \(B_0^{01}\) represents the state transition from \((l_1, l_2) = (0, 0)\) to \((1, 1)\). Since the event of an HP user arrival may happen just at the end of a vacation or during a vacation period, this results in the service of the HP user starting immediately or after vacation interruption. Such two scenarios can be jointly described as \((v \otimes \beta_1 + V Q \otimes \beta_1)\).

Thus, \(B_0^{01}\) can be derived as

\[
B_0^{01} = \begin{bmatrix}
D_{1,1} \otimes (v + V Q) \otimes \beta_1 & 0
\end{bmatrix}.
\]

The detailed formulae of all submatrices in transition matrix \(P\) are shown in Appendix A.

B. Steady State Distribution

The system is stable only if the traffic intensity \(\rho = \frac{\lambda_1 + \lambda_2}{\mu_1 + \mu_2} < 1\).

Let \(x_{i,j}\) denote the probability that there are \(i\) HP and \(j\) LP users in the system. Note that \(i + j \leq k\). We further define the finite stationary probability vector \(x = \{x_0, x_1, x_2, \ldots, x_k\}\) as

\[
x = \begin{cases}
x_0 = \{x_{0,0}, x_{0,1}, x_{0,2}, \ldots, x_{0,k}\} \\
x_i = \{x_{i,0}, x_{i,1}, x_{i,2}, \ldots, x_{i,(k-1)}\}, & 1 \leq i \leq k-1. \\
x_k = x_{k,0}
\end{cases}
\]

Then, the stationary probability vector \(x\) satisfies the following equation set

\[
\begin{align*}
x (P - I) &= 0 \\
x e &= 1
\end{align*}
\]

where \(I\) is an identity matrix of appropriate finite dimension. We adopt the method of Gaussian elimination with block state reduction [23] to solve \(x\). We first reformat transition probability matrix \(P\) by removing \(n\) rows from the bottom as follows

\[
P_{k-n} = \begin{bmatrix}
A_0 & B_0 \\
C_1 & A_1 & B_1 \\
C_2 & A_2 & B_2 \\
& & \ddots \\
C_{k-n} & F_{k-n}
\end{bmatrix}, & 0 \leq n \leq k - 1
\]

where \(F_k = A_k\) and

\[
F_{k-n} = A_{k-n} + B_{k-n} (I - A_{k-n+1})^{-1} C_{k-n+1}, & 1 \leq n \leq k.
\]
By partitioning $P_{k-n}$ into two blocks $\{0\}$ and $\{1, 2, ..., k-n\}$, the matrix $P$ can be decomposed into four submatrices as

$$P_{k-n} = \begin{bmatrix} P_{00} & P_{0A} \\ P_{A0} & P_{AA} \end{bmatrix}$$

where $P_{AA}$ is the submatrix of $P$ in the state space $\{1, 2, ..., k-n\}$. We similarly represent $x$ as $(x_0, x_A)$ with $x_A = [x_1, x_2, ..., x_{k-n}]$. From (12), we have

$$\begin{align*}
  x_0 &= x_A P_{A0} (I - P_{00})^{-1} \\
  x_k &= x_A P_{k-n}
\end{align*}$$

(16)

where $x_0$ can be derived in a function of $\{x_1, ..., x_{k-n}\}$. $P_{k-n}^1$ is the transition matrix of the censored Markov chain on the state space $\{1, ..., k-n\}$, and can be determined by $P_{k-n}^1 = P_{AA} + P_{A0} (I - P_{00})^{-1} P_{0A}$. Also, the stationary vector of $P_{k-n}^1$ is proportional to $x_A$, i.e., $x_A P_{k-n}^1 = x_A$. Following the similar procedure, we can further partition $P_{AA}$ in state space $\{1, ..., k-n\}$ into subsets $\{1\}$ and $\{2, ..., k-n\}$, and obtain an expression for $x_1$ in function of $\{x_2, ..., x_{k-n}\}$ with $P_{k-n}^2 = P_{AA} + P_{A1} (I - P_{11})^{-1} P_{1A}$. In summary, from (14) and (16), the overall steady-state distribution can be calculated recursively as

$$\begin{align*}
  x_0 &= x_0 F_0 \\
  x_{k-n} &= x_{k-n-1} B_{k-n-1} (I - F_{k-n})^{-1}, \quad 0 \leq n \leq k - 1
\end{align*}$$

(17)

with $\sum_{n=0}^{k} x_n = 1$.

C. Performance Measures

In this subsection, we evaluate system performance in terms of queue length, user waiting time, blocking probability and system busy cycle.

Queue length: Define $y_n, n = \{0, 1, ..., k\}$, as the marginal probability density of finding $n$ users in the system. We have $y_n = \{x_{i,j}: i + j = n\} e$, and $e$ is a column vector of ones with an appropriate dimension. Then, the average number of users in the system is

$$\bar{L} = \sum_{n=0}^{k} n y_n.$$  

(18)

The probability that there are $i$ HP users in the system is

$$L_i = \sum_{j=0}^{k-i} x_{i,j} e, \quad 0 \leq i \leq k.$$  

(19)

When considering the probability of $j$ LP users in the system, we need to discuss two scenarios separately. $i)$ If there is no LP user in the system, the situation becomes either there is no HP user in the system or there exist HP users with one LP user in service. $ii)$ If the number of the LP users belongs to $\{1, 2, ..., k\}$, the situation further involves three scenarios: $a)$ there is no HP user in the system and the server is on vacation, $b)$ an HP user is in service, and $c)$ an HP user arrives but the service to one LP user is incomplete. Thus, the probability that there are $j$ LP users in the system can be derived as

$$L_2 = \begin{cases} 
  x_{0,0} e + \sum_{i=1}^{k} x_{i,0} e_1, & j = 0 \\
  x_{0,j} e + \sum_{i=1}^{k-j} x_{i,j} e_1 + \sum_{i=1}^{k} x_{i,(j-1)} e_2, & 1 \leq j \leq k
\end{cases}$$

(20)

where $e_1 = (e_0 e)^t$ and $e_2 = (e_0 e)^t$ indicate one HP or LP user in service, respectively. $e_0$ is a column vector of all zeros. From (19) and (20), the average numbers of the HP and LP users in the system are $L_1 = \sum_{i=0}^{k} L_1$ and $L_2 = \sum_{j=0}^{k} L_2$, respectively.

User waiting time: Define $W$ as the users’ average waiting time in the system. Then, we have $W = \bar{L} / \lambda$ by applying Little’s law. Similarly, the average waiting time of the HP and LP users can be calculated as $W_1 = L_1 / \lambda_1$ and $W_2 = L_2 / \lambda_2$, respectively.

Blocking probability: A newly arrived HP user can be blocked if $i)$ the system buffer is full with $k$ users (no matter what types of users) and the current service is in phase $S_1$ or $S_2$, or $ii)$ the system is in service and $S_2$. Thus, the blocking probability of the HP user can be calculated as

$$P_{B_1} = \frac{1}{\lambda_1} \begin{cases} 
  x_{i,j} [D_{21} \otimes S_{12}], & i + j = k \\
  x_{i,j} [D_{21} \otimes V], & i = 0, j = k
\end{cases}$$

(21)

When an LP user arrives, the blocking event happens in three cases. $i)$ If only one position in the buffer is available, i.e., the queue length is $i + j = k - 1$, and there is an HP user arriving at the same time, the HP user has a priority to occupy the only position, $ii)$ The buffer is full, $i + j = k$, and the system is in service, and $iii)$ There are $k$ LP users in the system, $i = 0, j = k$, and the server is on vacation. Thus, the blocking probability of the LP user can be calculated as

$$P_{B_2} = \frac{1}{\lambda_2} \begin{cases} 
  x_{i,j} [D_{11} \otimes S_{12}] + D_{12} \otimes S_{12} + D_{1,1} \otimes (s_0 \beta_1)_{12}, & i + j = k \\
  x_{i,j} [D_{12} \otimes V], & i = 0, j = k
\end{cases}$$

(22)

where $S_{12} = S_1 + S_2$, $D_{12} = D_{0,1} + D_{1,1}$, and $(s_0 \beta_1)_{12} = s_0 \beta_1 + s_0 \beta_2$.

System busy cycle: Since the system consists of alternative vacation and service periods, it is essential to analyze such cyclic behavior for the purpose of system performance optimization. Let $t_v$ and $t_s$ denote the lengths of vacation and service periods, respectively. The system can leave the vacation mode when $i)$ at least one LP user waiting in the buffer at the end of the current vacation, or $ii)$ an HP user arrival interrupts the vacation. The service period, $t_v$, starts immediately after the system leaves the vacation mode, and ends when the buffer becomes empty.

Definition 1 (system busy cycle). A system busy cycle starts from the beginning of a vacation period when the buffer is empty, and ends when the next service period completes. Let $t_b$ denote the length of the system busy cycle. It can be formulated as

$$t_b = t_v + t_s.$$  

(23)

To derive the mean length of system busy cycle, we first calculate the mean vacation length. Let $v(t)$ denote the probability that the length of overall vacation period experiences $t$ slots, and there are user arrivals in the last $t'$ slots. Note that $v(t)$ is a joint probability of two scenarios: $i)$ there is no user arrival and the buffer is empty in the first $t - t'$ slots, and $ii)$
there are LP user arrivals during last \( t' \) slots. Note that the arrival of an HP user is not considered since the vacation will be interrupted with the HP user arrivals. Thus, based on the distribution of queue length, \( v(t) \) can be formulated as

\[
v(t) = \sum_{t'=0}^{t} \left( x_{0,0} e \right)^{t-t'} \left( \sum_{j=1}^{k} x_{0,j} e \right)^{t'}.
\]  \( \text{(24)} \)

For the mean service period, let \( G(l, t) \) denote the probability in phase type that \( l \) users complete service within \( t \) slots. To calculate \( G(l, t) \), three cases have to be discussed.

- **Case I:** \( l = 1, t \geq 1 \). If one user of any type completes service in more than one slot, it means such service is not completed in the first \( t - 1 \) slots and it is finished in the last slot. Thus, we have

\[
G^1(l, t) = S_{1}^{t-1} (s_1 \beta_1) + S_2^{t-1} (s_2 \beta_2).
\]  \( \text{(25)} \)

- **Case II:** \( l = t, t \geq 1 \). If the server uses \( t \) slots to serve \( t \) users, apparently, each type of user completes its service in one slot. Thus, by considering \( l_1 \) HP users and \( l_2 \) LP users in the system, we have

\[
G^2(l, t) = (s_1 \beta_1)^{l_1} + (s_2 \beta_2)^{l_2}.
\]  \( \text{(26)} \)

- **Case III:** \( l \geq 2, t \geq l \). If the server uses \( t \) slots to serve \( l \) users, in this case, any slot can be considered in the situation of either service completion or incomplete service. For example, in slot \( t - 1 \), the server has to be in two situations: either the service of user \( l - 1 \) is completed or service of user \( l \) is incomplete since only one slot left. Thus, \( G(l, t) \) in this case can be retrieved recursively, and at the end, it turns to either Case I or Case II. Thus, for Case III, we have

\[
G^3(l, t) = (s_1 \beta)^{l_2} G(l-1, t-1) + S_{12} G(l, t-1).
\]  \( \text{(27)} \)

In summary, by jointly considering the preceding three cases, the probability that \( l \) users complete service in \( t \) slots can be calculated as

\[
G(l, t) = \begin{cases} 
S_1^{t-1} (s_1 \beta_1) + S_2^{t-1} (s_2 \beta_2) & l = 1, t \geq 1 \\
(s_1 \beta_1)^{l_1} + (s_2 \beta_2)^{l_2} & l = t, t \geq 1 \\
(s_1 \beta)^{l_2} G(l-1, t-1) + S_{12} G(l, t-1) & l \geq 2, t > l 
\end{cases}
\]  \( \text{(28)} \)

Let \( \mathcal{H}(l, t) \) denote the probability in phase type that a service period lasts \( t \) slots given there are \( l \) users at the beginning of the service period. The derivation of \( \mathcal{H}(l, t) \) needs to consider the following two cases.

- **Case I:** There is no user arrival during the service period. This case is simple because only the initial users at the beginning of service need to be considered. Such procedure can be described as \( \mathcal{H}(l, t) = G(l, t) \otimes D_0^l \), where \( D_0^l \) means there is no arrival in \( t \) slots.

Case II: There exist new arrivals within the service period. In this case, assume that at time \( t' \), the server finishes the service to the initiated \( l \) users and there are \( t' \) new arrivals in \( t' \). Such procedure can be described as

\[
\mathcal{G}(l, t') \otimes \sum_{l'=1}^{2t'} d(l', t'), \text{ where } d(l', t') \text{ indicates the probability that there are } l' \text{ user arrivals during } t' \text{ slots, and can be calculated by}
\]

\[
d(l', t') = \sum_{i=1}^{t'} \left( l' \left( l' - i \right) \left( \frac{D_{0,1} - D_{2,1}}{D_{1,1}} \right) \right) \mathcal{H}(l', t' - t')
\]  \( \text{(29)} \)

Note that the maximum number of user arrivals is \( 2t' \), since in one slot, there are at most two user arrivals with one for each type. From \( t' \), we can assume a beginning of a new service period, which has \( l' \) users at the initial state. We can repeat the similar procedure till the end of the service period. Therefore, \( \mathcal{H}'(t) \) can be calculated as

\[
\mathcal{H}'(t) = \mathcal{G}(l, t) \otimes D_0^t
\]

\[
+ \sum_{l'=1}^{t-1} \left( \mathcal{G}(l, t') \otimes \sum_{t'=1}^{2t'} d(l', t') \right) \mathcal{H}(l', t' - t')
\]  \( \text{(30)} \)

Let \( h(l, t) \) denote the probability that a service period initiated by \( l \) users lasts \( t \) slots. Then, we have

\[
h(l, t) = \begin{cases} 
\psi \mathcal{H}(l, t) e, & l \leq \min(t, k) \\
0, & l > t
\end{cases}
\]  \( \text{(31)} \)

where \( \psi \) is the stationary probability vector that satisfying

\[
\sum_{l=1}^{\infty} \psi(l) = 1.
\]

From (24) and (31), the mean values of service and vacation periods can be calculated as

\[
\bar{t}_s = \sum_{t=1}^{\infty} t h(l, t), \text{ and } \bar{t}_v = \sum_{t=0}^{\infty} t v(t), \text{ respectively. Then, from (23), the mean system busy cycle equals } \bar{t}_b = \bar{t}_v + \bar{t}_s.
\]

IV. **Adaptive Service Rate and Vacation Length at the HeNB**

In this section, we first explain the tradeoff between system parameters and performance. Then, we propose the ASV method to maximize the HeNB energy efficiency and further introduce the one-step look-ahead method to reduce the computational complexity.

A. **Relationship on System Parameters and Performance**

Based on the previously queueing analysis, the HeNB operates in consecutive busy cycles with alternative service and vacation periods. Therefore, the service rate and the vacation length are two key factors determining system performance in terms of energy efficiency and QoS. We summarize the relationship among vacation length, service rate, system energy efficiency and QoS requirements as follows.

- **Vacation length v.s. Energy efficiency:** In order to achieve enhanced energy efficiency, in any busy cycle, it is essential to have a long and uninterrupted vacation instead of multiple short vacations, since a long vacation can save energy consumption in the listening periods.

- **Vacation length v.s. QoS:** A long vacation may result in users’ waiting time over maximum delay requirement or exceeding the blocking probability threshold. A long vacation also has a high chance to be interrupted by the
arrival of the HP users. In addition, changing vacation length may affect the initial state, i.e., the number of users in the buffer, of the next service period so as to cause a heavier burden to the following service period.

- **Service rate v.s. Vacation and Energy efficiency**: When the HeNB enters a service period, it may adopt a high service rate to quickly clear the buffer so that the HeNB take a long vacation. However, choosing high service rate consumes more energy, which may degrade the energy efficiency.

- **Service rate v.s Vacation and QoS**: In the condition that the HeNB is in the service period and the HP user is expected to arrive soon, it is wise to adopt a low service rate so that the service of the incoming HP user can be completed before the initiation of a vacation to avoid a quick vacation interruption. However, reducing service rate may increase users’ waiting time and blocking rate.

### B. Maximizing System Energy Efficiency

To balance the tradeoff on system performance, in this section, our objective is to maximize the system energy efficiency and satisfy QoS requirements at the same time by adjusting the service rate and vacation length. Since the system behavior is running with alternative vacation and service periods, we can define the system energy efficiency, denoted by $E_{\text{sys}}$, as

$$E_{\text{sys}} = \sum_{n=1}^{n_b} E^b_n(r_n, v_n)$$  \hspace{1cm} (32)

where $n_b$ denotes the number of system busy cycles. $E^b_n(r_n, v_n)$ denotes system energy efficiency for each busy cycle $n$, which is the function of selected service rate $r_n$ and vacation length $v_n$. In any busy cycle, $E^b(r, v)$ can be formulated as a logarithmic function of system throughput over the total energy consumption on both service and vacation periods. Specifically,

$$E^b(r, v) = \log \left( \frac{\sum_{i=1}^{l_1+l_2} r_i t_i}{\sum_{i=1}^{l_1+l_2} p_i + \sum_{\alpha=1}^{N^b_v} (v_\alpha p_v + p_l)} \right)$$  \hspace{1cm} (33)

where $r_i$, $p_i$, $t_i$ denote the service rate, the corresponding energy consumption and the number of slots in service for user $i$, respectively. $N^b_v$ denotes the total number of consecutive vacations in a busy cycle. $v_\alpha$ denotes the vacation length in the number of slots. Note that although there is no system throughput when the server is on vacation, the system still consumes some power in vacation and listening periods, which are denoted as $p_v$ and $p_l$, respectively. Commonly, $p_l \gg p_i \gg p_v \geq 0$. Here, we adopt a logarithmic function for energy efficiency to avoid potential marginal solutions.

Then, the problem of maximizing system energy efficiency can be formulated as

$$\max_{\{r\}, \{v\}} \quad E_{\text{sys}} = \sum_{n=1}^{n_b} E^b_n(r_n, v_n)$$  \hspace{1cm} (34)

s.t. \hspace{0.5cm} \begin{align*}
\mathcal{P}_{B_1} &\leq \eta_1, \mathcal{P}_{B_2} \leq \eta_2 \\
\overline{W_1} &\leq \omega_1, \overline{W_2} \leq \omega_2
\end{align*} \hspace{1cm} (35) \hspace{1cm} (36)

where constraint (35) guarantees that blocking probability of the HP (LP) user is not greater than a predefined threshold $\eta_1 (\eta_2)$. Constraint (36) guarantees that the waiting time of the HP (LP) users is less than a predefined maximum delay $\omega_1 (\omega_2)$. Constraint (37) limits the number of slots in service must be greater than the number of users served since at least one slot is needed to serve a user.

### C. The ASV method with Dual Decomposition Solution

We try to solve the problem of maximization defined in (34) ~ (38) by achieving the maximum HeNB energy efficiency in each busy cycle. Thus, we rewrite the problem (34) in each busy cycle as

$$\mathbf{P}^* : \max_{\{r\}, \{v\}} \log \left( \sum_{i=1}^{l_1+l_2} r_i t_i \right) - \log \left( \sum_{i=1}^{l_1+l_2} p_i + \sum_{\alpha=1}^{N^b_v} (v_\alpha p_v + p_l) \right)$$  \hspace{1cm} (39)

s.t. \hspace{0.5cm} (35) ~ (38).

Next, we adopt the Lagrangian dual decomposition method to relax the coupled constraints by introducing a Lagrange multiplier $\xi$. The dual problem is:

$$\mathbf{D}^* : \min_{\xi} \quad f_r(\xi) + g_v(\xi)$$  \hspace{1cm} (40)

where

$$f_r(\xi) = \begin{cases} \max_{\{r\}} & \log \left( \sum_{i=1}^{l_1+l_2} r_i t_i \right) - \xi \\ \text{s.t.} & \mathcal{P}_{B_1} \leq \eta_1, \overline{W_1} \leq \omega_1 \\
& \sum_{i=1}^{l_1+l_2} t_i \geq l_1 + l_2, l_1 + l_2 = 0, 1, \ldots, k \\
& \{r\} \in \mathbb{R} \end{cases}$$  \hspace{1cm} (41)

$$g_v(\xi) = \begin{cases} \max_{\{v\}} & \xi - \log \left( \sum_{i=1}^{l_1+l_2} p_i + \sum_{\alpha=1}^{N^b_v} (v_\alpha p_v + p_l) \right) \\ \text{s.t.} & \mathcal{P}_{B_2} \leq \eta_2, \overline{W_2} \leq \omega_2 \\
& l_1 + l_2 \leq k \\
& \{v\} \in \mathbb{V} \end{cases}$$  \hspace{1cm} (42)

When the constraints in the problem (39) are all linear equalities and inequalities, based on *Slater’s condition* (or *constraint qualification*) [24], the problem (39) holds strong duality. Thus, the optimal value of problem (39), denoted by $P^*$, and the optimal value of Lagrange dual problem (40), denoted by $D^*$, must have same optimal value, which means the optimal duality gap is zero, i.e., $D^* = P^*$. Since the constraints (35)~(36) are non-linear inequalities, the problem (39) holds weak duality based on *Slater’s condition* (or *constraint qualification*) [24]. Thus, the optimal value of problem (39), denoted by $P^*$, and the optimal value of Lagrange dual problem (40), denoted by $D^*$, hold $D^* \leq P^*$. Since the dual
problem (40) is always convex, we can always find the best lower bound on the primal problem (39). Therefore, given the dual optimal $\xi^*$, we can at least obtain the near optimal solution by solving the decoupled optimization problems (41) and (42) separately without coordination among the service and vacation periods.

To solve the dual maximization problems, we propose a solution consisting of inner and outer optimization, which captures the special structure in system behavior, i.e., vacation periods followed by a service period in each busy cycle. The adaptive service rate from (41) is derived in the inner optimization, while the adaptive vacation length from (42) can be obtained in the outer optimization.

The basic idea of the proposed ASV method can be described by the following three steps.

- **Step 1.** In inner optimization, by considering all possible initial states, the local optimal sets of service rates $r^*$, denoted by $\hat{R}$, can be derived by adopting dynamic programming as detailed in Section IV-C1.

- **Step 2.** In outer optimization (depicted in Section IV-C2), when a vacation length is selected, the queue length at the time when the vacation ends and the service starts can be estimated based on the queueing analysis in section III. Then, by applying $\hat{R}$ from step 1, the optimal vacation length $v^*$, denoted by $V^*$, can be obtained and the corresponding $r^*$ can be selected according to the related initial states. If there are multiple global optimums, the system will choose any one of them.

- **Step 3.** By applying step 1 and step 2 in each busy cycle, the system is expected to achieve maximum energy efficiency in a long run.

We discuss the detailed solution procedures in the following subsections.

1) **Inner Optimization**: Define $t'_{n'} \in \mathbb{T}$, $n = 0, 1, ..., T - 1$, as the time epochs that a service starts for an HP (LP) user, where $\mathbb{T}$ represents the total HeNB operational time. At each time epoch $t'_{n'}$, the HeNB makes an action to select a service rate $r_i \in \mathbb{R}$. Note that the slots required in service $t_i$ and the power consumption $p_i$ are the functions of $r_i$. Define a utility function $u(r_i)$ of energy efficiency for serving user $i$ as

$$u(r_i) = \frac{r_i t_i}{p_i}. \tag{43}$$

Then, at time epoch $t'_{n'}$, the HeNB determines $r_i^*$ as

$$r_i^* = \arg \max_{r_i} U(t'_{n'}) \tag{44}$$

where

$$U(t'_{n'}) := \max_{r_i \in \hat{R}, r'_i \in \mathbb{T}} u_{n'}(r_i). \tag{45}$$

Obviously, the optimization of $r_i$ is a multistage decision making problem, which can be solved through the following dynamic programming approach.

Define a decision sequence $\Lambda = \{a_n\}, n = \{0, 1, ..., T - 1\}$, representing actions of selecting appropriate service rate $r_i$. If this action space consists of a feasible solution to problem $U(t'_{n'})$, $\Lambda$ will be deemed feasible at state $t'_{n'} \in \mathbb{T}$. Similarly, if $\Lambda$ constitutes an optimal solution to problem $U(t'_{n'})$, such decision sequence will be deemed optimal with respect to state $t'_{n'}$.

**Theorem 1.** At each time epoch $t'_{n'} \in \mathbb{T}$, $U(t'_{n'})$ has at least one optimal solution.

**Proof:** The proposition clearly holds for each $t'_{n'}$, since all admissible service rate $\mathbb{R}$ is compact, i.e., finite and non-empty. Based on Weierstrass extreme value theorem [25], for each $t'_{n'}$, $r_i \in \mathbb{R}$ is in a closed and bounded interval. Thus, the continuous function of $r_i \in \mathbb{R}$ must attain the maximum and minimum values, each at least once. Thus, $U(t'_{n'})$ has at least one optimal solution. \hfill \square

**Theorem 2.** Given policy $\pi(t'_{n'}, a_n)$ for any pair of time epoch and action $(t'_{n'}, a_n)$, $t'_{n'} \in \mathbb{T}$ and $a_n \in \Lambda$, there exists a policy $\pi^*$ that is optimal for $U(t'_{n'})$.

**Proof:** Let $\Lambda^* = \{a_0^*, a_1^*, ..., a_{T-1}^*\}$ be any optimal solutions to $U(t'_{n'})$. Define a policy $\pi^*(t'_{n'}, a_n) \in \Lambda^*, t'_{n'} \in \mathbb{T}$. Note that applying this policy can always generate the sequence of decisions $(a_0^*, a_1^*, ..., a_{T-1}^*)$ at time epochs 0, 1, ..., $T-1$. Since $(a_0^*, a_1^*, ..., a_{T-1}^*)$ is an optimal solution to problem $U(t'_{n'})$, it follows that $\pi^*$ is an optimal policy for the problem $U(t'_{n'})$. \hfill \square

Since the action depends only on current state without relevance to the prior history, $\pi(t'_{n+1}, a_{n+1})$ is only determined by $\pi(t'_{n'}, a_n)$, which satisfies Markovian properties. For any policy $\pi$, define an evaluation function $\varphi^\pi_n(t'_{n'})$ for stage $n = \{0, 1, ..., T\}$, which evaluates the system energy efficiency when there are $l'$ users in the system. Particularly, at final stage $T$, the service period is completed and there are no users in the system such that $\varphi^\pi_T(l'_T) = 0$. Thus, the evaluation function at any previous stage can be computed via backward induction by using the Bellman equation as

$$\varphi^\pi_n(t'_{n'}) = \max_{\pi(t'_{n'}, a_n)} \left\{ u_n(r_n) + \gamma \cdot \varphi^\pi_{n+1}(l'_{n+1}) \right\} \tag{46}$$

s.t. $l_{n+1}' = l'_{n} + d'_n - 1 > 0, n \in [0, T - 1]$ \tag{35} \sim \tag{36}

where $d'_n$ denotes the number of user arrivals at stage $n$, and $\gamma$, $0 < \gamma \leq 1$, denotes the discount factor. $\gamma = 1$ represents the undiscounted case.

Let $\Pi^* = \{\pi^*(t'_{n'}, a_n)\}$ denote optimal policies. Once maximum $\varphi^\pi_n(l'_n)$ has been found, an optimal action $a^*_n$ for stage $n$ can be determined with the underlying optimal policy $\pi^*(t'_{n'}, a_n)$ as

$$\pi^*(t'_{n'}, a_n^*) = \arg \max_{\pi(t'_{n'}, a_n)} \varphi^\pi_n(t'_{n'}). \tag{47}$$

**Proposition 1.** The computational complexity of the inner optimization is $O(kNT)$.

**Proof:** The initial state of any service period consists of two scenarios: i) Due to vacation interruption, the system has one HP user and $l' - 1$ LP users, and ii) When vacation ends without interruption, the system has $l'$ LP users. By applying exhaustive search, maximum $2 \cdot k$ rounds are needed for traversing all conditions since $1 \leq l' \leq k$. The computation should also consider all the admissible service rates $N$ and all
states $T$, i.e., the number of all served users, which results in the total computational complexity equal to $O(kN^2T)$.  

2) Outer Optimization: In outer optimization, to derive the optimal vacation length, we adopt an online reinforcement learning algorithm, called Q-learning algorithm [26]. Since there may be multiple vacations in each busy cycle, we define time epochs at the beginning of each vacation as $t'_\epsilon$, $\epsilon = 1, 2, ..., N_b^\epsilon$ with $N_b^\epsilon < \infty$. Let $\zeta_\epsilon(v_\epsilon)$, $\epsilon = 1, 2, ..., N_b^\epsilon$, denote all admissible values of vacation length at each stage $\epsilon$. Recall that $\{v_\epsilon\} \in \mathbb{V}$, $i = 1, 2, ..., M$.

Assume that the system has an approximated Q-value denoted by $Q_\epsilon(t'_\epsilon, v_\epsilon)$. It means the system selects vacation length $v_\epsilon \in \mathbb{V}$ at time epoch $t'_\epsilon$ which generates a value $Q_\epsilon(t'_\epsilon, v_\epsilon)$. Obviously, if an optimal Q-value is achievable, the optimal vacation length can also be determined. At initial stage $\epsilon = 1$, the vacation length can be selected by adopting random walk in the admissible set $\mathbb{V}$, i.e., $v_\epsilon \in \mathbb{V}$. Then, the selected vacation length $v_\epsilon$ can trigger the system to generate cost function $c$, which is formulated as

$$c_\epsilon^* (v_\epsilon) = \begin{cases} E_\epsilon^\beta (r_\epsilon (l_{\epsilon_1}^\epsilon), v_\epsilon), & l_{\epsilon_1}^\epsilon \in [1, k] \quad \text{if } l_{\epsilon_1}^\epsilon = 0, \\ 0, & \end{cases}$$

(48)

where $E_\epsilon^\beta$ refers to the energy efficiency defined in (34). $r_\epsilon \in \mathbb{R}$ is the optimal set of service rates derived from the inner optimization with corresponding $l_{\epsilon_1}^\epsilon$, which denotes the initial number of users at the beginning of next service period. Particularly, the system starts a new vacation when $l_{\epsilon_1}^\epsilon = 0$ with a zero cost.

Then, for multiple vacation periods in one busy cycle, the optimization problem can be formulated as

$$\max_{v_\epsilon \in \mathbb{V}} \sum_{\epsilon = 1}^{N_b^\epsilon} c_\epsilon^* (v_\epsilon)$$

(49)

s.t. $P_{e_2} \leq \eta_2, \overline{W}_2 \leq \omega_2$.

According to Q-learning algorithm, let $\alpha_\epsilon (t'_\epsilon, v_\epsilon), 0 < \alpha_\epsilon \leq 1$, denote the learning rate, which can update Q-value at each iteration based on the system requirements. When the system state changes to $t_{\epsilon+1}$, then $Q_{\epsilon+1}$ can be calculated as

$$Q_{\epsilon+1} (t'_{\epsilon+1}, v_\epsilon) = (1 - \alpha_\epsilon + \beta_\epsilon) Q_\epsilon (t'_\epsilon, v_\epsilon) + (\alpha_\epsilon \beta_\epsilon) [c_\epsilon^* (v_\epsilon) + \gamma \Theta_\epsilon (t'_{\epsilon+1}, v_{\epsilon+1})]$$

(50)

where $\beta_\epsilon \in [0, 1)$ denotes the rate of penalty due to vacation interruption. $\beta_\epsilon = 0$ if vacation is not interrupted. Note that $0 \leq \beta_\epsilon < \alpha_\epsilon < 1$. $\Theta_\epsilon (t'_{\epsilon+1} + 1, v_{\epsilon+1})$ indicates the estimated optimal Q-value in the next state $t_{\epsilon+1}$ and can be calculated as

$$\Theta_\epsilon (t'_{\epsilon+1}, v_{\epsilon+1}) = \max_{v_{\epsilon+1}} Q_\epsilon (t'_{\epsilon+1}, v_{\epsilon+1})$$

(51)

$\gamma \in (0, 1]$ is a discount factor as described in the previous section. The learning rate $\alpha_\epsilon$ determines the degree how current value obtained can override the old value. When $\alpha_\epsilon = 0$, the system will stop learning, while $\alpha_\epsilon \to 1$ enables the system to only accept the up-to-date value. Note that a constant learning rate is often adopted in practice.

Let $c_\epsilon^*$ and $v_\epsilon^*$ denote the desired optimal values. We have

$$c_\epsilon^* = \max_{v_\epsilon} Q_\epsilon (t'_\epsilon, v_\epsilon)$$

(52)

$v_\epsilon^* = \arg \max_{v_\epsilon} Q_\epsilon (t'_\epsilon, v_\epsilon)$.

(53)

Before the learning process starts, it returns an arbitrary fixed Q-value. This algorithm repeats at the beginning of each vacation period and ends when $t'_{\epsilon+1}$ is a final state. For all final states, $\Theta_\epsilon (t'_{\epsilon+1}, v_{\epsilon+1})$ stops updating and thus retains its initial value. Without loss of generality, we adopt $\Theta_\epsilon (t'_{\epsilon+1}, v_{\epsilon+1}) = 0$.

Proposition 2. The computational complexity of the outer optimization is $O(MN_b^\epsilon)$.

Proof: The outer optimization terminates after at most

$$\sum_{\epsilon=1}^{N_b^\epsilon} \sum_{v=1}^{M} Q_\epsilon (t'_\epsilon, v_\epsilon)$$

steps, which results in the computational complexity equal to $O(MN_b^\epsilon)$.

D. One-step Look-ahead Method

In the outer optimization, since the total number of vacations in a busy cycle, $N_b^\epsilon$, is a small integer, the computational complexity mostly depends on the number of admissible vacation lengths $M$, which can be determined by the mean vacation length based on the previous queueing analysis. However, in inner optimization, the underlying idea is to use backward recursion to approximate the optimality of $r_\epsilon$. Therefore, if $l'_\epsilon$ is large, which means in a service period, a lot of users are waiting for service, or if $l \to \infty$, backward recursion may involve considerable computational complexity. Thus, the inner optimization dominates the complexity of the overall solution procedure.

In order to reduce the computational complexity of the inner optimization, we propose an effective method, called look-ahead policy. In this policy, the system makes a decision, denoted by $\pi^\epsilon (l'_\epsilon, a_n)$, at the beginning of service period. The one-step look-ahead policy is

$$\varphi_n^\epsilon (l'_\epsilon) = \max_{n'} \left\{ u_{n+1} + N_b^\epsilon \right\}$$

(54)

where $N_b^\epsilon (l'_n, a_n) = \gamma^\epsilon \cdot \varphi_n^\epsilon (l'_n)$ is the approximated value function of the next stage. Define $u_n = \Phi_n$. Then, we have $u_{T+1} = \Phi_{T+1}$ at the final stage. Thus, by tracing back only one stage, the system can obtain a close to the maximum value based on (46) under identical system constraints.

When policy $\pi^\epsilon$ achieves the maximum value, $\Phi_{n+1}$ can be obtained on the basis of the one-step look-ahead approximation with respect to $n_{T+2}$. In other words, for all possible states $l'_n$, if using $\Phi_{n+1} (l'_n)$ instead of $\Phi_{n+1} (l'_n)$, we obtain

$$\Phi_{n+1} (l'_n) = \max_{n'} \left\{ u_{n'+1} + N_b^\epsilon \right\}$$

(55)

where $N_b^\epsilon$ is the approximation of the value function $\Phi_{n+1}$. Intuitively, the results will be more accurate if adopting more steps ahead. It has less computational complexity due to its approximation and truncation rules. However, this adaptation can be transformed to near optimality if it executes $(T+1)$-step look-ahead policy. The computational complexity of one-step look-ahead method is $O(kN^2)$, which is determined only by the buffer size and the number of admissible service rates in the current stage. Since in practice, $N \ll T$, we have $O(kN^2) \ll O(kN^2T)$. 
V. Simulation Results

In the simulation, the HeNB is postulated to maintain two separate queues for the HP and LP users. Both queues follow the FCFS policy and share the same buffer size $k$. Following the discussion in [27], [28], the arrival rates of the HP and LP users are set as $\lambda_1 = 0.05$ users/slot and $\lambda_2 = 0.2$ users/slot, respectively. The HP (LP) user arrivals are independent regardless of the server in service or vacation period. The admissible service rates are set as $\mathcal{R} = \{0.2, 0.4, 0.6, 0.8\}$. A slot is normalized to $1 \text{s}$. The HeNB mode transient period between service and vacation is omitted due to its quite short time period, e.g., $70 \mu s$ [2]. Based on the observations from queueing analysis shown in Figs. 2 and 3, we set the maximum waiting times for the HP and LP users as $\omega_1 = 10 \text{s}$ and $\omega_2 = 20 \text{s}$, respectively, and the thresholds of blocking probabilities for the HP and LP users as $\eta_1 = 0.02$ and $\eta_2 = 0.06$, respectively. All other system parameters are compliant with 3GPP LTE HeNB setup [19]. For all simulation results, we evaluated the system after 1000 slots until the system is in steady state. All the simulations are completed in Matlab R2012b version 8.0.0.783.

Figs. 2 and 3 show the key characteristics of the analytical MAP/PH/1/k queue with multiple vacations and user priorities. Fig. 2 presents both simulation and analytical results on the cumulative distribution functions (CDF) of the waiting time for both HP and LP users. It is demonstrated that analytical results match the numerical results very well, which validates the accuracy of the proposed analytical queueing model. In addition, it is obvious that the HP user with the privilege of vacation interruption has lower waiting time compared to the LP user. Fig. 3 shows the effect of user priorities on blocking probabilities with a variation on vacation length and service rate. Clearly, long vacations can result in high blocking probability. Another observation is that the blocking rate of the HP users is quite lower compared to the LP users. It is because $i)$ when an HP user enters the queue, it always has priority to be served, and $ii)$ the HP users can interrupt the vacation periods without waiting in the queue. Note that when a large vacation length is applied, poor QoS performances such as user waiting time and blocking rate become inevitable even though high service rates are selected.

Fig. 4 compares the proposed ASV method with a traditional method, in which the HeNB adopts fixed service rate and vacation length. For comparison purpose, the service rates $\mu_1$ and $\mu_2$ are set as the same values, i.e., $\mu_1 = \mu_2$, so that we can highlight the effects of dynamic vacation length on system performance. From the figure, we can observe that for a buffer size $k = 20$, the proposed ASV method outperforms the traditional method in system energy efficiency by up to 20%. Such gain can even be enhanced for larger buffer sizes. In addition, as traffic intensity increases, we observe that the performance improvement due to adaptive vacation length reduces gradually. This is because the server becomes busier with fewer slots arranged for vacations. Such observation suggests that the system should increase buffer size or apply high service rate so as to attain the opportunity of long vacations. In addition, from Fig. 4, we can observe that the system energy efficiency degrades with the increase of buffer size because a larger buffer size may result in long and multiple consecutive vacations. Thus, the service period in one busy cycle may become much shorter than the vacation period. Since the vacation and listening periods also consume energy, long and multiple consecutive vacations may reduce energy efficiency due to increased total energy consumption in a busy cycle while keeping system throughput unchanged. In addition to the degradation of energy efficiency, long and multiple vacations may result in the growth of user waiting time. However, the proposed ASV method can make such degradation gradually compared to the traditional method. From an energy saving point of view, the improved energy efficiency implies the HeNB experiences more power off periods by suitably adjusting the system parameters. Thus, it can be expected that the proposed ASV method can be more aggressive in performance improvement when the network traffic is low such that the HeNB has more slots for vacations.

![Figure 2. The cumulative distribution functions (CDF) of waiting time (in slot) for the HP and LP users.](image-url)

![Figure 3. The blocking probabilities v.s. mean vacation length with buffer size $k = 10$.](image-url)
Fig. 4. Effects of traffic intensity $\rho$ on system energy efficiency subject to vacation length $v$ and buffer size $k$.

Fig. 5 compares the average number of busy cycles and vacations between the ASV and the traditional methods. It is obvious that the lengths of busy cycles and vacations change dynamically with the variation of buffer size in the ASV method, while in the traditional method, both performance measures almost remain the same regardless of what buffer size the system adopts. Therefore, the ASV method is more flexible and can better adapt to the dynamics of system parameters.

In Figs. 6 and 7, the average waiting time and blocking rate for both types of users are presented. The waiting times of the HP users are almost same in both the ASV and the traditional methods, which is close to 0, due to their higher priority in service and vacation interruption. However, for the LP users, the average waiting time tends to flatten in the traditional method, while, in the ASV method, the average waiting time may rise as the buffer size increases until it meets the delay threshold. In Fig. 7, it is obvious that the ASV method is better than the traditional method in terms of average blocking rates for both types of users when the HeNB buffer is small.

VI. CONCLUSIONS

In this paper, we investigated the operation of the HeNBs in femtocells with hybrid access and power on/off characteristics. We developed the MAP/PH/1/k queueing model with user priorities and vacations to better describe the system behaviors. Based on the formulated queueing model, key performance metrics in terms of users’ waiting time, blocking rate and system busy cycle were derived. Based on the analytical results, we proposed an adaptive service rate and vacation length (ASV) method in order to maximize the HeNBs’ energy efficiency and guarantee QoS provisioning. In addition, the
one-step look-ahead method was proposed to reduce the computational complexity. The simulation results demonstrated that the proposed ASV method significantly enhanced the HeNBs’ energy efficiency. The current work can be extended to multiple-server scenarios by considering the heterogeneous cellular networks with the cooperation of a macro-BS and multiple HeNBs.

**APPENDIX A**

**BLOCK SUBMATRICES OF TRANSITION MATRIX \( \mathcal{P} \)**

\[
B_0 = \begin{bmatrix}
B_{00} & B_{01} & B_{02} \\
B_{10} & B_{11} & B_{12} \\
B_{20} & B_{21} & B_{22}
\end{bmatrix}
\]

where \( B_0 \in \mathbb{D}^{(k+1) \times k}, B_{00} = \begin{bmatrix} D_{1,0} \otimes (v + \mathcal{Q}) \otimes \beta_1 & 0 \end{bmatrix}, B_{01} = \begin{bmatrix} D_{1,1} \otimes (v + \mathcal{Q}) \otimes \beta_1 & 0 \end{bmatrix}, B_{10} = \begin{bmatrix} D_{1,0} \otimes (s_2 \beta_1) & 0 \end{bmatrix}, B_{11} = \begin{bmatrix} D_{1,1} \otimes (s_2 \beta_1) & 0 \end{bmatrix}, B_{20} = \begin{bmatrix} D_{1,0} \otimes (s_1 \beta_2) & 0 \end{bmatrix}, B_{21} = \begin{bmatrix} D_{1,1} \otimes (s_1 \beta_2) & 0 \end{bmatrix}, B_{22} = \begin{bmatrix} 0 & 0 \end{bmatrix} \]

And \( A_k = \begin{bmatrix} D \otimes S_1 + D_{10} \otimes (s_1 \beta_1) & 0 \end{bmatrix} \).

\[
A_k = \begin{bmatrix}
B_{10} & B_{00} \\
B_{11} & B_{01} \\
B_{12} & B_{02}
\end{bmatrix}
\]

where \( B_i \in \mathbb{D}^{(k-i+1) \times (k-i-1)}, i = 1, 2, \ldots, k-2, B_{10} = \begin{bmatrix} D_{1,0} \otimes (s_1 \beta_1) & 0 \end{bmatrix}, B_{00} = \begin{bmatrix} D_{1,1} \otimes (s_1 \beta_1) & 0 \end{bmatrix}, B_{11} = \begin{bmatrix} D_{1,0} \otimes (s_1 \beta_1) & 0 \end{bmatrix}, B_{12} = \begin{bmatrix} D_{1,1} \otimes (s_1 \beta_1) & 0 \end{bmatrix}, B_{20} = \begin{bmatrix} D_{1,0} \otimes (s_2 \beta_1) & 0 \end{bmatrix}, B_{21} = \begin{bmatrix} D_{1,1} \otimes (s_2 \beta_1) & 0 \end{bmatrix}, B_{22} = \begin{bmatrix} 0 & 0 \end{bmatrix} \]

\[
C_i = \begin{bmatrix}
C_{10} & C_{00} \\
C_{11} & C_{01} \\
C_{12} & C_{02}
\end{bmatrix}
\]

where \( C_i \in \mathbb{D}^{(k-i+1) \times (k-i+1)}, i = 1, 2, \ldots, k-2, C_{10} = \begin{bmatrix} D_{0,0} \otimes (s_1 \beta_1) & 0 \end{bmatrix}, C_{00} = \begin{bmatrix} D_{0,1} \otimes (s_1 \beta_1) & 0 \end{bmatrix}, C_{11} = \begin{bmatrix} D_{0,0} \otimes (s_1 \beta_1) & 0 \end{bmatrix}, C_{01} = \begin{bmatrix} D_{0,1} \otimes (s_1 \beta_1) & 0 \end{bmatrix}, C_{12} = \begin{bmatrix} D_{0,0} \otimes (s_2 \beta_1) & 0 \end{bmatrix}, C_{02} = \begin{bmatrix} D_{0,1} \otimes (s_2 \beta_1) & 0 \end{bmatrix} \]

References


Jun Cai received the B.Sc. and M.Sc. degrees from Xi’an Jiaotong University, Xi’an, China, in 1996 and 1999, respectively, and the Ph.D. degree from the University of Waterloo, ON, Canada, in 2004, all in electrical engineering. From June 2004 to April 2006, he was with McMaster University, Hamilton, ON, as a Natural Sciences and Engineering Research Council of Canada Postdoctoral Fellow. Since July 2006, he has been with the Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, Canada, where he is currently an Associate Professor. His current research interests include energy-efficient and green communications, dynamic spectrum management and cognitive radio, radio resource management in wireless communications networks, and performance analysis. Dr. Cai served as the Technical Program Committee Co-Chair for the IEEE Vehicular Technology Conference 2012 Fall Wireless Applications and Services Track, the IEEE Global Communications Conference (Globecom) 2010 Wireless Communications Symposium, and International Wireless Communications and Mobile Computing (IWCMC) Conference 2008 General Symposium; the Publicity Co-Chair for IWCMC in 2010, 2011, 2013, and 2014; and the Registration Chair for the First International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine) in 2005. He also served on the editorial board of the Journal of Computer Systems, Networks, and Communications and as a Guest Editor of the special issue of the Association for Computing Machinery Mobile Mobile Networks and Applications. He received the Best Paper Award from Chinacom in 2013, the Rh Award for outstanding contributions to research in applied sciences in 2012 from the University of Manitoba, and the Outstanding Service Award from IEEE Globecom in 2010.

Xiaolong Li received the B.E. degree from Harbin Institute of Technology, Harbin, China in 2003 and the Ph.D. degree from Hunan University, Changsha, China in 2008. Since July 2008, he has been with Guilin University of Electronic Technology, Guilin China, where he is currently a Professor. In 2013, he won the best paper award at the ChinaCom conference. His research interests include sensor networks, M2M networks, and ad hoc networks.

Shiwei Huang received the B.E. and M.S. degrees in communication engineering from Guilin University of Electronic Technology, Guilin, China, in 2010 and 2012, respectively. He is currently working towards the Ph.D. degree in telecommunications with the Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, Canada. His current research interests include wireless relay networks, cooperative spectrum sensing, and energy-efficient communications.