Wireless Pers Commun (2010) 53:111–131
DOI 10.1007/s11277-009-9673-8

Performance Analysis of Spillover-Partitioning Call Admission Control in Mobile Wireless Networks

Okan Yilmaz · Ing-Ray Chen · Gregory Kulczycki · William B. Frakes

Published online: 6 March 2009

Abstract We propose and analyze spillover-partitioning call admission control (CAC) for servicing multiple service classes in mobile wireless networks for revenue optimization with quality of service (QoS) guarantees. We evaluate the performance of spillover-partitioning CAC in terms of execution time and optimal revenue obtainable by comparing it with existing CAC algorithms, including partitioning, threshold, and partitioning-threshold hybrid admission control algorithms. We also investigate fast spillover-partitioning CAC that applies a greedy heuristic search method to find a near optimal solution fast to effectively trade off solution quality for solution efficiency. The solution found by spillover-partitioning CAC is evaluated by an analytical model developed in the paper. We demonstrate through test cases that spillover-partitioning CAC outperforms existing CAC algorithms for revenue optimization with QoS guarantees in both solution quality and solution efficiency for serving multiple QoS service classes in wireless networks.

Keywords Call admission control · Quality of service · Performance analysis · Revenue optimization · Mobile networks

1 Introduction

Next generation mobile wireless networks will carry real-time multimedia services such as video and audio and non-real-time services such as images and files. An important goal is

O. Yilmaz (✉) · I.-R. Chen · G. Kulczycki · W. B. Frakes
Computer Science Department, Virginia Tech, Falls Church, VA, USA
e-mail: oyilmaz@vt.edu

I.-R. Chen
e-mail: irchen@vt.edu

G. Kulczycki
e-mail: gregwk@vt.edu

W. B. Frakes
e-mail: wfrakes@vt.edu

to adapt to user needs and growing population without compromising the Quality of Service (QoS) requirements.

Two important QoS metrics in cellular networks are the blocking probability of new calls and the dropping probability of handoff calls due to unavailability of wireless channels. Mobile users in a cellular network establish a connection through their local base stations. A base station may support only a limited number of connections (channels assigned) simultaneously due to limited bandwidth. A handoff occurs when a mobile user with an ongoing connection leaves the current cell and enters another cell. An ongoing connection may be dropped during a handoff if there is insufficient bandwidth in the new cell to support it. We can reduce the handoff call drop probability by rejecting new connection requests. Reducing the handoff call drop probability could result in an increase in the new call blocking probability. As a result, there is a tradeoff between handoff and new call blocking probabilities.

Call admission control (CAC) for single-class network traffic, such as voice has been studied extensively in the literature [1–16,23]. The Guard channel algorithm [1] assigns a higher priority to handoff calls so that more channels are reserved for handoff requests. Hong and Rappaport [2] proposed a cutoff threshold algorithm with no distinction made initially between new and handoff calls which are treated equally on a FCFS basis for channel allocation until a predetermined channel threshold is reached. When the threshold is reached, new calls are blocked (cutoff), allowing only handoff calls. They subsequently [3] proposed a priority oriented algorithm where queuing of handoff calls is allowed. Guerin [4] demonstrated that queuing of both new and handoff calls improves channel utilization while reducing call blocking probabilities. Yang and Ephremides [5] proved that FCFS based sharing [6] can generate optimal solutions for some systems. Most of the above mentioned research methods focus on voice-based cellular systems.

Fang [7] presented a thinning algorithm which supports multiple types of services by calculating the call admission probability based on the priority and the current traffic situation. When the network approaches congestion, call admissions are throttled based on the priority levels of calls, i.e., lower priority calls are blocked, and hence thinned to allow higher priority calls. Wang et al. [8] classified traffic as real-time and non-real-time traffic and divided the channels in each cell into three parts, namely, new and handoff real-time calls, new and handoff non-real-time calls, and real-time and non-real-time handoff calls. For overflow of real-time and non-real-time service handoff requests from the first two groups of channels, real-time handoff service requests are given a higher priority than non-real-time service requests. Lai et al. [9] compared complete sharing CAC [6], in which all channels are shared with no restriction, with partition call admission control and determined that complete sharing can generate higher bandwidth utilization.

Grand and Horlait [10] proposed a mobile IP reservation protocol which guarantees end-to-end QoS for mobile terminals. In their CAC scheme they blocked new flows when a threshold value is reached. Nasser and Hassanein [11] proposed a threshold-based CAC scheme. While allocating a common threshold value for all new calls, separate thresholds for different types of handoff calls are allocated based on the bandwidth requirement order. The bandwidth reserved for existing calls is reduced or expanded depending on the availability of bandwidth resources. Ogbonmwan et al. [12] investigated the use of three threshold values for a system with two service classes. Three separate thresholds are used to reserve channels for voice handoff calls, new voice calls, and handoff data calls. These threshold values are periodically re-evaluated based on workload conditions. Similarly, Jeong et al. [13] proposed a handoff scheme for multimedia wireless traffic. Their CAC scheme reserves a guard band for handoff calls in order to minimize the delay which occurs after a handoff. Zhang and Liu [14] and Huang et al. [23] proposed adaptive guard channel based CAC schemes to control channels allocated for handoff calls based on the ratio of dropped handoff calls. Cheng and Lin [15] proposed a hierarchical wireless architecture over IPv6-based networks. They used a coordinated CAC scheme which adjusts guard channels based on current workload conditions. Chen et al. [16] used a threshold-based CAC scheme to offer differentiated QoS to mobile customers. They introduced priority levels in service classes, which let users decide the priority of their traffic in case of congestion. These above-cited studies are based on threshold-based admission control and the goal is to satisfy the handoff call dropping probability requirement, while maximizing resource usage. Our work in this paper is to satisfy QoS requirements in terms of both handoff and new call dropping probabilities while maximizing system revenue for multiple service classes.

There have been CAC studies that partition system resources and allocate distinct partitioned resources to serve distinct service classes [17–25]. Haung and Ho [17] presented a distributed CAC algorithm that runs in each cell and partitions channel resources in the cell into three partitions: one for real-time calls, one for non-real-time calls, and one for both real-time and non-real-time calls to share. To be able to satisfy more stringent QoS requirements of handoff calls, they also applied a threshold value to new calls. To estimate call arrival rates in each cell in the heterogeneous network, they used an iterative algorithm. Choi and Bahk [18] compared partitioning-based CAC with complete sharing CAC [6] in the context of power resources rather than channel resources in QoS-sensitive CDMA networks. They concluded that partitioning CAC exhibits comparable performance with sharing CAC.

Li et al. [19] proposed a hybrid cutoff priority algorithm for multimedia services. Different cutoff thresholds were assigned to individual services. Handoff calls were given a higher threshold while new calls, controlled by a lower threshold, are served only if there are still channels available bounded by the lower threshold. Each service class is characterized by its QoS requirements in terms of channels required. Ye et al. [20] proposed a bandwidth reservation and reconfiguration mechanism to facilitate handoff processes for multiple services.

All these CAC algorithms cited above make acceptance decisions for new and handoff calls to satisfy QoS requirements in order to keep the dropping probability of handoff calls and/or the blocking probability of new calls lower than pre-specified thresholds. Also all these algorithms concern QoS requirements, without considering revenue issues of service classes. Chen et al. [21] first proposed the concept of maximizing the "payoff" of the system by admission control in the context of multimedia services. Chen et al. [24] later considered a class of threshold-based CAC algorithms for maximizing the "reward" earned by the system with QoS guarantees. They have also utilized threshold-based CAC algorithms designed for reward optimization to derive optimal pricing, leveraging a demand-pricing relation that governs demand changes as pricing changes [25]. In this paper, we propose a new partitioning-based CAC algorithm, namely spillover-partitioning CAC, with the goal to maximize system revenue with QoS guarantees. We compare this algorithm with existing CAC algorithms. Our results show that spillover-partitioning CAC generates higher revenue (for solution quality) in a shorter time (for solution efficiency). We also introduce a *fast* spillover CAC algorithm which tradeoffs solution quality for solution efficiency. We show that the approximate solution obtained is close to the optimal solution but the solution efficiency is greatly improved.

The rest of the paper is organized as follows. Section 2 states the system model and gives assumptions used in characterizing the operational environment of a mobile wireless network. Section 3 describes the spillover-partitioning call admission control algorithm based on partitioning and complete sharing for revenue optimization with QoS guarantees in mobile wireless networks. Section 4 analyzes performance characteristics of spillover-partitioning algorithms and compares their performance against existing CAC algorithms in terms of maximum revenue generated with QoS guarantees and time spent for finding a solution. Finally, Sect. 5 summarizes the paper, and outlines future research areas.

2 System Model

A cellular network consists of a number of cells, each of which has a base station to provide network services to mobile hosts within the cell. We assume there are a number of distinct service classes characterized by the service priority. For example, a high-priority multimedia service class versus a low-priority voice service class. For ease of exposition, we assume that there are two service classes. We will refer to the high-priority service class as class 1, and the low-priority service class as class 2 in the paper. Further, there are handoff and new calls for each service class with handoff calls being more important than new calls. Each service class, other than requiring a number of channels for the intrinsic bandwidth QoS requirement, imposes a system-wide QoS requirement. For example, the handoff call drop probability of class 1 being less than 5% could be a QoS requirement. Dropped handoff calls dissatisfy users more than blocked new calls do. Each service class i has a QoS constraint on the handoff blocking probability $\mathrm{Bt}_{\mathrm{h}}^{\mathrm{i}}$ and the new call blocking probability $\mathrm{Bt}_{\mathrm{n}}^{\mathrm{i}}$. There is no queueing. If no channel is available at the time of admission, a call will be dropped or blocked. This no queueing design is most applicable to real-time service classes where queueing does not make sense.

From the perspective of a single cell, each service class is characterized by its call arrival rate (calls per unit time), including new calls initiated by mobile users in the cell and for handoff calls from neighbor cells, and its departure rate (calls completed per unit time). Let $\lambda_{\mathrm{n}}^{\mathrm{i}}$ denote the arrival rate of *new* calls of service class i and $\mu_{\mathrm{n}}^{\mathrm{i}}$ be the corresponding departure rate. Similarly, let $\lambda_{\mathrm{h}}^{\mathrm{i}}$ denote the arrival rate of *handoff* calls of service class i, and $\mu_{\mathrm{h}}^{\mathrm{i}}$ be the corresponding departure rate. These parameters can be determined by inspecting statistics collected by the base station in the cell and by consulting with base stations of neighbor cells [24].

Without loss of generality we assume that a cell has C channels where C can vary depending on the amount of bandwidth available in the cell. When a call of service class i enters a service area from a neighboring cell, a handoff call request is generated. Each call has its specific QoS *channel demand* dictated by its service class attribute. The term "channel demand" refers to the number of channels required by a service call. Let k^i denote the number of channels required by a service call of class i. An example is that a class 1 multimedia call would require four wireless channels as its channel demand and hence $k^1 = 4$. A class 2 voice call would require just one wireless channel and hence $k^2 = 1$.

The total revenue obtained by the system is inherently related to the pricing algorithm employed by the service provider. While many pricing algorithms exist [22], the most prevalent with general public acceptance to date is the "charge-by-time" scheme by which a user is charged by the amount of time in service. Let v^i be the price for a call of service class i per unit time. That is, if a call of service class i is admitted into a cell, and subsequently handed off to the next cell or terminated in the cell, a reward of v^i multiplied by the amount of time the service is rendered in the cell will be "earned" by the system. There is no distinction for handoff versus new calls in pricing as long as the call is in the same service class. The performance model developed in the paper will allow the service provider to calculate the revenue earned per *unit time* under an admission control algorithm by each individual cell such that the revenue obtained by the system is maximized while satisfying QoS constraints.

Here, we note that the class priority of a service class is inherently associated with the service class and is not related to traffic demand. A high-priority class call has a high reward rate associated to it and typically requires more channels for service (such as a multimedia call). It could be that a low-priority class has a high traffic demand and thus the system will reserve more channels to the class to satisfy the QoS requirement and to maximize the reward rate obtainable. However, the priority or importance of a class remains the same and will not be changed with increasing traffic demands.

The total revenue $\mathrm{R_T}$ generated by each cell per unit time is the sum of the revenue generated from each service class:

$$\mathrm{R_T} = \mathrm{R_h^1} + \mathrm{R_n^1} + \mathrm{R_h^2} + \mathrm{R_n^2} \tag{1}$$

Here, $\mathrm{R_h^i}$ represents revenue earned from servicing class i handoff calls per unit time, and $\mathrm{R_n^i}$ represents revenue earned from servicing class i new calls per unit time.

The QoS constraints are expressed in terms of blocking probability thresholds, $\mathrm{Bt_h^1}$, $\mathrm{Bt_n^1}$, $\mathrm{Bt_h^2}$, and $\mathrm{Bt_n^2}$, for class 1 handoff, class 1 new, class 2 handoff, and class 2 new calls, respectively. Suppose that the handoff dropping probability and new call blocking probability of class i generated by a CAC algorithm are $\mathrm{B}_\mathrm{h}^{i}$ and $\mathrm{B}_\mathrm{n}^{i}$. Then the imposed QoS constraints are satisfied when:

$$\mathrm{B_h^1} < \mathrm{Bt_h^1};\ \mathrm{B_n^1} < \mathrm{Bt_n^1};\ \mathrm{B_h^2} < \mathrm{Bt_h^2};\ \mathrm{B_n^2} < \mathrm{Bt_n^2}. \tag{2}$$

The optimization problem that, we are solving is to maximize R_T in Eq. 1 subject to the constraints specified in Eq. 2.

3 Spillover-Partitioning Call Admission Control

For ease of disposition, we assume that two service classes exist with class 1 being the high-priority class. The algorithm can be easily extended to the case in which more classes exist. Let $\mathrm{c\_min}^{i}_{\mathrm{n/h}}$ denote the minimum number of channels needed to satisfy the new/handoff QoS constraints for class i alone. We first sort all service calls by $\mathrm{c\_min}^{i}_{\mathrm{n/h}}$ to determine the *service order*. For example if the order is $\mathrm{c\_min_h^1} \geq \mathrm{c\_min_n^1} \geq \mathrm{c\_min_h^2} \geq \mathrm{c\_min_n^2}$, then our *service order* would be class 1 handoff calls, class 1 new calls, class 2 handoff calls, and class 2 new calls. We then divide channels into partitions to serve calls in this service order. The number of partitions doubles the number of service classes in order to service both new and handoff calls. When two classes exist, there are four partitions $\mathrm{P_1}$, $\mathrm{P_2}$, $\mathrm{P_3}$, and $\mathrm{P_4}$. The first partition is reserved for class 1 handoff calls only; the second partition is reserved for class 1 calls including both handoff and new calls; the third partition is reserved for class 1 calls and class 2 handoff calls; and the last partition is open to all call types.

The algorithm is "spillover" in the sense that if a service call cannot be admitted into P_i then it will overflow to P_{i+1} and so on, as long as P_i is a partition that can accept it.

Figure 1 illustrates these four partitions allocated by the spillover-partitioning CAC algorithm. When a call is received, the partition with the lowest index which can accept this call is used. If this partition does not have enough channels to accommodate the call, then the next partition is used. As an example, when a high priority (class 1) handoff call is received, $\mathrm{P_1}$ is used unless this partition is full. If this partition is full then $\mathrm{P_2}$ is used and so on. Similar rules will apply for other service calls. For example, if a high priority new call comes, $\mathrm{P_2}$ would be used unless this partition is full. Spillover-partitioning, by the virtue of allocating *several* partitions to high-priority classes, can satisfy stringent constraints of high-priority

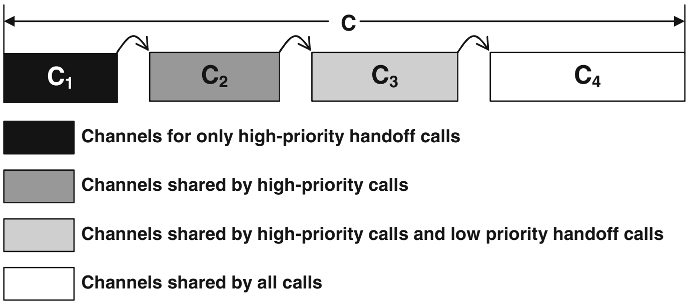
C
C1
C2
C3
C4
Channels for only high-priority handoff calls
Channels shared by high-priority calls
Channels shared by high-priority calls and low priority handoff calls
Channels shared by all calls

Fig. 1 Spillover-partitioning admission control

calls. It improves utilization by sharing certain partitions among different service calls. (e.g., P_4 is shared by all service calls). Due to the use of partitioning, it also greatly reduces computational complexity. Finally, by letting multiple service calls share most of the partitions, it produces optimal solutions. To satisfy the imposed QoS constraints, the spillover-partitioning algorithm looks for "feasible" channel allocation solutions in the form of (C_1, C_2, C_3, C_4) generated such that B^1_n, B^1_h, B^2_n, and B^2_h satisfy the QoS constraints specified by Condition (2). Here, C_i represents the number of channels allocated to P_i.

We model P_1 as an M/M/n/n queue. Calls that cannot be accommodated in P_1 will be spilled into $P_2$and so on. Since P_1 is modeled as M/M/n/n queue, the high priority handoff arrival rate into P_2, denoted by $\lambda^1_{hP_2}$, is simply the spillover rate from the M/M/n/n queue in P_1, i.e.,

$$\lambda^1_{hP_2} = \lambda^1_h \frac{\frac{1}{n^1_{P_1}!}\left(\frac{\lambda^1_h}{\mu^1_h}\right)^{n^1_{P_1}}}{1+\sum_{j=1}^{n^1_{P_1}} \frac{1}{j!}\left(\frac{\lambda^1_h}{\mu^1_h}\right)^j} \tag{3}$$

where $n^i_{P_j} = \left\lfloor \frac{C_j}{k^i} \right\rfloor$ denotes the maximum number of class i calls that can be accommodated in P_j. Here, we note that the arrival process of high priority handoff calls due to overflow traffic from P_1 to P_2 is also assumed to be Poisson as an approximation to obtain close-form solutions. We also model P_2 as an M/M/n/n queue shared by high priority calls only. The spillover rates from P_2 to P_3 thus are given by:

$$\lambda^1_{hP_3} = \lambda^1_{hP_2} \frac{\frac{1}{n^1_{P_2}!}\left(\frac{\lambda^1_{hP_2}}{\mu^1_h}+\frac{\lambda^1_n}{\mu^1_n}\right)^{n^1_{P_2}}}{1+\sum_{j=1}^{n^1_{P_2}} \frac{1}{j!}\left(\frac{\lambda^1_{hP_2}}{\mu^1_h}+\frac{\lambda^1_n}{\mu^1_n}\right)^j} \tag{4}$$

$$\lambda^1_{nP_3} = \lambda^1_n \frac{\frac{1}{n^1_{P_2}!}\left(\frac{\lambda^1_{hP_2}}{\mu^1_h}+\frac{\lambda^1_n}{\mu^1_n}\right)^{n^1_{P_2}}}{1+\sum_{j=1}^{n^1_{P_2}} \frac{1}{j!}\left(\frac{\lambda^1_{hP_2}}{\mu^1_h}+\frac{\lambda^1_n}{\mu^1_n}\right)^j} \tag{5}$$

where $\lambda^i_{hP_j}$ and $\lambda^i_{nP_j}$ are, respectively, class i handoff and new call spillover rates to P_j.

As P_3 contains more service class types with distinct channel demands, there is no simple analytical solution available based on queueing models as we have done for P_1 to P_2

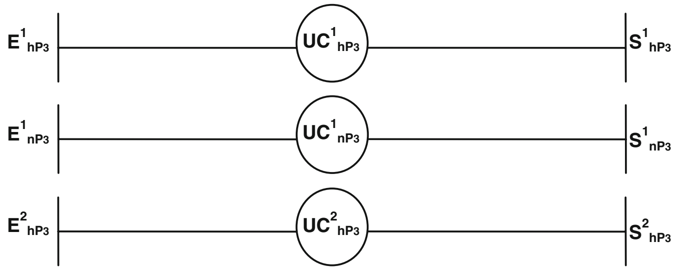

Fig. 2 SPN model for the third partition

and it requires the use of a more sophisticated performance model for P_3. We develop a mathematic model based on Stochastic Petri Net (SPN) [26] as shown in Fig. 2 to model P_3 and derive the spillover rates from P_3 to P_4. The reason we use SPN for performance assessment is that SPN provides a concise representation of the underlying Markov model which potentially may contain tens of thousands of states. A SPN model also allows general time distributions rather than just exponential distributions to be associated with event times if necessary. We use *places* (circles) to hold calls being admitted and serviced by the system, and *transitions* (vertical bars) to model event arrivals. A transition is enabled when its input place (if exists) is non-empty and its associated enabling predicate (if exists) is evaluated true. A transition rate is associated with a transition to specify how often the event associated with the transition occurs. Since P_3 can only accommodate class 1 handoff and new calls and class 2 handoff calls, in Fig. 2 we use places $\mathrm{UC}^1_{\mathrm{hP3}}$, $\mathrm{UC}^1_{\mathrm{nP3}}$ and $\mathrm{UC}^2_{\mathrm{hP3}}$ to hold class 1 handoff calls, class 1 new calls, and class 2 handoff calls, respectively. We use $\mathrm{M}(\mathrm{UC}^1_{\mathrm{hP3}})$, $\mathrm{M}(\mathrm{UC}^1_{\mathrm{nP3}})$, and $\mathrm{M}(\mathrm{UC}^2_{\mathrm{hP3}})$ to represent the number of calls they hold. Transition $\mathrm{E}^1_{\mathrm{hP3}}$ models arrivals of class 1 handoff calls at rate $\lambda^1_{hP_3}$; $\mathrm{E}^1_{\mathrm{nP3}}$ models arrivals of class 1 new call arrivals at rate $\lambda^1_{nP_3}$; $\mathrm{E}^2_{\mathrm{hP3}}$ models class 2 handoff call arrivals at rate λ^2_h; $\mathrm{S}^1_{\mathrm{hP3}}$ models departures of class 1 handoff calls with a service rate of $\mathrm{M}(\mathrm{UC}^1_{\mathrm{hP3}})$ multiplied by per-call service rate μ^1_{h}; $\mathrm{S}^1_{\mathrm{nP3}}$ models departures of class 1 new calls with a service rate of $\mathrm{M}(\mathrm{UC}^1_{\mathrm{nP3}})$ multiplied by per-call service rate μ^1_{n}; S^2_{hP3} models departures of class 2 handoff calls with a service rate of $\mathrm{M}(\mathrm{UC}^2_{\mathrm{hP3}})$ multiplied by per-call service rate μ^2_{h}.

A new service request arrival is admitted in P_3 only if the partition has enough channels to accommodate the new request. Therefore, we assign enabling predicates to guard $\mathrm{E}^1_{\mathrm{hP3}}$, $\mathrm{E}^1_{\mathrm{nP3}}$ and $\mathrm{E}^2_{\mathrm{hP3}}$, with C_3 being the constraint. Specifically, the enabling predicate of $\mathrm{E}^1_{\mathrm{hP3}}$ and $\mathrm{E}^1_{\mathrm{nP3}}$ is $[\mathrm{M}(\mathrm{UC}^1_{\mathrm{hP3}}) + \mathrm{M}(\mathrm{UC}^1_{\mathrm{nP3}})]\mathrm{k}^1 + \mathrm{k}^1 + \mathrm{M}(\mathrm{UC}^2_{\mathrm{nP3}})\mathrm{k}^2 \leq \mathrm{C}_3$. The enabling predicate of $\mathrm{E}^2_{\mathrm{hP3}}$ is $[M(\mathrm{UC}^1_{\mathrm{nP3}}) + \mathrm{M}(\mathrm{UC}^1_{\mathrm{hP3}})]\mathrm{k}^1 + \mathrm{k}^2 + \mathrm{M}(\mathrm{UC}^2_{\mathrm{hP3}})\mathrm{k}^2 \leq \mathrm{C}_3$. Handoff calls and new calls in class 1 and handoff calls in class 2 rejected by P_3 will be spilled into P_4 with rates $\lambda^1_{hP_4}$, $\lambda^1_{nP_4}$and $\lambda^2_{hP_4}$, respectively, given by:

$$\lambda^1_{hP_4} = \lambda^1_{hP_3} - rate(E^1_{hP_3}) \tag{6}$$

$$\lambda^1_{nP_4} = \lambda^1_{nP_3} - rate(E^1_{nP_3}) \tag{7}$$

$$\lambda^2_{hP_4} = \lambda^2_{hP_3} - rate(E^2_{hP_3}) \tag{8}$$

Here, *rate*($\mathrm{E}^{\mathrm{i}}_{\mathrm{c}}$) is calculated by the expected value of a random variable X defined as $X = \lambda^{\mathrm{i}}_{\mathrm{c}}$ if $\mathrm{E}^{\mathrm{i}}_{\mathrm{c}}$ is enabled; 0 otherwise. To obtain *rate*($\mathrm{E}^{\mathrm{i}}_{\mathrm{c}}$) the SPN model is first converted into a Markov model from which the steady state probability distribution is solved analytically. Then *rate*($\mathrm{E}^{\mathrm{i}}_{\mathrm{c}}$) is computed as the probability-weighted average of X.

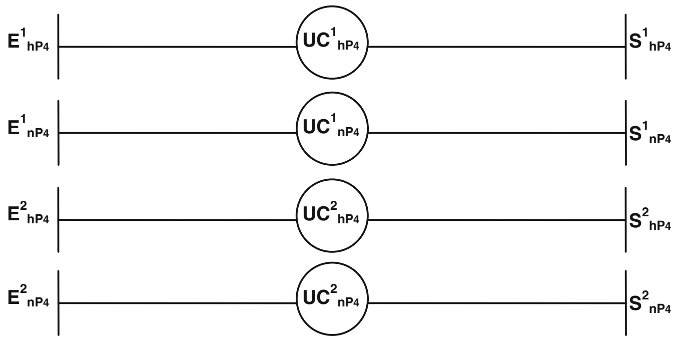

Fig. 3 SPN model for the fourth partition

Similarly, we develop an SPN model shown in Fig. 3 to model P_4 as well as to compute B^i_h and B^i_n of class i in P_4. The SPN model for P_4 is very similar to the SPN model for P_3 except that this model includes one more SPN subnet for modeling class 2 new calls. It can be shown that the blocking/dropping probabilities B^1_n, B^1_h, B^2_n, and B^2_h can be calculated by:

$$B^1_h = \frac{\left(\lambda^1_{hP_4} - rate(E^1_{hP_4})\right)}{\lambda^1_{hP_4}} \tag{9}$$

$$B^1_n = \frac{\left(\lambda^1_{nP_4} - rate(E^1_{nP_4})\right)}{\lambda^1_{nP_4}} \tag{10}$$

$$B^2_h = \frac{\left(\lambda^2_{hP_4} - rate(E^2_{hP_4})\right)}{\lambda^2_{hP_4}} \tag{11}$$

$$B^2_n = \frac{\left(\lambda^2_{n} - rate(E^2_{nP_4})\right)}{\lambda^2_{n}} \tag{12}$$

The revenue earned per unit time, R^1_n, R^1_h, R^2_n, and R^2_h in Eq. 1, can be calculated from the four performance models developed for P_1, P_2, P_3, and P_4. Specifically, since class 2 new calls can only be in the 4th partition, R^2_n is calculated as:

$$R^2_n = \left(1 - B^2_n\right)\lambda^2_n \cdot v^2 \tag{13}$$

Class 2 handoff calls can only be in the last two partitions. Hence R^2_h is calculated by summing the revenue earned from the last two partitions as follows:

$$R^2_h = rate(E^2_{hP_3}) \cdot v^2 + \left(1 - B^2_h\right)\lambda^2_{hP_4} \cdot v^2 \tag{14}$$

Class 1 new calls can be in the last three partitions. Hence, R^1_n is calculated as:

$$R^1_n = (\lambda^1_n - \lambda^1_{nP_3}) \cdot v^1 + rate(E^1_{nP_3}) \cdot v^1 + \left(1 - B^1_n\right)\lambda^1_{nP_4} \cdot v^1 \tag{15}$$

Finally, class 1 handoff calls can be in any of the four partitions. Hence R^1_h is calculated by the revenue earned in all four partitions as follows:

$$R^1_h = (\lambda^1_h - \lambda^1_{hP_2}) \cdot v^1 + (\lambda^1_{hP_2} - \lambda^1_{hP_3}) \cdot v^1 + rate(E^1_{hP_3}) \cdot v^1 + \left(1 - B^1_h\right)\lambda^1_{hP_4} \cdot v^1 \tag{16}$$

Table 1 Pure spillover-partitioning CAC

```
Optimal_t search(){
  // Determine the minimum number of channels needed by each service call

  $c\_min^1_h$ = get_min(C, $k^1$, $\lambda^1_h$, $\mu^1_h$) * $k^1$
  $c\_min^1_n$ = get_min(C, $k^1$, $\lambda^1_n$, $\mu^1_n$) * $k^1$
  $c\_min^2_h$ = get_min(C, $k^2$, $\lambda^2_h$, $\mu^2_h$) * $k^2$
  $c\_min^2_n$ = get_min(C, $k^2$, $\lambda^2_n$, $\mu^2_n$) * $k^2$

  for($C_1$=0, B1H=0; $C_1$<=C; $C_1$=$C_1$+$k^1$) {
    for($C_2$=0, B1N=0; $C_2$<=C; $C_2$=$C_2$+$k^1$) {
      for($C_3$=0, B2H=0; $C_3$<=C; $C_3$=$C_3$+$k^2$) {

        $C_4$ = C - ($C_1$+$C_2$+$C_3$)

          // check if at least min number of channels are reserved for each
          // service call
          if(($c\_min^1_h$<=C)&&($c\_min^1_n$<=($C_2$+$C_3$+$C_4$))  &&  ($c\_min^2_h$<=$C_3$+$C_4$)&&($c\_min^2_n$<=$C_4$))) {

              optimal= check_optimality($C_1$,$C_2$,$C_3$,$C_4$)
        }

        B2N = B2N || ($B^2_n$ <= $Bt^2_n$)
        B2H = B2H || ($B^2_h$ <= $Bt^2_h$)
        B1N = B1N || ($B^1_n$ <= $Bt^1_n$)
        // decreasing $C_4$ won't generate a feasible solution
        if ($B^2_n$ > $Bt^2_n$) break;

      }

      // increasing $C_3$ won't satisfy $Bt^2_n$
      if(($B^2_h$ > $Bt^2_h$) && !B2N) break;
    }

    // increasing $C_2$ won't satisfy $Bt^2_h$
    if(($B^1_n$ > $Bt^1_n$) && !B2H) break

  }
  return (optimal)
}
```

3.1 Search for Optimal Partitioning that Generates Maximal Revenue with QoS Guarantees

We consider two variations of spillover-partitioning. The *pure* spillover-partitioning algorithm finds the optimal partition allocation that would maximize the revenue earned per unit time while satisfying QoS constraints specified in (2). The method essentially is exhaustive search with heuristics being applied to improve search performance by eliminating combinations that would not generate a feasible solution. A second variation is the *fast* spillover-partitioning algorithm that applies a greedy heuristic search method to find a near optimal solution fast at the expense of search quality. It should be noted that a solution found by the *pure* or *fast* spillover-partitioning algorithm will be evaluated by the analytical model developed. As we shall see, the solution found by the *fast* spillover-partitioning algorithm is very close to that found by the *pure* spillover-partitioning algorithm but the search efficiency is greatly improved.

3.1.1 Pure Spillover-Partitioning

Table 1 shows a pseudo code listing the *pure* spillover-partitioning algorithm utilizing exhaustive search for finding optimal resource allocation. Pure spillover-partitioning CAC first eliminates all channel combinations that cannot even satisfy the QoS constraint of class 2 new calls. The logic behind the elimination of these channel combinations is that these combinations could not possibly lead to a feasible solution that could satisfy QoS constraints of all service classes. Exhaustive search is then performed to find the best reward rate achievable.

Specifically, this algorithm first determines the *minimum* number of channels needed to satisfy the QoS constraints of new/handoff calls. This is done by modeling the call admission process of new (or handoff) calls of each service class as an M/M/n/n queue and determining the minimum number of channels that would have satisfied the QoS constraints based on the Erlang-B function to know the call loss probability. This helps to eliminate all (C_1, C_2, C_3, C_4) combinations that do not provide at least the minimum number of channels to each service call in the rest of the search. For a (C_1, C_2, C_3, C_4) combination that satisfies the minimum-channel requirement, we increase C_1, C_2, C_3 and leave the rest to C_4. At some point we reach a C_1, C_2, C_3 combination such that allocating more channels to P_1, P_2, P_3 will not leave enough channels to satisfy the QoS constraint of class 2 new calls (admitted only to P_4). When this happens, we stop increasing the number of channels allocated to P_3 because otherwise it would result in fewer channels allocated to P_4. Similarly, if we see that the QoS constraint of class 2 new calls cannot be satisfied in P_3 and P_4, we eliminate cases in which P_2 is allocated with more channels. We apply the same logic to guide the channel allocation to P_1. After all feasible solutions are found, the algorithm searches for the optimal solution by exhaustive search.

The time complexity of the pure spillover-partitioning algorithm is $O(C^N)$ where C is the number of channels and N is the number of service classes multiplied by 2 (for handoff and new calls). In our example system, N is 4 since there are two service classes (class 1 and 2), each having handoff and new calls. This algorithm tries all possible combinations until it fails to satisfy the QoS constraint of the service class with the least stringent requirements. In the worst case the three loops in Table 1 will try $O(C^N)$ different channel combinations. Therefore, the pure spillover CAC algorithm has a time complexity of $O(C^N)$.

3.1.2 Fast Spillover-Partitioning

Our fast spillover algorithm employs a greedy search method to guide the search of (C_1, C_2, C_3, C_4) to satisfy Condition (2). It may lead to a near optimal solution. This search consists of two parts. In the first part we again look for a partition allocation that generates a feasible solution, and in the second part we search for a partition allocation that generates the optimal revenue, at least locally. Specifically, this algorithm searches for a feasible solution by adjusting the partition size available to each service class in admitting its new or handoff calls. If in the current iteration it fails to satisfy the QoS constraints of a service class, in the next iteration it increases the partition size available to this service class. Similarly, if in the current iteration it satisfies the QoS constraints of a service class, then in the next iteration it decreases the partition size available to this service class to make more resources available to the service classes to attempt to satisfy the QoS constraints. The logic behind the greedy search is to increase the reward rate by checking all channel combinations which are "Δ-close" to the channel combination that currently yields the highest reward rate with QoS guarantees. Here a channel combination (C_1, C_2, C_3, C_4) is "Δ-close" to channel combination (C_1', C_2', C_3', C_4') if $C_1 - \Delta k^1 \leq C_1' \leq C_1 + \Delta k^1, C_2 - \Delta k^1 \leq C_2' \leq C_2 + \Delta k^1, C_3 - \Delta k^2 \leq C_3' \leq C_3 + \Delta k^2, C_4 = C - (C_1 + C_2 + C_3)$, and $C_4' = C - (C_1' + C_2' + C_3')$. The greedy algorithm stops and returns the optimal channel combination found when none of the "Δ-close" channel combinations can further increase the reward rate obtainable while satisfying QoS constraints.

Table 2 shows a pseudo code listing of the fast spillover-partitioning CAC algorithm based on greedy search. The greedy_search method takes Δ as input specifying the locality of the maximum reward. For example, for $\Delta = 2$ the algorithm stops when the partition allocation (C_1, C_2, C_3, C_4) leads a reward that is greater than the reward generated for all partition allocations (C_1', C_2', C_3', C_4'), where $C_1 - 2k^1 \leq C_1' \leq C_1 + 2k^1, C_2 - 2k^1 \leq C_2' \leq C_2 + 2k^1, C_3 - 2k^2 \leq C_3' \leq C_3 + 2k^2, C_4' = C - (C_1' + C_2' + C_3')$. This greedy search

Table 2 Pure spillover-partitioning CAC

```
Optimal_t greedy_search($\Delta$) {
  feasible = find_feasible()
  optimal = find_optimal(feasible)
  return(optimal)
}
Optimal_t find_feasible() {
  // Determine the minimum number of channels needed by each service call
  c_min$^1_h$ = get_min(C, k$^1$, $\lambda^1_h$, $\mu^1_h$) * k$^1$
  c_min$^1_n$ = get_min(C, k$^1$, $\lambda^1_n$, $\mu^1_n$) * k$^1$
  c_min$^2_h$ = get_min(C, k$^2$, $\lambda^2_h$, $\mu^2_h$) * k$^2$
  c_min$^2_n$ = get_min(C, k$^2$, $\lambda^2_n$, $\mu^2_n$) * k$^2$
  for (feasible = NULL;;) {  // do until break
    // set partitions according to minimum channels needed
    pad = k$^1$ - (c_min$^2_h$ % k$^1$)
    // pad channels to allocate P$_1$ and P$_2$ with a multiple of k$_1$ channel
    C$_4$ = c_min$^2_n$ + pad
    C$_3$ = c_min$^2_h$ - c_min$^2_n$
    C$_2$ = (c_min$^1_n$ - C$_3$ - C$_4$)
    C$_1$ = (C - c_min$^1_n$)
      // check if at least min number of channels are reserved for each
      // service call
      if ((c_min$^1_h$<=C)&&(c_min$^1_n$<=(C$_2$+C$_3$+C$_4$))&&      (c_min$^2_h$ <= C$_3$+C$_4$)&&(c_min$^2_n$
      <= C$_4$))) {
      feasible = check_optimality(C$_1$,C$_2$,C$_3$,C$_4$)
      if(feasible == NULL){
        changed = false
        // increase the minimum channel needed for each service call if QoS
        // is not satisfied
        if(B$^2_n$ > Bt$^2_n$) {
          changed = true
          c_min$^2_n$ = c_min$^2_n$ + k$^2$
        }
        if(B$^2_h$ > Bt$^2_h$) {
          changed = true
          c_min$^2_h$ = c_min$^2_h$ + k$^2$
        }
        if(B$^1_n$ > Bt$^1_n$) {
          changed = true
          c_min$^1_n$ = c_min$^1_n$ + k$^1$
        }
        if(B$^1_h$ > Bt$^1_h$) {
          changed = true
          c_min$^1_h$ = c_min$^1_h$ + k$^1$
        }
        // Adjust minimum number of channels to preserve partition order
        if(!changed)  c_min$^2_n$ = c_min$^2_n$ + k$^2$
        if(c_min$^2_n$ > c_min$^2_h$) c_min$^2_h$ = c_min$^2_n$
        if(c_min$^2_h$ > c_min$^1_n$) c_min$^1_n$ + k$^1$
        if(c_min$^1_n$ > c_min$^1_h$) c_min$^1_h$ = c_min$^1_n$
      }
      else break; // a feasible solution is found
    }
    else break; // no solution is found
  }
  return (feasible)
}
Optimal_t find_optimal(feasible) {
  cur = feasible
  optimal = NULL
  // do until finding the local maximum
  for (;(optimal==cur);cur=optimal)
    // search for a local maximum
    for(C$_1$=cur.C$_1$-(k$^1$*$\Delta$);C$_1$<=cur.C$_1$+(k$^1$*$\Delta$);C$_1$=C$_1$+k$^1$)
      for(C$_2$=cur.C$_2$-(k$^1$*$\Delta$);C$_2$<=cur.C$_2$+(k$^1$*$\Delta$);C$_2$=C$_2$+k$^1$)
         for(C$_3$=cur.C$_3$-(k$^2$*$\Delta$);C$_3$<=cur.C$_3$+(k$^2$*$\Delta$);C$_3$=C$_3$+k$^2$){
          C$_4$ = C - (C$_1$+C$_2$+C$_3$)
          if(c_min$^2_n$<=C$_4$){
            optimal = check_optimality(C$_1$,C$_2$,C$_3$,C$_4$)
        }
  return(optimal)
}
```

method first determines a feasible solution via find_feasible, and later uses this solution as the starting point to determine a feasible solution that generates a reward higher than the reward generated by all (C'_1, C'_2, C'_3, C'_4) combinations.

Here is how find_feasible works: It first determines the minimum number of channels needed by each service call to satisfy the QoS constraints. Later it repeats the following greedy search steps until it finds a near optimal partition allocation. It sets partitions such that (a) P_4 contains the minimum number of channels needed by class 2 new calls; (b) P_3 and P_4 contain the minimum number of channels needed by class 2 handoff calls; (c) P_2, P_3 and P_4 contain the minimum number of channels needed by class 1 new calls; and (d) finally, P_1 contains the remaining channels. We configure P_4 such that P_1 and P_2 each have a number of channels that are multiple of k^1. Later we check if the current partition allocation generates a feasible solution. If the current allocation does not lead a feasible solution, we increase the minimum number of channels needed by each service class that fails to satisfy QoS constraints, such that it can accommodate one more call. Finally, we adjust the number of channels available for each service call such that $c^1_h \geq c^1_n \geq c^1_h \geq c^1_h$ is preserved and we repeat these steps until we find a feasible solution.

For the current partition allocation (C_1, C_2, C_3, C_4), find_optimal evaluates all combinations of (C'_1, C'_2, C'_3, C'_4), where $C_1 - \Delta^* k^1 \leq C'_1 \leq C_1 + \Delta^* k^1, C_2 - \Delta^* k^1 \leq C'_2 \leq C_2 + \Delta^* k^1, C_3 - \Delta^* k^2 \leq C'_3 \leq C_3 + \Delta^* k^2, C'_4 = C - (C'_1 + C'_2 + C'_3)$ to determine the partition allocation that would generate the highest reward. If the optimal allocation is different from the current allocation, we set the current allocation as the optimal and repeat evaluation until we determine that the current value is optimal.

While in practice fast spillover CAC takes a much shorter time to execute compared with pure spillover CAC, the theoretical worst case time complexity of this algorithms is still $O(C^N)$, with C being the number of channels and N being the number of service classes multiplied by 2, again due to the three loops in the find_optimal function in Table 2.

4 Numeric Data and Analysis

We compare spillover-partitioning algorithm with existing CAC algorithms with revenue optimization and QoS guarantees, including partitioning, threshold-based, partitioning-threshold hybrid CAC algorithms [24] in terms of execution time and revenue maximization. We briefly explain these baseline CAC algorithms as follows. Partitioning CAC divides the total number of channels into several fixed partitions, with each partition being reserved to serve handoff or new calls of a particular service class. For our example system there are four partitions: class 1 handoff calls, class 1 new calls, class 2 handoff calls, and class 2 new calls. Once a partition is reserved, it cannot be used by others. Thus, each partition may be modeled as an M/M/n/n queue by which the rejection/blocking probability in each partition can be easily calculated as the probability of not being able to serve one more call in that specific partition. The optimization problem here is to find the best partition $(C^1_h, C^1_n, C^2_h, C^2_n)$ such that $C^1_h + C^1_n + C^2_h + C^2_n = C$ with reward optimization and QoS guarantees. Threshold-based CAC creates thresholds to differentiate handoff calls from new calls, viz., C^1_{hT} is the threshold for class 1 handoff calls; C^1_{nT} is the threshold for class 1 new calls; C^2_{hT} is the threshold for class 2 handoff calls; and C^2_{nT} is the threshold for class 2 new calls. The meaning of a threshold is that when the total number of channels already allocated exceeds this threshold, the system will not admit calls of the corresponding service type any more. The optimization problem is to find the best set of thresholds $(C^1_{hT}, C^1_{nT}, C^2_{hT}, C^2_{nT})$ such that $C^1_{nT} \leq C, C^1_{hT} \leq C, C^2_{nT} \leq C$ and $C^2_{hT} \leq C$ with reward optimization and QoS satisfaction.

Table 3 Input parameters and default values used in numerical analysis

Parameter	Description	Value
λ_h^1	Class 1 handoff call arrival rate	$1.5 \le \lambda_h^1 \le 6.2$
λ_n^1	Class 1 new call arrival rate	$3.0 \le \lambda_n^1 \le 12.4$
λ_h^2	Class 2 handoff call arrival rate	$2.7 \le \lambda_h^2 \le 28.4$
λ_n^2	Class 2 new call arrival rate	$3.6 \le \lambda_n^2 \le 37.9$
k^1	Class 1 bandwidth requirement	4 channels
k^2	Class 2 bandwidth requirement	1 channel
C	Total number of channels in the cell	80 channels
$\mu_h^1, \mu_n^1, \mu_h^2, \mu_n^2$	Normalized departure rates of class 1 and class 2 calls	1
B_h^1t	Class 1 handoff call threshold blocking probability	0.02
B_n^1t	Class 1 new call threshold blocking probability	0.05
B_h^2t	Class 2 handoff call threshold blocking probability	0.04
B_n^2t	Class 2 new call threshold blocking probability	0.10

Finally partitioning-threshold hybrid CAC (or hybrid CAC for short) is a mix of partitioning and threshold-based CAC. It has five partitions, with four of them reserved for servicing individual handoff or new calls of service class, and one being reserved as a shared partition for servicing all service classes based on thresholds. The optimization problem here is to find the best partition ($C_h^1, C_n^1, C_h^2, C_n^2, C_S$) and the best set of thresholds ($C_{hT,S}^1, C_{nT,S}^1, C_{hT,S}^2, C_{nT,S}^2$) within C_S such that $C_h^1 + C_n^1 + C_h^2 + C_n^2 + C_S = C$ and $C_{nT,S}^1 \le C_S$, $C_{hT,S}^1 \le C_S$, $C_{nT,S}^2 \le C_S$ and $C_{hT,S}^2 \le C_S$ with reward optimization and QoS guarantees.

To compare these algorithms, we run each algorithm under a set of arrival and departure rates. As in [25], we used arrival rates in the range of $1.5 \le \lambda_h^1 \le 6.2$, $3.0 \le \lambda_n^1 \le 12.4$, $2.7 \le \lambda_h^2 \le 28.4$, and $3.6 \le \lambda_n^2 \le 37.9$. Table 3 summarizes the input parameters and default values used in our numerical analysis. We test each algorithm for 25 different arrival rate combinations of class 1 and 2 service classes. The departure rate is normalized to one with respect to the arrival rate. The reason for testing CAC algorithms with a wide range of arrival rates is to create diverse input traffic conditions which some of them representing heavy-load traffic conditions and some of them presenting medium-load to light-load conditions. The upper bound arrival rate of new calls (or handoff calls) in a class has been carefully chosen such that a system with 80 channels can satisfy the QoS constraints when the system serves only new calls (handoff calls correspondingly) of the service class alone. Out of the 25 test cases, we have created 10 test cases representing heavy-load input traffic conditions and the remaining for medium to light-load traffic conditions.

4.1 Performance Comparison in Solution Efficiency

We have implemented all CAC algorithms in C and run the test cases on a PC with 2.0 GHz Intel© Centrino processor running the Windows© XP operating system. Figure 4 shows a "notched-box plot" summarizing the time spent to determine the optimal partition allocation that would generate the maximum reward with QoS guarantees for the set of arrival/departure rates considered. The thick line in a box plot represents the median value in the 20 cases tested. The bottom and top lines of the shaded box represent the execution time values obtained are below 25 and 75%, respectively. The horizontal bar at the top represents the

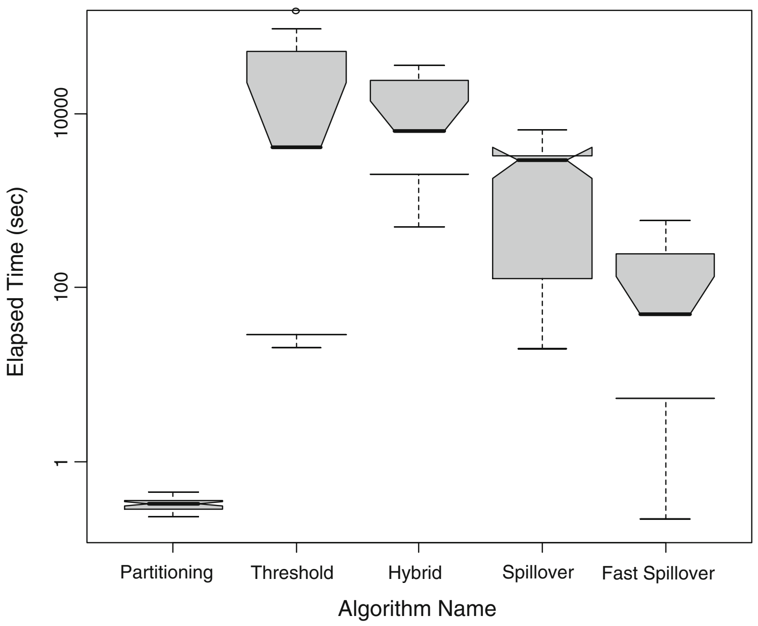

Fig. 4 Notched-box plot for time spent

Table 4 Average time spent to determine channel allocation for revenue optimization by CAC algorithms

Algorithm name	Partitioning	Threshold-based	Hybrid	Spillover	Fast spillover
Average elapsed time (s)	0.32	29,572	27,595	2,202	151

largest non-outlier observation. Notches around medians indicate the uncertainty about the medians obtained by the algorithm. We summarize the average elapsed time needed for each algorithm in Table 4.

With less than a second to calculate the optimal partition allocation, the partitioning algorithm performs the best in terms of execution time. This algorithm allocates fixed partitions, one for each service class. As each partition can be modeled as an M/M/n/n queue, closed-form analytical solutions can be pre-generated and looked up to calculate the blocking/dropping probabilities and the reward obtainable. The second best algorithm is the *fast* spillover-partitioning algorithm with 151 seconds on the average. This algorithm performs significantly better than other algorithms which do not have closed-form solutions. The next best algorithm is the *pure* spillover-partitioning algorithm with 2,202 s. This algorithm calculates the maximum channel allocation in less than 8% of the average time spent by the threshold-based and the hybrid CAC algorithms. The heuristics applied improve the execution time of this algorithm significantly, while not sacrificing the optimality of the solution. The next best algorithm is the partitioning-threshold hybrid CAC with 27,595 s. This algorithm has fixed partitions, one for each service class. The remaining channels are allocated to a share partition that accommodates all service classes based on threshold-based CAC. Finally, threshold-based CAC is the worst in execution time, with 29,572 s on the average. This algorithm is pure threshold-based, with each service class having a threshold. It is computationally expensive to determine optimal thresholds.

4.2 Performance Comparison in Solution Quality

Figure 5 shows a notched-box plot diagram for the maximum reward generated with QoS guarantees by the optimal channel allocation of each CAC algorithm. Figure 6 shows a notched-box plot diagram for the revenue ratio relative to the upper bound revenue obtained,

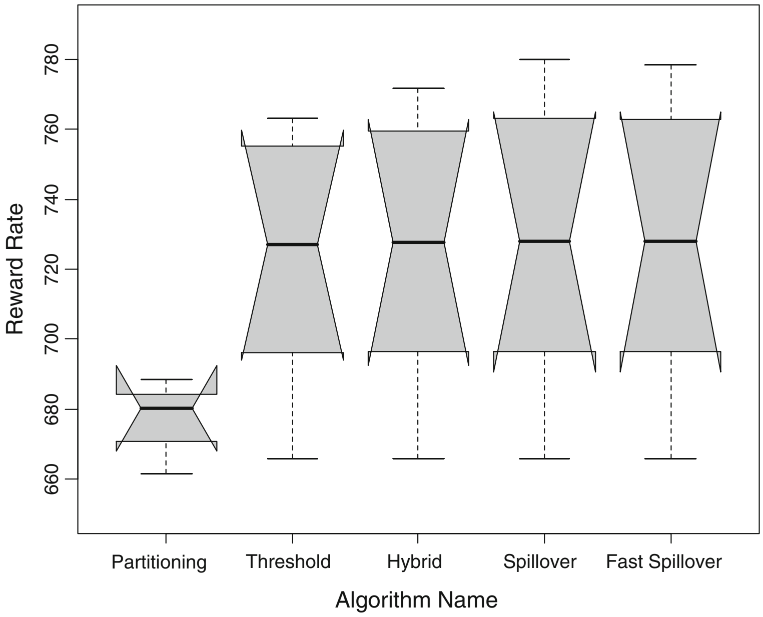

Fig. 5 Notched-box plot for revenue generated

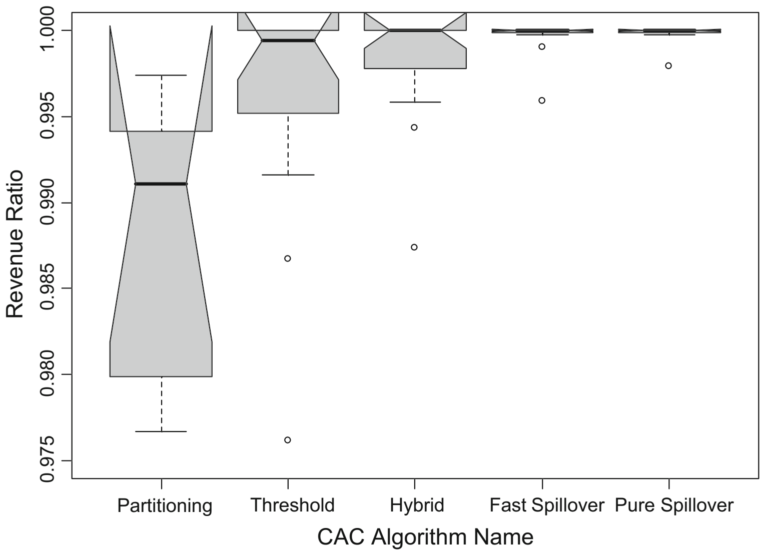

Fig. 6 Notched-box plot for revenue ratio relative to the upper bound revenue

Table 5 Average and maximum reward rates generated by CAC algorithms

Algorithm name	Partitioning	Threshold-based	Hybrid	Spillover	Fast spillover
Average reward rate	667	707	708	710	710
Maximum reward rate	688	763	771	780	778
Revenue ratio	0.9884	0.9956	0.9979	0.9998	0.9995

i.e., the ratio of the reward rate obtained by each CAC algorithm to the upper bound reward rate obtained by an *ideal* CAC algorithm with infinite channel resources to accept calls. We also summarize the average and maximum reward rates as well as the revenue ratio by each algorithm in Table 5. We see that both pure and fast spillover CAC algorithms compare favorably over existing CAC algorithms and are able to generate revenue very close to the upper bound revenue generated. The revenue generated by fast spillover CAC is comparable to that generated by pure spillover CAC, both having about the same revenue ratio relative to the upper bound revenue obtained.

Figure 7 shows specific revenue generated by spillover-partitioning CAC against that generated by an *ideal* CAC algorithm with infinite channel resources to accept calls, as well as against those generated by existing CAC algorithms for 10 high-load test cases (presented in the order of increasing traffic load). These test cases represent situations in which the system is heavily loaded with high call traffic under which spillover performs significantly better than existing CAC algorithms for revenue optimization with QoS guarantees. In these cases, the partitioning algorithm fails to generate feasible solutions. The partitioning algorithm reserves a fixed partition for each service class. Thus, it could not share resources effectively and very often could not even generate a feasible solution when call arrival rates are high. As a result, it performs the worst. At higher arrival rates, both *pure* and *fast* spillover-partitioning CAC are capable of generating revenue very close to the maximum obtainable revenue generated by the *ideal* CAC algorithm. Further, they perform about 1% better than partitioning-threshold hybrid CAC and 2% better than the threshold-based CAC when the system is heavily loaded.

While 1–2% in solution quality in reward rate may not seem much, the net gain in revenue (e.g., dollars) expressed here is (in the unit of) per unit time (e.g., minute). Thus, the cumulative reward accrued over a period of time would be very significant. Moreover, the solution

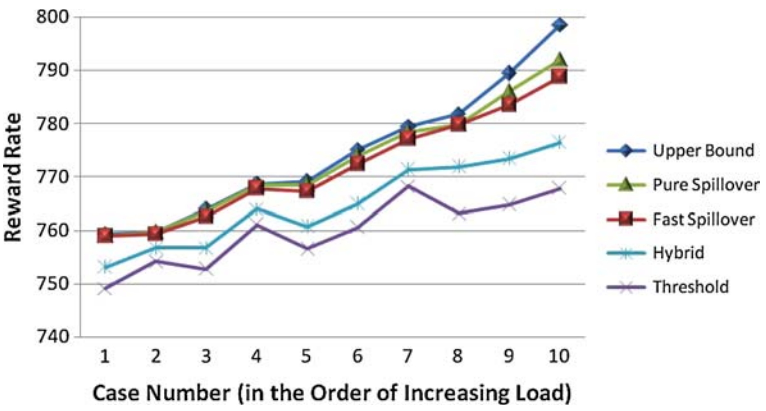

Fig. 7 Comparing spillover-partitioning CAC against ideal and existing CAC algorithms

efficiency by spillover-partitioning makes it practically possible to obtain at least near-optimal solutions to be applied at runtime, compared with threshold-based CAC algorithms, which, because of high computational complexity, would take days to obtain a solution.

4.2.1 Sensitivity Analysis of Solution Quality

Recall that fast spillover CAC consists of two phases. The first phase searches for a feasible initial solution. The second phase performs a greedy search to increase the solution quality. The greedy search continues until the algorithm cannot improve the solution quality any further. Therefore, the initial solution conceivably may have a high impact on the final solution quality. Below we perform a sensitivity analysis to test the sensitivity of the solution quality of the initial solution (relative to the solution found by pure spillover CAC) with respect to the traffic demand and QoS constraints, and see whether it affects the solution quality of the final solution of fast spillover CAC.

Figure 8 shows the solution quality of the initial solution as the traffic demand increases, going from case 1 with $\lambda_h^1 = 3.66$, $\lambda_n^1 = 7.32$, $\lambda_h^2 = 2.69$, and $\lambda_n^2 = 3.59$ to case 12 with $\lambda_h^1 = 3.66$, $\lambda_n^1 = 7.32$, $\lambda_h^2 = 5.99$, and $\lambda_n^2 = 7.98$. We observe that the solution quality of the initial solution relative to that of the pure spillover algorithm remains about the same and is largely insensitive to the traffic demand, with the minimum solution quality ratio at 0.96 and the maximum solution quality ratio at 0.98. Figure 8 also shows the solution quality of the final solution found by fast spillover CAC. We see that the solution quality of the final solution found by fast spillover CAC is largely insensitive to the initial solution found in the first phase of the algorithm.

Figure 9 shows the solution quality of the initial solution as we reduce QoS constraints, going from case 1 with $B_h^1t = 0.01$, $B_n^1t = 0.025$, $B_h^2t = 0.02$, and $B_n^2t = 0.05$ to case 5 with $B_h^1t = 0.05$, $B_n^1t = 0.125$, $B_h^2t = 0.1$, and $B_n^2t = 0.25$. The figure shows three traffic demand cases, namely, high, medium, and low. We see that in all traffic demand cases when QoS constraints are reduced (made less stringent), the solution quality of the initial solution decreases because less stringent QoS constraints allow the find_feasible function to find a solution more easily. This indicates that the initial solution is sensitive to QoS constraints. However, the final solution result found by the greedy search remains the same regardless of the solution quality of the initial solution. We conclude that the solution quality of the final

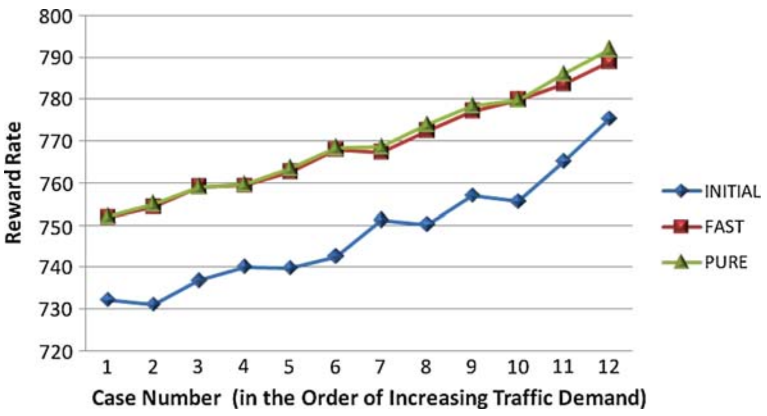

Fig. 8 Sensitivity of initial solution quality with respect to traffic demand

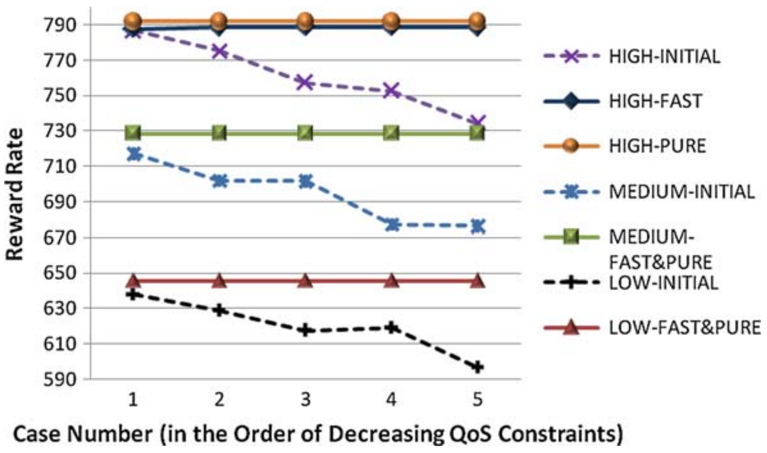

Fig. 9 Sensitivity of initial solution quality with respect to QoS constraints

solution found by the fast spillover CAC algorithm is insensitive to the solution quality of the initial solution.

5 Conclusion and Future Work

In this paper we have developed and analyzed spillover-partitioning CAC for serving multiple service classes in mobile wireless networks for revenue optimization with QoS guarantees. We compared the performance of spillover-partitioning CAC with existing CAC algorithms in terms of the solution efficiency (time spent) and solution quality (revenue generated). We presented two versions of spillover-partitioning CAC: *pure* spillover-partitioning CAC that determines the exact optimal solution and *fast* spillover-partitioning CAC that determines a near optimal solution by using a greedy search method. Although partitioning CAC was significantly faster than all other algorithms, it performed poorly in terms of revenue optimization. The threshold-based and the hybrid CACs performed reasonably well in terms of solution quality. However, these algorithms performed poorly in solution efficiency. We showed that both versions of spillover-partitioning CAC are able to generate higher rewards than existing CAC algorithms while providing QoS guarantees. The 1–2% difference in solution quality is considered significant because the objective function is revenue *per unit time*. Moreover both versions are able to generate solutions with much higher search efficiency. In particular, *fast* spillover-partitioning CAC offers very high solution efficiency while generating near optimal solutions comparable to optimal solutions found by pure spillover-partitioning or threshold-based CAC algorithms.

We also performed a sensitivity analysis of the solution quality of the initial solution found by fast spillover CAC (relative to the upper bound) with respect to the total traffic demand and QoS constraints. We observed that the solution quality of the initial solution is relatively insensitive to the traffic demand. However, the initial solution quality is strongly affected by QoS constraints, exhibiting a lower solution quality when QoS constraints are less stringent. Nevertheless, we observed that the final solution quality is relatively insensitive to the initial solution quality found by fast spillover CAC over a wide range of traffic demands and QoS constraints. In the future, we plan to thoroughly validate analytical results with more test cases generated through random test case generation as well as with simulation studies.

References

1. Lin, Y. B., & Chlamtac, I. (2001). *Wireless and mobile network architecture*. New York, NY: Wiley.
2. Hong, D., & Rappaport, S. S. (1989). Priority oriented channel access for cellular systems serving vehicular and portable radio telephones. *Communications, Speech and Vision, IEE Proceedings I, 131*(5), 339–346.
3. Hong, D., & Rappaport, S. S. (1986). Traffic model and performance analysis for cellular mobile radio telephone systems with prioritized and non-prioritized handoff procedures. *IEEE Transactions on Vehicular Technology, VT35*(3).
4. Guerin, R. (1988). Queuing-blocking systems with two arrival streams and guarded channels. *IEEE Transactions on Communication, 36*, 153–163.
5. Yang, S.-T., & Ephiremides, A. (1996). *On the optimality of complete sharing policies of resource allocation. IEEE 35th Decision and Control* (pp. 299–300). Japan: Kobe.
6. Epstein, B., & Schwartz, M. (1995). *Reservation strategies for multi-media traffic in a wireless environment*. 45th IEEE vehicular technology conference (pp. 165–169). Chicago, IL.
7. Fang, Y. (2003). Thinning algorithms for call admission control in wireless networks. *IEEE Transactions on Computers, 52*(5), 685–687.
8. Wang, J., Zeng Q., & Agrawal, D. P. (2003). Performance analysis of a preemptive and priority reservation handoff algorithm for integrated service-based wireless mobile networks. *IEEE Transactions on Mobile Computing, 2*(1), 65–75.
9. Lai, F. S., Misic, J., & Chanson, S. T. (1998). *Complete sharing versus partitioning: Quality of service management for wireless multimedia networks*. 7th international conference on computer communications and networks, (pp. 584–593). Lafayette, Louisiana.
10. Le Grand, G., & Horlait, E. (2001). *A Predictive end-to-end QoS scheme in a mobile environment*. 6th IEEE symposium on computers and communications pp. (534–539). Hammamet, Tunisia.
11. Nasser, N., & Hassanein, H. (2004). *Prioritized multi-class adaptive framework for multimedia wireless networks*. IEEE international conference on communications (pp. 4295–4300). Paris, France.
12. Ogbonmwan, S. E., Li, W., & Kazakos, D. (2005). *Multi-threshold bandwidth reservation scheme of an integrated voice/data wireless network*. 2005 international conference on wireless networks,(pp. 226–231). Maui, Hawaii: Communications and Mobile Computing.
13. Jeong, D., Choi, D.-K., & Cho, Y. (2001). *The performance analysis of QoS provisioning method with buffering and CAC in the multimedia wireless internet*. 54th IEEE vehicular technology conference, (pp. 807–811). Atlantic City, New Jersey.
14. Zhang, Y., & Liu, D. (2001). *An adaptive algorithm for call admission control in wireless networks*. IEEE Global telecommunications conference (pp. 3628–3632). San Antonio, TX.
15. Cheng, S.-T., & Lin, J.-L. (2005). IPv6-based dynamic coordinated call admission control mechanism over integrated wireless networks. *IEEE Journal on Selected Areas in Communications, 23*, 2093–2103.
16. Chen, H., Kumar, S. & Kuo, C.-C. J. (2000).*Differentiated QoS aware priority handoff in cell-based multimedia wireless network*. IS&T/SPIE's 12th International Symposium (pp. 940–948). San Joes, CA.
17. Haung, Y.-R., & Ho, J.-M. (2002). Distributed call admission control for a heterogeneous PCS network. *IEEE Transactions on Computers, 51*, 1400–1409.
18. Choi, J.-G., & Bahk, S. (2001). *Multiclass call admission control in QoS-sensitive CDMA networks*. IEEE international conference on communications (pp. 331–335). Helsinki, Finland.
19. Li, B., Lin, C., & Chanson, S. T. (1998). Analysis of a hybrid cutoff priority algorithm for multiple classes of traffic in multimedia wireless networks. *Wireless Networks, 4*, 279–290.
20. Ye, J., Hou, J., & Papavassilliou, S. (2002). A comprehensive resource management for next generation wireless networks. *IEEE Transactions on Mobile Computing, 1*(4), 249–263.
21. Chen, I. R., & Chen, C. M. (1996). Threshold-based admission control policies for multimedia servers. *The Computer Journal, 39*(9), 757–766.
22. Keon, N. J., & Anandalingam, G. (2003). Optimal pricing for multiple services in telecommunications networks offering quality-of-service guarantees. *IEEE/ACM Transactions On Networking, 11*(1), 66–80.
23. Huang, L., Kumar, S., & Kuo, C.-C. J. (2004). Adaptive resource allocation for multimedia QoS management in wireless networks. *IEEE Transactions on Vehicular Technology, 53*, 547–558.
24. Chen, I. R., Yilmaz, O., & Yen, I. L. (2006). Admission control algorithms for revenue optimization with QoS guarantees in mobile wireless networks. *Wireless Personal Communications, 38*(3), 357–376.

25. Yilmaz, O., & Chen, I. R. (2006). *Utilizing call admission control to derive optimal pricing of multiple service classes in wireless cellular networks*. 12th IEEE international conference on parallel and distributed systems (pp. 605–612). Minneapolis, MN.
26. Trivedi, K. S., Ciardo, G., & Muppala, J. (1999). *SPNP Version 6 User Manual*. Department of Electrical Engineering, Durham, NC: Duke University.

Author Biographies

Okan Yilmaz received his Bachelor of Science and Master of Science degrees in computer engineering and information science from Bilkent University, Ankara, Turkey. He received his Ph.D. degree in computer science from Virginia Tech. His research interests include wireless communications, multimedia, mobile computing, and software engineering. Dr. Yilmaz has worked at various telecommunication software companies. He currently works at NeuStar Inc., Sterling, Virginia. Dr. Yilmaz is a member of IEEE/CS.

Ing-Ray Chen received the BS degree from the National Taiwan University, Taipei, Taiwan, and the MS and PhD degrees in computer science from the University of Houston. He is a professor in the Department of Computer Science at Virginia Tech. His research interests include mobile computing, wireless networks, security, data management, multimedia, distributed systems, real-time intelligent systems, and reliability and performance analysis for which he has published over 60 journal articles. Dr. Chen currently serves as an editor for *Wireless Personal Communications, The Computer Journal, Security and Communication Networks, Journal of Wireless Networks and Mobile Computing*, and *International Journal on Artificial Intelligence Tools*. He is a member of the IEEE/CS and ACM.

Gregory Kulczycki received his PhD degree in Computer Science from Clemson in 2004. He is currently an Assistant Professor of Computer Science at Virginia Tech. His research areas include performance analysis, component-based software development, software engineering, and formal methods. Dr. Kulczycki is a member of the IEEE/CS and ACM.

William B. Frakes is an associate professor in the computer science department at Virginia Tech. He chairs the IEEE TCSE committee on software reuse, and was an associate editor of IEEE Transactions on Software Engineering. He has a B.L.S. from the University of Louisville, an M.S. from the University of Illinois at Urbana-Champaign, and an M.S. and Ph.D. from Syracuse University.