A Class of Call Admission Control Algorithms for Resource Management and Reward Optimization for Servicing Multiple QoS Classes in Wireless Networks and Its Applications

Okan Yilmaz

Dissertation submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Computer Science and Applications

Committee:
Ing-Ray Chen, Chair
Csaba J. Egyhazy
Mohamed Eltoweissy
William B. Frakes
Gregory W. Kulczycki

Date: November 17, 2008
Falls Church, Virginia

Keywords: Call admission control, quality of service, performance analysis, reward optimization, optimal pricing, wireless networks, resource management.

A Class of Call Admission Control Algorithms for Resource Management and Reward Optimization for Servicing Multiple QoS Classes in Wireless Networks and Its Applications

Okan Yilmaz

ABSTRACT

We develop and analyze a class of CAC algorithms for resource management in wireless networks with the goal not only to satisfy QoS constraints, but also to maximize a *value* or *reward* objective function specified by the system. We demonstrate through analytical modeling and simulation validation that the CAC algorithms developed in this research for resource management can greatly improve the *system reward* obtainable with QoS guarantees, when compared with existing CAC algorithms designed for QoS satisfaction only.

We design *hybrid partitioning-threshold*, *spillover* and *elastic* CAC algorithms based on the design techniques of *partitioning*, *setting thresholds* and *probabilistic call acceptance* to use channel resources for servicing distinct QoS classes. For each CAC algorithm developed, we identify optimal resource management policies in terms of partitioning or threshold settings to use channel resources. By comparing these CAC algorithms head-to-head under identical conditions, we determine the best algorithm to be used at runtime to maximize system reward with QoS guarantees for servicing multiple service classes in wireless networks.

We study solution correctness, solution optimality and solution efficiency of the class of CAC algorithms developed. We ensure solution optimality by comparing optimal solutions achieved with those obtained by ideal CAC algorithms via exhaustive search. We study solution efficiency properties by performing complexity analyses and ensure solution correctness by simulation validation based on real human mobility data. Further, we analyze the tradeoff between solution optimality vs. solution efficiency and suggest the best CAC algorithm used to best tradeoff solution optimality for solution efficiency, or vice versa, to satisfy the system's solution requirements. Moreover, we develop design principles that remain applicable despite rapidly evolving wireless network technologies since they can be generalized to deal with management of "resources" (e.g., wireless channel bandwidth), "cells" (e.g., cellular networks), "connections" (e.g., service calls with QoS constraints), and "reward optimization" (e.g., revenue optimization in optimal pricing determination) for future wireless service networks.

To apply the CAC algorithms developed, we propose an application framework consisting of three stages: workload characterization, call admission control, and application deployment. We demonstrate the applicability with the optimal pricing determination application and the intelligent switch routing application.

ACKNOWLEDGEMENTS

I would like to extend my sincere thanks to everyone who has supported me in achieving the biggest milestone of my life.

I am deeply indebted to Dr. Ing-Ray Chen for his invaluable support, advice, and encouragement that he has provided during my dissertation research. His vast experience and guidance on researching, formalizing, documenting and publishing new ideas has played a significant role in completion of my dissertation in a timely manner with desired quality. I was very fortunate to have him as my advisor.

The members of my dissertation committee Dr. Csaba J. Egyhazy, Dr. Mohamed Eltoweissy, Dr. William B. Frakes, and Dr. Gregory W. Kulczycki have generously given their time, support and expertise to bring my dissertation into the final form. I would like to express my gratitude to them for their insights and suggestions. Especially, I am thankful to Dr. Frakes for his guidance on the statistical analysis of my experiment results.

I would like to thank Dr. Injong Rhee of North Carolina State University for allowing me to use mobile user trace data collected by his research group.

I am thankful to the NPAC Management Group members Sandy Lee, Edward Barker, Jim Rooks, and Anjo Thoppil for their continual support and flexible working hours that they have provided during the coursework and the research phases of my study. My

special thanks go to my company NeuStar for sponsoring this study. I also appreciate the friendship and encouragements I have received from the NPAC Development Team.

I thank my brother, Dr. S. Hakan Yilmaz, and sister, Demet Yilmaz, for always believing in me, and my parents, Halil and Yildiz Yilmaz, for teaching me the value of science and mathematics. They have always encouraged me to pursue the highest degree of education.

I am very grateful to my dear wife and friend Serife Turkol Yilmaz for her invaluable and endless support to keep my sanity intact in tough times during my study. She and our cat, Jack, were the sole witnesses of how hard it was to achieve this goal of my life. Without the moral support of my wife, I cannot imagine how I could have seen the light at the end of this tunnel. Finally, I thank her for the greatest gift, the little one.

I dedicate this dissertation to my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 **Introduction**

Over the past few years, the popularity of mobile devices such as cellular phones, laptops, blackberries, hand-held devices, etc. has drastically increased. This has raised the demand for diverse services provided by personal communication service (PCS) wireless networks to mobile users. Wireless networks strive not only by supporting an increasing number of customers but also by providing multiple Quality of Service (QoS) services. This dissertation research concerns the development of a class of call admission control (CAC) algorithms for resource management and for optimizing a given "objective" function specified by the system while satisfying QoS of multiple service classes in wireless networks.

1.1. Background and Problem Definition

A PCS wireless network typically consists of cells, each being covered by a base station. Each base station provides network services to subscribers in its cell. Figure 1-1 shows an example wireless network with three adjacent cells based on a hexagonal representation. In each cell, a mobile unit accesses the wireless network via the base station (BS). Base stations are connected through a base station controller (BSC). Figure 1-1 also can be viewed as an abstract model for future wireless networks where diverse radio access technologies may be used in different cells. A cell may be based on 802.11 or CDMA radio access, in which case an access point/router would be used for bandwidth allocation to regulate connections or calls to the cell. Without loss of

generality, we use a base station to refer to an access point/router, and a cell to refer to a subnet in the latter case.

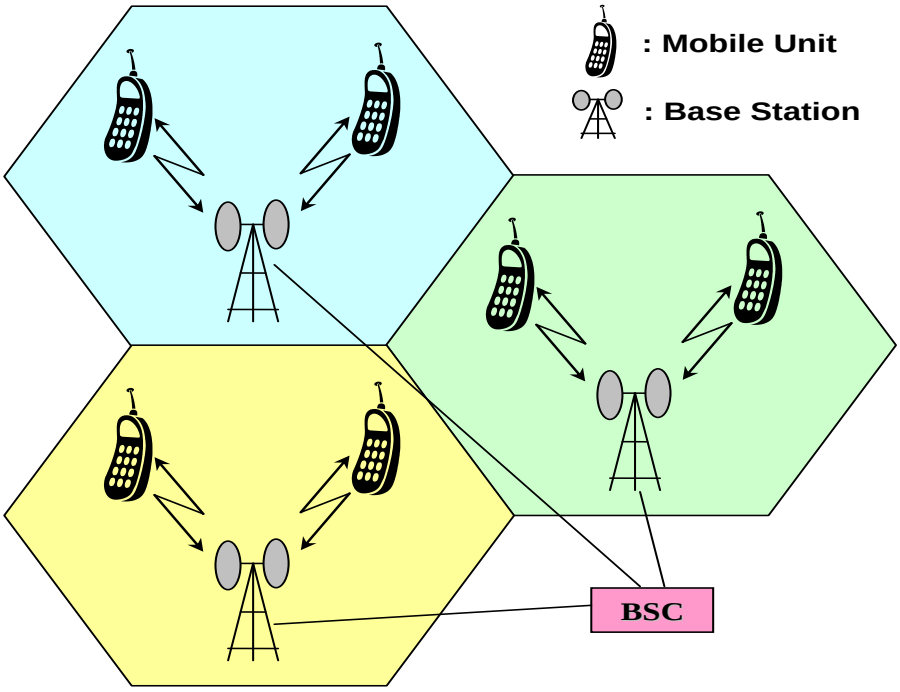

Figure 1-1: PCS Wireless Networks.

A base station typically provides a limited number of communication channels shared by mobile subscribers in the cell. Increasing demands from mobile subscribers with different service profiles require the system to adapt to user needs and growing population without compromising the QoS of services demanded by mobile subscribers that roam in the network. Regardless of the type of radio access technology used, a central issue is bandwidth allocation for ongoing and new connections (or calls) in the cell to maximize an "objective" function specified by the system while satisfying the user QoS constraints. The objective function may be of many forms, e.g., "value", or "reward" to the system from the perspective of system design.

For QoS measures, two metrics have been used in PCS networks, namely, the blocking probability of new calls and the dropping probability of handoff calls due to unavailability of wireless channels. A handoff occurs when a mobile subscriber with an ongoing connection leaves the current cell and enters another cell. Thus, 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 dropping probability by rejecting new call connection requests. Reducing the handoff call dropping probability could result in an increase in the new call blocking probability. As a result, there is a tradeoff between the handoff and new call blocking probabilities. The tradeoff between handoff call dropping probability and new call blocking probability is addressed by CAC algorithms which make call acceptation/rejection decisions with a goal to minimize the handoff dropping probability and/or the new call blocking probability. This dissertation aims to design CAC algorithms with the goals not only to satisfy QoS constraints in terms of dropping/blocking probabilities, but also to maximize the reward obtainable by the system.

To demonstrate the applicability of CAC algorithms developed, this dissertation research develops an application framework and exemplifies with an "optimal pricing" determination application typically performed by service providers. As illustrated in Figure 1-2, the application framework consists of three stages: workload characterization, CAC algorithms for reward optimization with QoS guarantees, and application deployment. These three stages may be viewed as connected in a pipeline fashion with the output of one stage being the input to the next stage.

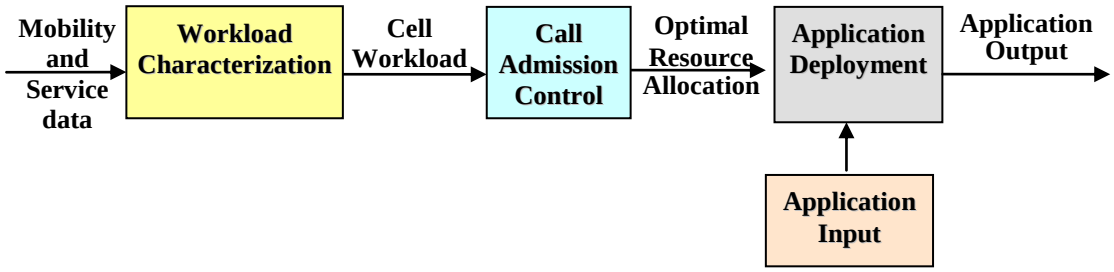

Figure 1-2: Application Framework.

Stage 1: Workload Characterization

The first stage is workload characterization which characterizes arrival and departure rates of users in multiple service classes in wireless networks based on mobility and service data statistically collected by the system. The objective is to accurately estimate handoff and new call arrival and departure rates of each cell in the network. This involves the development a workload characterization algorithm and its validation. The subject is treated in Chapter 4.

Stage 2: Call Admission Control

The second stage is call admission control that makes acceptance decisions to new and handoff calls of multiple service classes to maximize system reward while satisfying specified QoS constraints in terms of the dropping probability of handoff calls and blocking probability of new calls. This is the main research thrust of the dissertation research and is treated in Chapters 5-7 where three new CAC algorithms, namely, partitioning-threshold hybrid CAC, spillover CAC, elastic threshold-based CAC, are developed to tradeoff solution optimality vs. solution efficiency. Further, we perform a

comparative analysis of this class of CAC algorithms in terms of their solution optimality and solution efficiency characteristics in Chapter 8.

Stage 3: Application Deployment

The third stage is to apply CAC algorithms developed to applications that concern CAC and value optimization for servicing multiple QoS classes with QoS guarantees in wireless networks. We illustrate the applicability with the optimal pricing determination application and the intelligent switch routing application. Utilizing a simple demand-pricing empirical formula to predict call arrival rate changes from price changes, we apply CAC algorithms developed in stage 2 to determine optimal pricing for distinct QoS classes such that the overall system value (revenue in this case) obtained is maximized while satisfying QoS constraints of multiple service classes. For the intelligent switch routing application, we apply CAC algorithms to determine the best way to do packet routing through multiple paths with different charge rates such that the net reward obtained by the intelligent switch is maximized with QoS guarantees to prioritized packets. This subject is treated in Chapter 9.

1.2. Research Hypotheses

We summarize the dissertation research hypotheses below. For each hypothesis, we propose mechanisms by which the hypothesis may be supported. Specifically, Hypothesis 1 is for stage 1 design, and Hypothesis 2 (encompassing Hypothesis 2a and Hypothesis 2b) is for stage 2 design.

Hypothesis 1

The *workload characterization* algorithm developed in Chapter 4 of the dissertation

accurately computes the new call and handoff call arrival and departure rates in a cell, given the history of new call and handoff call arrivals and departures by individual mobile units roaming in the cell.

Dependent variables:

- Observed new call (handoff call) arrival/departure rates in the cell.

Independent variables:

- Computed new call (handoff call) arrival/departure rates in the cell.

Method: Use discrete event simulation to obtain the observed new call (handoff call) arrival/departure rates in the cell with the *batch mean analysis* [11], [17], [38] method being applied for achieving 95% confidence interval and 10% accuracy, and with sensitivity analysis being conducted to test the sensitivity of the observed new call (handoff call) arrival and departure rates with respect to the call duration time distribution and the cell residence time distribution.

Effect Size[1]*:* The fraction of variability between observed new call (handoff call) arrival/departure rates vs. computed new call (handoff call) arrival/departure arrival rates.

Hypothesis 2

Hypothesis 2a

There exists an *optimal* resource allocation setting at which the system reward generated is maximized in a cell running *partitioning-threshold hybrid CAC* (developed in Chapter 5), *spillover CAC* (developed in Chapter 6) or *elastic threshold-based CAC* (developed in Chapter 7) of the dissertation, when given new call and handoff call arrival and departure rates, QoS constraints (i.e., blocking and rejection probabilities), charge-by-time reward

[1] Effect size is the sensitivity and effect of independent variables on dependent variables of a hypothesis.

rates, and bandwidth requirements of service classes in the cell, as input, and system resources are sufficient to handle the input traffic workload to satisfy QoS constraints.

Dependent Variables:

- Reward rate generated (reward generated per time unit) by a cell

Independent Variables:

- *partitioning-threshold hybrid CAC* (developed in Chapter 5), *spillover CAC* (developed in Chapter 6) or *elastic threshold-based CAC* (developed in Chapter 7)
- New call arrival rate in the cell
- New call departure rate in the cell
- Handoff call arrival rate in the cell
- Handoff call departure rate in the cell
- QoS constraints of service classes in the cell
- Reward rates of individual service classes in the cell
- Resource (bandwidth) requirements of service classes in the cell

Method:

- Use mathematical modeling and analysis based on queuing theory and stochastic Petri nets to generate quantitative data showing the existence of an optimal resource allocation setting which will maximize the system reward with QoS guarantees over a wide range of parameter values characterizing the input workload condition.
- Use simulation to validate that the reward generated at the optimal resource allocation setting is maximum with the *batch mean analysis* method being

applied for achieving 95% confidence interval and 10% accuracy, and with sensitivity analysis being conducted to test the sensitivity of the observed optimal setting with respect to the call inter-arrival time and departure time distributions.

Effect Size: Reward rate increase achievable by using the CAC algorithm developed at the optimal setting identified compared with at non-optimal settings.

Hypothesis 2b

The *partitioning-threshold hybrid CAC* (developed in Chapter 5), *spillover CAC* (developed in Chapter 6) or *elastic threshold-based CAC* (developed in Chapter 7) of the dissertation can produce higher reward with QoS guarantees in a cell than existing *partitioning* and *threshold-based* CAC algorithms, when given new call and handoff call arrival and departure rates, QoS constraints (i.e., blocking and rejection probabilities), charge-by-time reward rates and bandwidth requirements of service classes in the cell, as input.

Dependent Variables:

- Reward rate generated in a cell by applying a given CAC algorithm.

Independent Variables:

- CAC algorithm:
    - *Partitioning-threshold hybrid CAC* (developed in Chapter 5), *spillover CAC* (developed in Chapter 6) or *elastic threshold-based CAC* (developed in Chapter 7)
    - Partitioning CAC algorithm (existing)

- Threshold-based CAC algorithm (existing)

Method:

- Use mathematical modeling and analysis based on queuing theory and Stochastic Petri Nets (SPN) to generate quantitative data showing that *partitioning-threshold hybrid CAC* (developed in Chapter 5), *spillover CAC* (developed in Chapter 6) or *elastic threshold-based CAC* (developed in Chapter 7) can generate a higher reward rate with QoS guarantees than *partitioning* and *threshold-based* CAC over a wide range of parameter values characterizing the input workload condition.
- Use simulation to validate that the reward rate generated QoS guarantees is higher than that by *partitioning* and *threshold-based CAC* with the *batch mean analysis* method being applied for achieving 95% confidence interval and 10% accuracy, and with sensitivity analysis being conducted to test the sensitivity of the result with respect to the call inter-arrival time and departure time distributions.

Effect Size: Reward rate increase achievable compared with *partitioning* and *threshold-based* CAC algorithms.

1.3. Contribution

The major contribution of the dissertation research lies in the development of CAC algorithms for resource management in PCS networks to support multiple service classes with distinct QoS constraints. The design principles developed are applicable despite rapidly evolving wireless network technologies since they can be generalized with management of "resources" (e.g., wireless channel bandwidth), "cells" (e.g., cellular

networks), "connections" (e.g., service calls with QoS constraints), and "reward optimization" (e.g., revenue optimization in optimal pricing) as future wireless service networks must consider both reward maximization and QoS satisfaction when multiplexing limited wireless resources to multiple service classes with distinct QoS constraints.

Another contribution is the development of an application framework consisting of three stages connected in a pipeline fashion to facilitate the applicability of the CAC algorithms developed to applications that concern with CAC and reward optimization with QoS guarantees. The workload characterization algorithm developed in Chapter 4 is for the first stage that dynamically adapts to environment changes and allows each cell to accurately calculate arrival and departure rates of new and handoff calls in its cell, as input to the CAC algorithms developed for the second stage.

The CAC algorithms developed in Chapters 5-7 for the second stage are innovative in the sense that unlike existing CAC algorithms, they consider not only QoS guarantees but also reward optimization for servicing multiple QoS classes in wireless PCS networks. They combine the benefits of partitioning CAC and threshold-based CAC. We develop, hypothesize and prove that they can perform better than existing partitioning-based or threshold-based CAC algorithms. We thoroughly study their solution correctness, solution optimality and computational complexity. We ascertain the solution optimality by comparing analytical results with those obtained via exhaustive search. We ascertain the computational complexity by performing complexity analyses. We ascertain solution correctness by simulation validation. In our simulation studies, we use real mobility data to model user mobility to demonstrate the feasibility. Further, in

Chapter 8 we analyze the tradeoff between solution optimality vs. solution efficiency and suggest the best CAC algorithm to be used to best tradeoff solution optimality for solution efficiency, or vice versa, to satisfy the system's solution optimality and solution efficiency requirements.

Finally, the optimal pricing determination application and the intelligent switch routing application developed in Chapter 9 exemplify the applicability of the design principles developed for applications that concern QoS satisfaction and reward optimization for servicing multiple QoS classes. In the "optimal pricing" determination application, we consider the effect of price changes on call arrival and departure rates, and apply CAC algorithms develop for stage 2 to determine optimal pricing for multiple service classes which can be set periodically to maximize the system reward (revenue) received by service providers while providing QoS guarantees to all service classes. In the intelligent switch routing application, we also apply CAC algorithms developed in stage 2 to determine the best way to route service calls of different priority classes through trunks (channels) available in multiple routing paths with different charge rates such that the net reward obtained is maximized while QoS constraints of service calls arriving at the switch dynamically are satisfied.

1.4. Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 surveys related work. Chapter 3 gives the system model and discusses assumptions used in characterizing the operational environment of a PCS wireless network. Chapter 4 develops a workload characterization algorithm by which a cell in the PCS wireless network can use to gather call arrival and departure information from individual mobile users and neighbor cells for

providing input to stage 2 of the application framework. In Chapter 4, we use simulation to support Hypothesis 1 and ascertain the accuracy of the workload characterization algorithm developed. Chapters 5-7 cover the class of CAC algorithms developed for reward optimization and QoS satisfaction, namely, partitioning-threshold hybrid CAC, spillover CAC, and elastic threshold-based CAC, respectively. These algorithms employ the concept of resource partitioning and/or thresholds. They differ in their solution optimality and solution efficiency. In these chapters, we support Hypothesis 2. Further, we develop performance models for assessing and comparing performance characteristics of these CAC algorithms, which are validated by simulation. Chapter 8 compares and contrasts these CAC algorithms and draws conclusions regarding solution optimality and solution efficiency, as well as conditions under which one CAC will perform the best compared with other CAC algorithms. In Chapter 9, we exemplify the 3rd stage of the application framework by means of an "optimal pricing" determination application which aims to derive optimal pricing for multiple service classes in PCS networks with QoS guarantees, and an intelligent switch routing application which aims to determine the best way to route prioritized service calls to trunks belong to multiple paths each with a distinct charge rate such that the net reward obtained by the switch is maximized while satisfying imposed QoS constraints of multiple service calls. Finally, Chapter 10 summarizes the dissertation research and outlines some future research areas extending from the dissertation research.

CHAPTER 2 **Related Work**

In this chapter, we survey the literature for prior work on workload characterization, CAC, and optimal pricing determination applications, which are the main pieces of our proposed application framework. We critique the deficiency of prior work in these areas in the literature and elaborate the novelty of our approach for resource management and reward optimization for servicing multiple QoS classes in PCS networks.

2.1. Workload Characterization

In the literature, several mobility prediction models have been proposed to facilitate resource reservation for handoff calls [2], [59], [63], [41], [50], [10]. Aljahai and Znati [2] presented a mobility prediction model used by call admission control algorithms to reserve channel resources for predicted handoff calls. The mobility prediction is done for individual mobile units based on the communication among those cells in the predicted path of a mobile unit. Soh and Kim [59] used a road topology database to allocate channel resources to mobile units based on their mobility prediction patterns. Wu, Wong and Li [63] developed a mobility prediction scheme that allows the system to use the workload conditions in the current cell and the new cell that the call would be handed off to determine the time and the location of the handoff request. Lu and Wu [41] proposed a mobility prediction scheme which divides the cell into inner and outer parts to determine the target cell of a mobile unit on the move. The idea is to use mobility prediction to do resource reservation in the target cells to decrease the handoff call dropping probability and increase the resource utilization. Similarly, Oliveria, Kim

and Suda [50] and Choi, Kim, Kim, and Kim [10] proposed to use a mobile unit's history profile based random walk to do mobility prediction.

In all these prior studies, mobility prediction is applied to each individual mobile unit and the mobility calculation required message exchanges among adjacent cells. Since both the communication and computational complexities of these prediction models are high, it is difficult to apply them at runtime.

Existing workload estimation algorithms have not considered new call and handoff call rates of mobile users. In this dissertation research, we develop a workload characterization algorithm which dynamically computes the call arrival rate and call departure rate to a cell. Our workload characterization algorithm utilizes mobility and call history data to determine new call and handoff call rates. The calculated workload is then used by a call admission control algorithm to allocate resources in each cell for reward optimization while satisfying QoS constraints of various service calls. Our approach differs from [78], [25], [8] in the way we adjust the channel allocation dynamically based on the calculated workload collected from runtime.

2.2. Call Admission Control Algorithms

We can categorize CAC algorithms into two groups: single-class call admission control algorithms and multi-class call admission control algorithms.

2.2.1. Single-class CAC Algorithms

Thus far, CAC for single-class network traffic, such as voice (real-time), has been studied extensively in the literature. The *Guard channel algorithm* [40] assigns a higher priority to handoff calls so that a certain number of channels are reserved for handoff

requests. Hong and Rappaport [22] 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 also [23] proposed a priority oriented algorithm where queuing of handoff calls is allowed. Guerin [20] demonstrated that queuing of both new and handoff calls improves channel utilization while reducing call blocking probabilities. Yang and Ephremides [65] proved that FCFS based sharing [14] can generate optimal solutions for some systems. Most of the above mentioned research methods focus on voice-based cellular systems.

2.2.2. Multi-class CAC Algorithms

Fang [16] 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 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, Zeng and Agarwal [61] 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, Misic, and Chanson [36] compared complete sharing CAC [14], 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 [19] 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 [48] 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, Li and Kazakos [49] proposed 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, and these threshold values are periodically reevaluated based on workload conditions. Similarly, Jeong, Choi, and Cho [26] 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 [78] and Huang, Kumar, and Kuo [25] proposed adaptive guard channel based CAC schemes to adjust the number of channels allocated for handoff calls based on the ratio of dropped handoff calls. Cheng and Lin [8] proposed a hierarchical wireless architecture over IPv5-based networks. They used a coordinated CAC scheme which adjusts the number of guard channels based on the current workload conditions. Chen, Kumar, and Kuo [5] 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 that their traffic receives in case of congestion. These above

studies cited are generally based on threshold-based admission control and the goal is to satisfy the handoff call dropping probability constraint while maximizing resource usage.

There have been CAC studies that partition system resources and allocate distinct partitioned resources to service distinct service classes. Haung and Ho [21] presented a distributed CAC algorithm, running in each cell, which 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 constraints 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 [9] compared partitioning-based CAC with complete sharing CAC [14] in the context of power resources rather than channel resources in QoS-sensitive CDMA networks. They determined that complete partitioning CAC shows very similar performance with complete sharing CAC.

Li, Lin and Chanson [39] 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 constraints in terms of the number of channels required. Ye, Hou and Papavassilliou [66] 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 constraints 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 constraints, without considering reward optimization issues. No prior work exists in analyzing the statistical relationship between the blocking probability of new calls and dropping probability of handoff calls has been done before. Chen and Chen [6] first proposed the concept of maximizing the "payoff" of the system through admission control in the context of multimedia services. This dissertation research extends the work to develop a class of CAC algorithms for channel resource management and reward optimization in PCS wireless networks.

In the dissertation research, we develop a class of CAC algorithms for resource management and reward optimization. We analyze the performance characteristics of these CAC algorithms by a comparative performance analysis both analytically and by simulation based on human mobility trace data to draw conclusions on solution optimality of these CAC algorithms. Since our CAC algorithms are derived from two main CAC techniques, namely, partitioning and threshold-based, we use threshold-based and partitioning CAC algorithms as our baseline CAC algorithms for performance comparison purposes. Here we note that a variant of threshold-based CAC, commonly known as trunk reservation [31], [46], [57], also concerns reward optimization for trunk management in wired networks. Further, it was proven that the solution generated would be optimal in the class of threshold-based CAC algorithms for the case in which the capacity constraint per call is constant (i.e., 1 trunk) for all service classes. In this dissertation, we demonstrate the solution optimality property of elastic CAC over threshold-based CAC for the more general case in which multiple service classes have

distinct intrinsic bandwidth requirements in wireless networks and that QoS constraints of service classes must be satisfied.

2.3. Optimal Pricing Determination

The CAC algorithms developed can be applied to the optimal pricing application. The goal is to determine optimal pricing of multiple service classes such that system reward is maximized while the QoS constraints of service classes are satisfied.

In the literature, the relationship between pricing and demand for telecommunication services have been analyzed by several studies. Aldebert, Ivaldi, and Roucolle [1] presented an empirical study that reveals the relationship between pricing and user demand for a service. Rappoport, Alleman, and Taylor [55] analyzed a consumer survey to estimate the demand for wireless internet access. Milne [47] emphasized the importance of the pricing of telecom services and correlated the affordability of consumers with the demand for these services. Yuksel and Kalyanaraman [77] analyzed the correlation between user's elasticity to telecommunication service price and network bandwidth. They also derived an optimal pricing strategy for a system with a single service class. Basar and Srikant [4] analyzed a single communications link shared by multiple users to optimize the system reward. They considered the case that link capacity increases with the number of users sharing the link and concluded that the system reward-per-unit-bandwidth increases with the number of users while the optimal price per-unit-bandwidth may either increase or decrease based on the link bandwidth. This dissertation work utilizes the empirical demand-pricing formula proposed by [1], [37], [55] to determine the optimal prices of multiple service classes, which not only to

optimize the system reward but also to satisfy the bandwidth requirements and QoS constraints of all service classes and call types in a multi-cell system.

Several dynamic pricing models, in which the price for the service changes based on the system load conditions, have been proposed to control call arrival rates to prevent congestion and satisfy QoS constraints [24], [11], [51], [54], [67], [27], [13], [29], [42], [43], [44] or establish fairness and social welfare [30], [32], [12], [58], [51], [52], [13], [75], [76], [34], [64], [15], [53], [45]. For example, Kelly [30] introduced a scheme that each user chooses a per unit time price that he/she is willing to pay. Kelly, Maulloo, and Tan [32] presented a *proportionally fair pricing* scheme that dynamically assigns pricing and communicates it with the user and the traffic is accepted based on the willingness to pay by the user. Hou, Yang, and Papavassilliou [24] proposed a dynamic pricing approach in response to changing call arrival rates to satisfy QoS constraints. In our dissertation research we dispose dynamic pricing as a valid approach since changing pricing during a call service is disturbing and is not acceptable by most of the callers. Rather, we consider static pricing that can be adjusted optimally by a service provider on a periodic basis to maximize system reward while still satisfying users' QoS constraints. In case of varying demand patterns, depending on day of week (e.g. weekdays vs. weekends) or time of day (e.g. daytime vs. nighttime), a service provider may elect to assign different optimal static pricing in different time segments. This can be done by applying the optimal pricing algorithm based on the workload characteristics in each time segment. With respect to the existing static optimal pricing schemes in the literature, our approach considers both QoS satisfaction and reward optimization for multiple classes, while Paschalidis and Tsitsiklis [52] considered optimal pricing for a single cell only for

welfare maximization without considering QoS constraints. Paschalidis and Yong [51] considered optimal pricing for welfare maximization in *wireline* networks only without considering QoS of multiple service classes. Wang and Brown [62] analyzed the role of CAC in optimal pricing for reward optimization in multi-class wireline networks also without considering QoS guarantees. We believe that reward maximization cannot be established in PCS wireless networks without considering QoS, since poor network quality would negatively affect the demand and consequently decrease the reward earned by the system.

In this dissertation research we utilize demand-pricing relation functions based on empirical studies [1], [37], [55] to predict demand changes when we vary pricing. Recently, Keon and Anandalingam [33] also utilized similar formulas to determine optimal pricing for telecommunications networks offering multiple service classes. In their work they determined optimal pricing for multiple service classes to maximize system reward. Further, they also considered distinct QoS constraints for service classes as in our research. Our work differs from their work as follows. First, their scheme was designed for wired networks in which traffic workloads are defined between a pair of source and destination switches. Thus the optimal price set is for a source and destination switch pair. Our work deals with optimal pricing for multiple service classes in wireless networks in which traffic workloads are characterized by arrival and departure rates of new and handoff calls by users in the system. Thus, the optimal price determined is on a service class basis, and is static across cells in the wireless network. Secondly, they utilized partitioning-based CAC only to determine optimal pricing. Our work develops and utilizes new CAC algorithms (partitioning-threshold hybrid, spillover, and elastic

threshold-based CAC algorithms) with the objective to generate high system reward with QoS guarantees [7], [68], [69], [71], [73]. Lastly, their search algorithm yields only near optimal prices for source and destination switch pairs while our research generates system-wide optimal pricing for multiple service classes.

CHAPTER 3 **System Model**

This chapter presents our system model. We consider a PCS wireless network which provides multiple service classes to its mobile subscribers. We discuss service classes, arrival, departure and reward rates of calls, and QoS constraints of service calls.

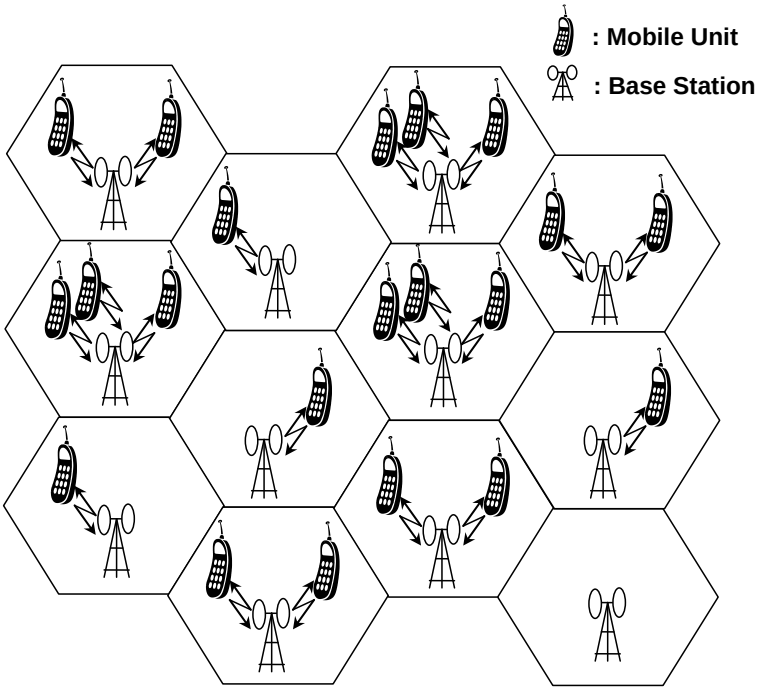

Figure 3-1: PCS Wireless Network Architecture.

A PCS wireless network is modeled by a flat architecture through which cells are connected. In the center of each cell, a base station is used to provide network services to mobile hosts within the cell. Figure 3-1 illustrates the architecture of a PCS wireless network. We assume that there exists a number of distinct *service classes,* S_1, S_2, … , S_n, which are characterized by the "service type" attribute. For example, the service types can be *real-time* and *non-real time*. Further, there are handoff and new calls in each

service type, with handoff calls having a higher priority than new calls. Figure 3-2 illustrates a new call originated from a mobile unit to a landline phone. We call a call originated from a mobile unit as a new call. If a mobile unit moves to an adjacent cell then a handoff call occurs.

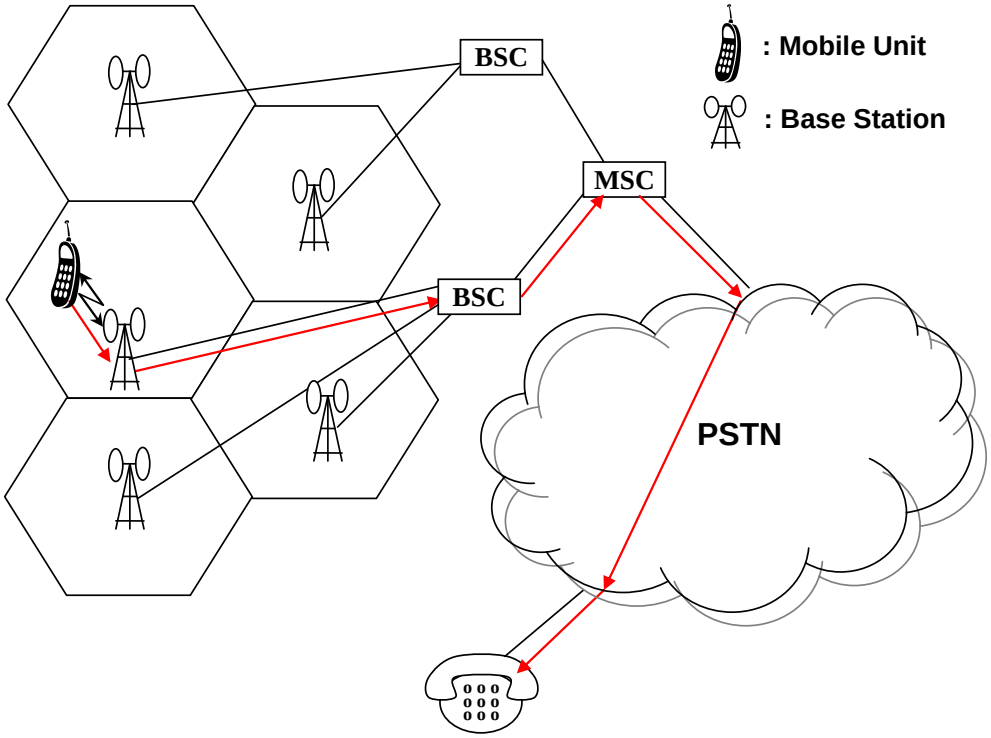

Figure 3-2: A New Call Originated from a Mobile Unit to a Landline Subscriber.

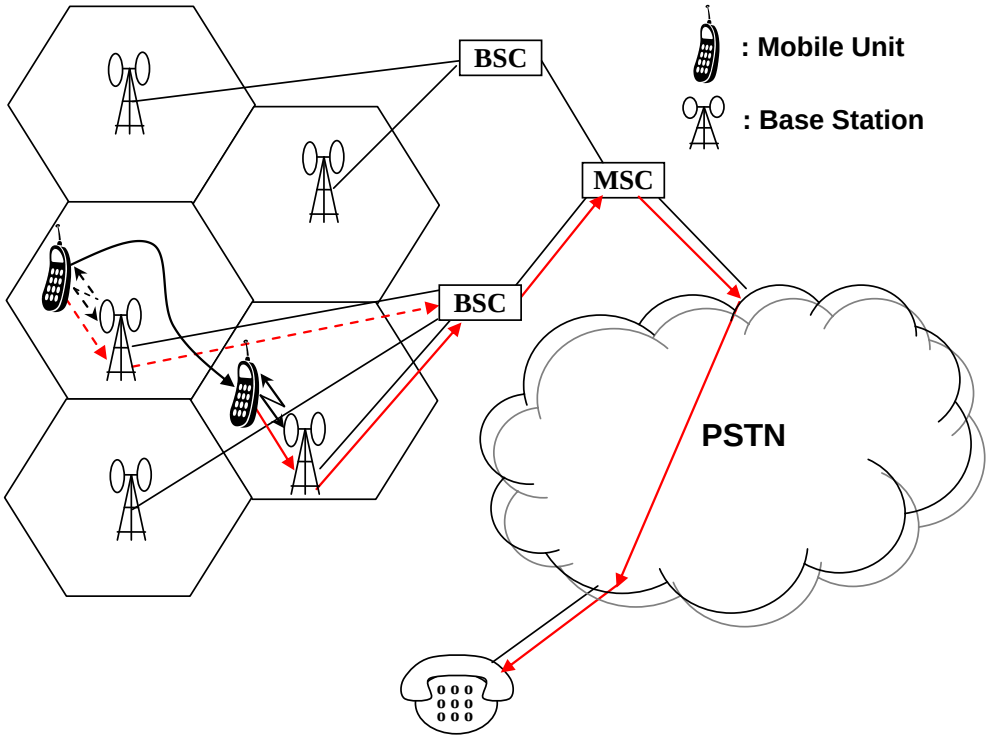

Figure 3-3: A Handoff Call Scenario.

Figure 3-3 illustrates a handoff call scenario. In this figure when the mobile unit moves to an adjacent cell, a handoff call is initiated in the new cell. Each call has its specific QoS bandwidth requirement dictated by its service traffic type attribute. For example a real-time service call may require four channels while a voice call may require one channel. Assume that a service call of service class *i* (regardless of handoff or new) requires k^i channels. 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. Each service type also can possibly impose a system-wide QoS constraint. For example, the handoff call dropping probability of a real-time service call must be less than 5% as a QoS constraint. Dropped handoff calls dissatisfy users more than blocked new calls do. In general for each service class, say *i*, a QoS constraint exists in terms of its handoff call dropping probability B^i_ht and its new call blocking probability B^i_nt.

We note that the main difference between QoS requirements and QoS constraints is that QoS requirements refer to minimum requirements that must be satisfied, e.g., a class 1 new call requires 4 wireless channels, whereas QoS constraints refer to desirable QoS properties that should be satisfied for acceptable service quality and customer satisfaction. We view both QoS requirements and QoS constraints as mandatory as user satisfaction is inherently related to reward optimization. Consequently, CAC algorithms designed for reward optimization and QoS satisfaction should always satisfy both QoS constraints and QoS requirements to provide high QoS to mobile users. Also note that in our analysis we do not address "fairness" since this has a conflicting goal with reward optimization.

From the perspective of a single cell, each service class is characterized by its arrival rate (including new service connections initiated by mobile users in the cell and for handoff service connections from neighbor cells), and departure rate (of leaving the cell). Let λ_n^i denote the arrival rate of *new* calls of service class *i* and μ_n^i be the corresponding departure rate. Similarly, let λ_h^i denote the arrival rate of *handoff* calls of service class *i*, and μ_h^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. In Chapter 4, we will develop a workload characterization algorithm for estimating these parameters.

From the perspective of a PCS network service provider, each service class has a "reward rate" associated such that the system receives some reward corresponding to the reward rate associated with the call multiplied with the service duration when the service is rendered. The system achieves total reward maximization in a distributed manner by

maximizing each individual cell's reward. That is, each cell makes admission control decisions for new and handoff call requests taking into consideration of the reward rate information of these service calls in order to maximize the reward received from servicing new and handoff calls in the cell. We assume that such a reward rate scheme is adopted by the service provider such that a call of service class i has a reward-rate of v^i per time unit. 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 with the amount of time the service is rendered in the cell will be "earned" by the system. There is no distinction for handoff vs. new calls in terms of the reward rate associated with a service class as long as the call is in the same service class. The network service provider could run the "optimal pricing determination" application (discussed in Chapter 8) periodically to determine the best reward rate v^i for service class i such that the total system reward obtained is maximized while satisfying QoS constraints of all services classes.

CHAPTER 4 **Workload Characterization Algorithm**

In wireless cellular systems, CAC algorithms allocate channel resources based on workload conditions. Workload condition of each cell depends on not only the current mobile subscribers roaming in the cell but also mobility and call patterns of those mobile subscribers in the neighboring cells. In this chapter we develop a workload characterization algorithm based on learning mechanisms to characterize the workload condition of each cell, leveraging information dynamically collected by mobile subscribers. The work reported in this Chapter is largely based on [7][2].

4.1. Methodology for Workload Characterization

We propose to use a simple yet efficient learning mechanism for calculating the values of λ^i_n, μ^i_n, λ^i_h and μ^i_h of service class *i* from a cell's perspective. This learning mechanism involves a mobility and service call pattern recognition algorithm being executed on individual mobile devices for scalability reasons. The algorithm summarizes mobility and service call information of each individual mobile user in two data structures, namely, a mobility matrix and a service call table.

The mobility matrix summarizes the statistics of the mobile user going from one cell to the next cell and the residence time (distribution) of each cell, given that the

[2] With kind permission from Springer Science+Business Media: I.R. Chen, O. Yilmaz and I.L. Yen, "Admission control algorithms for reward optimization with QoS guarantees in mobile wireless networks," *Wireless Personal Communications*, Vol. 38, No. 3, August 2006, pp. 357-376.

mobile user comes from a previous cell. Specifically, the matrix stores N_{BCD} and T_{BCD} where N_{BCD} is the number of times that the mobile user entering into cell *D* (the next cell) from cell *C* (the current cell), given that the previous cell is *B*, and T_{BCD} is the average dwell time of the mobile user in cell *C*, given that the previous cell is *B* and the next cell is *D*. For resource management, the information summarized in the mobility matrix is provided to the cells when the mobile user migrates into them. A design variation is to consider not only the previous cell but also a path consisting of the previous cell and the second previous cell. In this design variation, the mobility matrix will store N_{ABCD} and T_{ABCD} for the number of times that mobile user entering into cell *D* (the next cell) from cell *C* (the current cell), given that the previous two cells are *A* and *B* in the sequence, and the average dwell time of the mobile user in cell *C*, given that the next cell is *D* and the previous two cells are *A* and *B*. This design variation trades off storage and processing requirements for information accuracy and improves the accuracy in summarizing the mobility behavior of a mobile user.

Certainly any method used to summarize mobility and service call patterns of a mobile user must be light-weight, given the small processing, storage and communication capabilities of a mobile device nowadays. We propose to adopt a light-weight yet effective self-learning mechanism [35] originally designed for wireless LANs and now applied to cellular networks based on *rewarding* the correct state transition and *penalizing* incorrect state transitions such that the sum of all probabilities is one. Consider the design that takes the previous state, the current state and the next state into account. Also suppose for convenience that a cell has 6 neighbors. If the previous cell is B_1 and the current cell is *C*, then in the mobility matrix we would have entries for $N_{B_1CD_1}$, $N_{B_1CD_2}$,

$N_{B_1CD_3}$, $N_{B_1CD_4}$, $N_{B_1CD_5}$, and $N_{B_1CD_6}$ for six possible next cells D_1, D_2, D_3, D_4, D_5 and D_6 as neighbors to cell C. Suppose that the mobile user actually goes to cell D_4 from cell C. Then, $N_{B_1CD_4}$ would be incremented and the conditional probability of the mobile user entering from cell C to cell D_4 when the previous cell was B_1, $P_{B_1CD_4}$, would be calculated as:

$$P_{B_1CD_4} = \left(\frac{N_{B_1CD_6}}{N_{B_1CD_1} + N_{B_1CD_2} + N_{B_1CD_3} + N_{B_1CD_4} + N_{B_1CD_5} + N_{B_1CD_6}} \right) \quad (1)$$

The probability of the mobile user entering from cell C to cell D_4 would be calculated as:

$$P_{CD_4} = P_{B_1CD_4} + P_{B_2CD_4} + P_{B_3CD_4} + P_{B_4CD_4} + P_{B_5CD_4} + P_{B_6CD_4} \quad (2)$$

For a mobile user that exhibits a certain degree of regularity for movements and calls, the mobility probability values will summarize the regular paths taken by the mobile user. On the other hand, $T_{B_1CD_1}$, $T_{B_1CD_2}$, $T_{B_1CD_3}$, $T_{B_1CD_4}$, $T_{B_1CD_5}$, and $T_{B_1CD_6}$ are updated accordingly depending on the actual path taken by the mobile user. This can be determined by each mobile user easily by keeping track of the average dwell time that the mobile user stays in a particular cell, given the history of the previous cell and the next cell.

The second data structure, a service call table also maintained by individual mobile devices to summarize call patterns, is to be populated as the mobile device goes through a sequence of calls. Specifically, for each cell visited by the mobile device, the table stores four "rate" values related to calls, namely, the arrival rate of a new call made by the mobile device in cell C, denoted by $\Lambda_n(C)$, the departure rate of a new call by the mobile device at cell C, denoted by $\theta_n(C)$, the arrival rate of a handoff call from cell C into its neighbor cells, denoted by $\Lambda_h(C)$, and the departure rate of a handoff call in cell

C, denoted by $\theta_h(C)$. These four rate values reflect the frequency at which new calls are initiated and terminated, and also the frequency at which handoff calls arrive from neighbor cells and terminated by the mobile device, thus summarizing the call patterns of a mobile device. The computational procedure obtaining the values of these rate parameters is also very light-weight, involving only a few variables to be kept in the mobile device's memory which are being updated as calls are made and terminated by the mobile user roaming across cells in the wireless network.

Our approach is to have each cell make admission control decisions to allocate resources to calls based on summarized mobility and service call patterns of those mobile users currently in the cell to know their call expected arrival and departure rates, as well as of those mobile users in the neighbor cells in order to know their expected handoff call arrival rate. From a cell's perspective (say, cell *C*), the arrival rate of handoff calls from mobile users in all the neighbor cells (say, *B*'s), denoted by $\lambda_h(C)$, is given by:

$$\lambda_h(C) = \sum_{B \epsilon M} \sum_{\substack{all \\ users \\ in\ B}} \Lambda_h(B) x P_{BC} \qquad (3)$$

Here *M* is the set of neighbor cells of cell *C* and P_{BC} is the probability that if the mobile user is in cell *B* it will go to cell *C* as the next cell, which can be calculated easily by each individual mobile user through a look-up of its mobility probability matrix by summarizing the probabilities of P_{ABC}, i.e.,

$$P_{BC} = \sum_{all A} P_{ABC} \qquad (4)$$

Let $\lambda_n(C)$ denote the arrival rate of new calls, $\mu_n(C)$ denote the departure rate of a new call in cell *C*, and $\mu_h(C)$ denote the departure rate of a handoff call in cell *C*. Then,

$$\lambda_n(C) = \sum_{\substack{all\ users \\ in\ C}} \Lambda_n(C) \quad (5)$$

$$\mu_n(C) = \frac{\lambda_n(C)}{\sum_{\substack{all\ users \\ in\ C}} \frac{\Lambda_{n(C)}}{\theta_{n(C)}}} \quad (6)$$

and

$$\mu_h(C) = \frac{\lambda_h(C)}{\sum_{\substack{all\ users \\ in\ C}} \frac{\Lambda_{h(C)}}{\theta_{h(C)}}} \quad (7)$$

Note that the arrival rate of all new calls is an aggregate measure summing all new call arrival rates by individual users in the cell, while the departure rate per call is an average parameter, weighted average of all mobile users in the cell.

The design of resource management for reward optimization hinges on the adaptive admission control algorithm executed by individual cells. A cell, say *C*, will dynamically collect mobility and service call patterns in the form of $\Lambda_n(C)$, $\theta_n(C)$, $\Lambda_h(B)$, where B is a neighbor cell of C, and $\theta_h(C)$ from mobile users of a service class currently in its cell and will communicate with neighbor cells regarding the handoff arrival rate to have knowledge of the expected arrival and departure rates of new calls and handoff calls from these mobile users. Then each cell can intelligently allocate resources to serve new and handoff calls of various service types of different *charge* rates (i.e., a call of service class *i* has a "charge-rate" of v^i per time unit) to maximize the reward obtainable, limited by the amount of resources available (bandwidth channels).

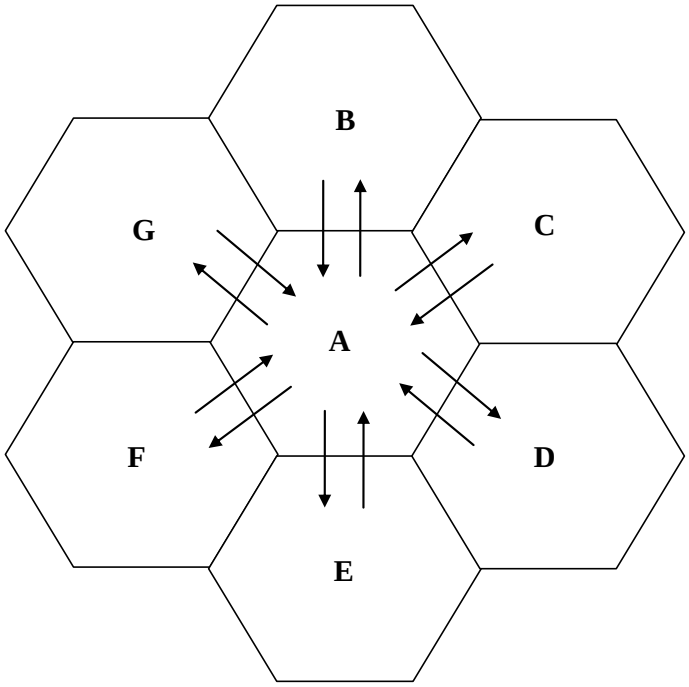

Figure 4-1: An Example System with Cell A Surrounded by Cells B, C, D, E, F and G.

4.2. Example

We illustrate how the workload characterization algorithm works with a 7-cell system shown in Figure 4-1. Cell A is surrounded by cells B through G. In this system handoff calls into cell A come from cells B, C, D, E, F, or G. Thus, the overall rate at which handoff calls enter cell A is the sum of the handoff call rates from cells B through G to cell A. The handoff rate of a mobile user from cell B to cell A is the handoff rate of that mobile user from cell B to all its neighboring cells ($\Lambda_h(B)$) multiplied with the probability that the mobile user moves from cell B to A (P_{BA}). This applies to every mobile user currently in cell B. Similarly, we apply the same calculation for the handoff rate from cells C, D, E, F and G to cell A. Consequently, the arrival rate of handoff calls going into cell A becomes:

$$\lambda_{h(A)}=\sum_{\substack{all\\users\\in\ B}}\Lambda_h(B)xP_{BA}+\sum_{\substack{all\\users\\in\ C}}\Lambda_h(C)xP_{CA}+\sum_{\substack{all\\users\\in\ D}}\Lambda_h(D)xP_{DA}+$$
$$\sum_{\substack{all\\users\\in\ E}}\Lambda_h(E)xP_{EA}+\sum_{\substack{all\\users\\in\ F}}\Lambda_h(F)xP_{FA}+\sum_{\substack{all\\users\\in\ G}}\Lambda_h(G)xP_{GA}$$

The arrival rate of new calls in cell A is simply equal to the rate at which new calls are initiated by all mobile users in cell A. Thus, the arrival rate of new calls in cell A is given by:

$$\lambda_n(A)=\sum_{\substack{all\\users\\in\ A}}\Lambda_n(A)$$

The departure rate of new calls in cell A is the rate at which new calls are terminated averaged over all mobile users in cell A. Therefore, the departure rate of new calls in cell A is:

$$\mu_n(A)=\frac{\theta_n(A)}{\sum_{\substack{all\\users\\in\ A}}\frac{\Lambda_n(A)}{\theta_n(A)}}$$

Similarly, the departure rate of handoff calls in cell A is equal to the rate at which handoff calls are terminated averaged over all mobile users in cell A. Thus,

$$\mu_h(A)=\frac{\theta_h(A)}{\sum_{\substack{all\\users\\in\ A}}\frac{\Lambda_h(A)}{\theta_h(A)}}$$

4.3. Simulation Validation

We develop a wireless network simulator to collect simulation results to validate analytical results obtained through mathematical modeling and analysis to support Hypothesis 1 stated in Chapter 1. In order to obtain creditable results, we use real mobility trace data [56] to model the mobility patterns of human mobile users. In addition, we randomly assign call arrival and departure rates to each mobile user and simulate user call patterns.

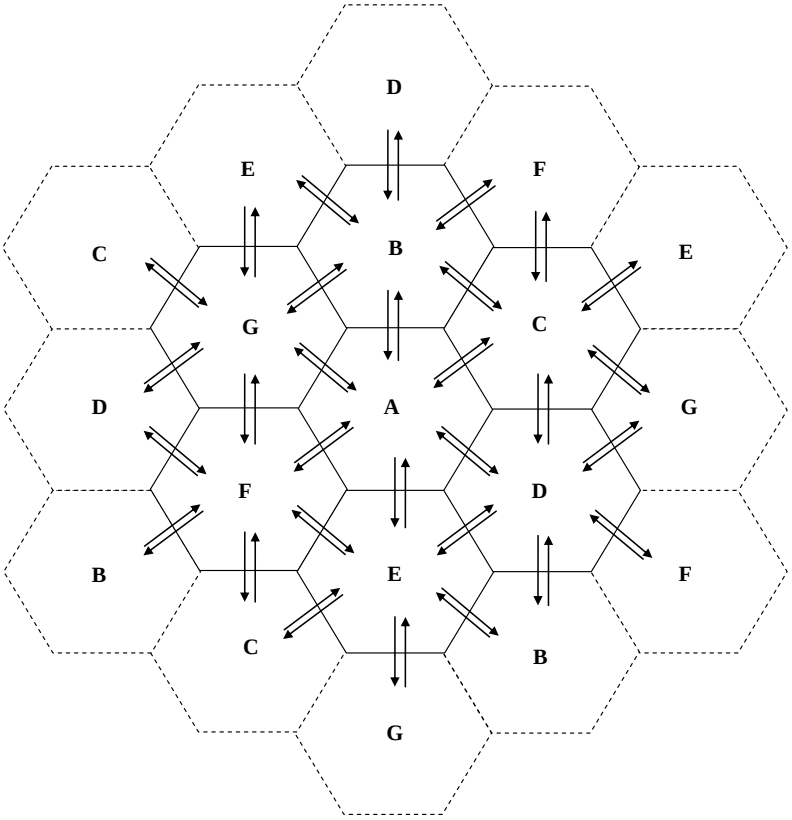

Figure 4-2: A Simulated Wireless Cellular Network with a Wrap-Around Structure.

4.3.1. Simulation Environment

We setup a simulation environment with seven cells as shown in Figure 4-2 with a wrap-around structure to simulate a large PCS wireless system such that a cell on the right (bottom) is connected to the left (top) and a cell is always surrounded by 6 neighbor cells.

Each cell serves a number of mobile users roaming into the cell and making calls from time to time. Each cell makes admission decisions based on the CAC algorithm developed. The system is configured with 1536 mobile users roaming among these cells. We use real trace data to model the mobility of mobile users [56], Poisson distribution to model call arrivals [18] and exponential distribution to model the duration of a call.

4.3.2. Human Mobility Trace Data

Most wireless network simulation studies use mobility models such as random waypoint [28], Brownian or Markovian random walk models. Although these models are very commonly used due to their simplicity, not much statistical validation has been done to determine the accuracy of these models. In order to provide more creditable results, we use real human mobility trace data in the simulation [56]. These mobility trace logs are observed within tens of kilometers in 5 different locations: Korea Advanced Institute of Science and Technology, North Carolina State University, New York City, North Carolina State Fair and Disney World. The mobility of 44 participants carrying GPS receivers were recorded between September 2006 and January 2007. The trace covers more than 1000 hours of human mobility logs. GPS examples are taken in 10 seconds intervals and latitude, longitude, and altitude of each participant is recorded in every trace entry.

In the simulation we determine the boundaries of each location and use our 7-cell map with wraparound structure covering each location. We assume that each cell has a base station in the center and determine the cell location of the participant in each trace entry by comparing the distance of that participant to each base station. The cell of the closest base station is set as the current cell of the participant at the time that the trace entry is generated. A cell change is considered a handoff.

4.3.3. Test Scenarios

For each service class we consider low, medium, and high arrival and departure rates and use these rates as the mean of the arrival and departure rates of mobile users. We repeat tests for the mobility trace data obtained in 3 different locations, namely, Korea Advanced Institute of Science and Technology, New York City, and North Carolina State Fair. We generated altogether 27 test scenarios resulting from a combination of 3 arrival and departure rates (low, medium, high) for high priority calls, 3 arrival and departure rates (low, medium, high) for low priority calls, and 3 separate locations. Table 4-1 illustrates an example of low, medium and high call arrival and departure rates used in the simulation. For each test scenario, we vary the mobile user population to stress the system with high workload and test the effect of workload on CAC algorithms developed.

Table 4-1: Example Call Arrival and Departure Rates Used in Simulation.

Service Class	Low (calls/unit time)		Medium (calls/unit time)		High (calls/unit time)	
	arrival rate	departure rate	arrival rate	departure rate	arrival rate	departure rate
Class 1	1/1800	1/120	1/3600	1/420	1/600	1/120
Class 2	1/5400	1/240	1/1200	1/120	1/300	1/90

4.3.4. Simulation Validation

We support Hypothesis 1 by means of simulation validation based on the simulation environment setup above. We first collect mobility history data of mobile users and then utilize the workload characterization algorithm developed to workload to each cell for 27 different test scenarios under varying mobile user populations. In our test scenarios, since the call arrival and departure rates observed from the simulation in cells do not necessarily follow any specific distribution, we do a *nonparametric correlation*

analysis to determine if the workload results calculated from the workload characterization algorithm for individual service classes are equal to those observed in the cell.

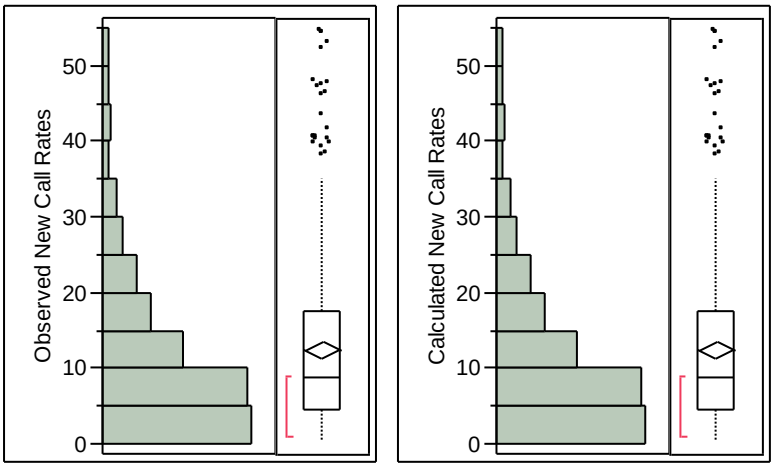

Figure 4-3: Histogram and Box Plot Diagrams of Calculated vs. Observed New Call Arrival/Departure Rates.

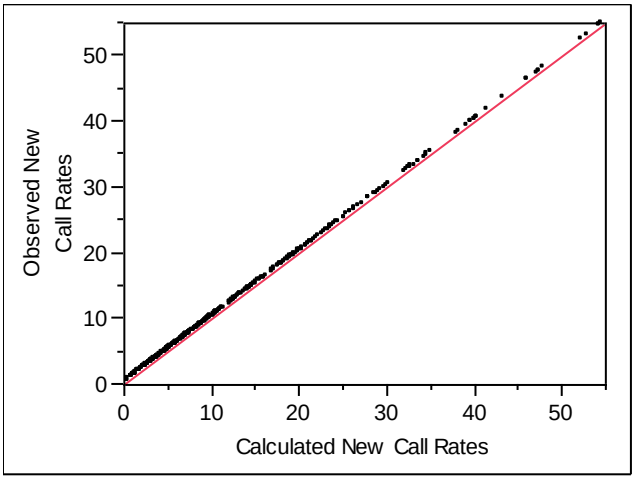

Figure 4-4: Bivariate Fit of Calculated New Call Arrival/Departure Rates vs. Observed New Call Arrival/Departure Rates.

Table 4-2: Comparison of Calculated vs. Observed New Call Arrival/Departure Rates.

New Call Arrival/Departure	Calculated	Observed	Absolute Difference	Percentage Difference
Number of observations	432	432	0	0%
Mean	12.395	12.395	-1.13E-07	-8.00E-07%
Median	8.795	8.795	-1.76E-08	-2.9E-07%
Standard Deviation	11.173	11.173	8.96E-07	8.77E-06%
Range	54.180	54.180	8.73E-06	1.11E-04%
Midspread	13.17	13.17	4.95E-07	5.56E-06%

4.3.4.1 Supporting Hypothesis 1 for New Call Workloads

We perform a nonparametric correlation analysis to support Hypothesis 1 for new call workloads. Figure 4-3 presents the histogram and box plot diagrams of calculated vs. observed arrival/departure rates. Both the histogram and box plot show very similar characteristics. Most of the arrival/departure rates are observed less than 10 calls per unit time. This was due to the heterogeneous nature of the system where the cell in the middle observed most of the new calls while the other cells observed relatively lower traffic. Figure 4-4 shows a bivariate fit line of calculated vs. observed new call arrival/departure rates. The fit line passes through the origin with a slope of 1 indicating a strong correlation. Table 4-2 compares calculated vs. observed new call arrival/departure rates. As the effect size, the Spearman's ρ and probability>$|\rho|$ are calculated as 1.0000 and 0, respectively, thereby supporting our hypothesis strongly and indicating that there is a perfect positive linear relationship between calculated and observed new call arrival/departure values. The fact that Spearman's probability>$|\rho|$ being 0 also demonstrates a highly significant correlation between calculated and observed new call arrival/departure rates.

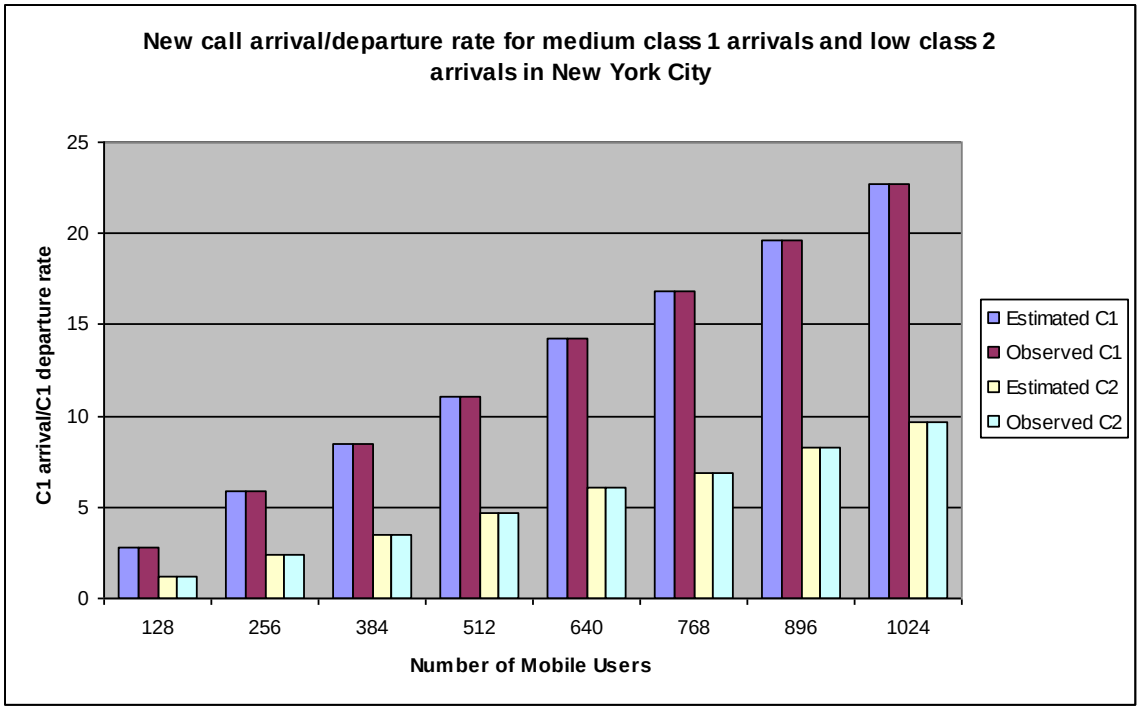

Figure 4-5: Comparison of Calculated vs. Observed New Call Arrival/Departure Rates under Medium Class 1 Arrivals and Low Class 2 Arrivals in New York City.

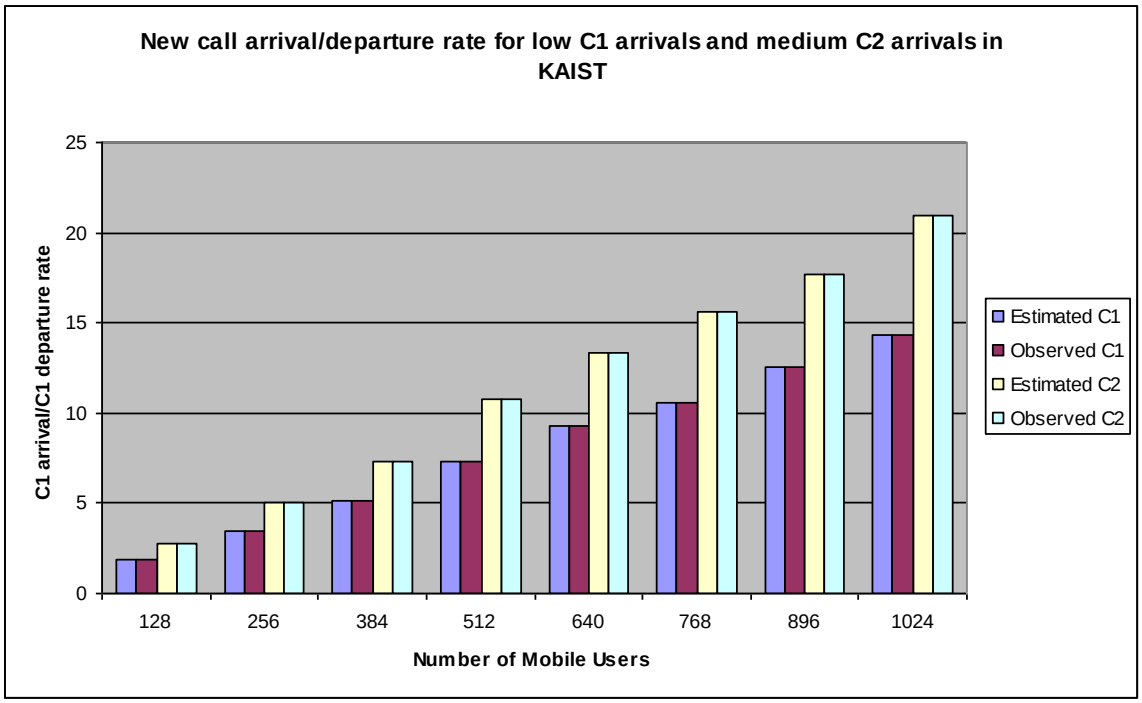

Figure 4-6: Comparison of Calculated vs. Observed New Call Arrival/Departure Rates under Low Class 1 Arrivals and Medium Class 2 Arrivals in KAIST.

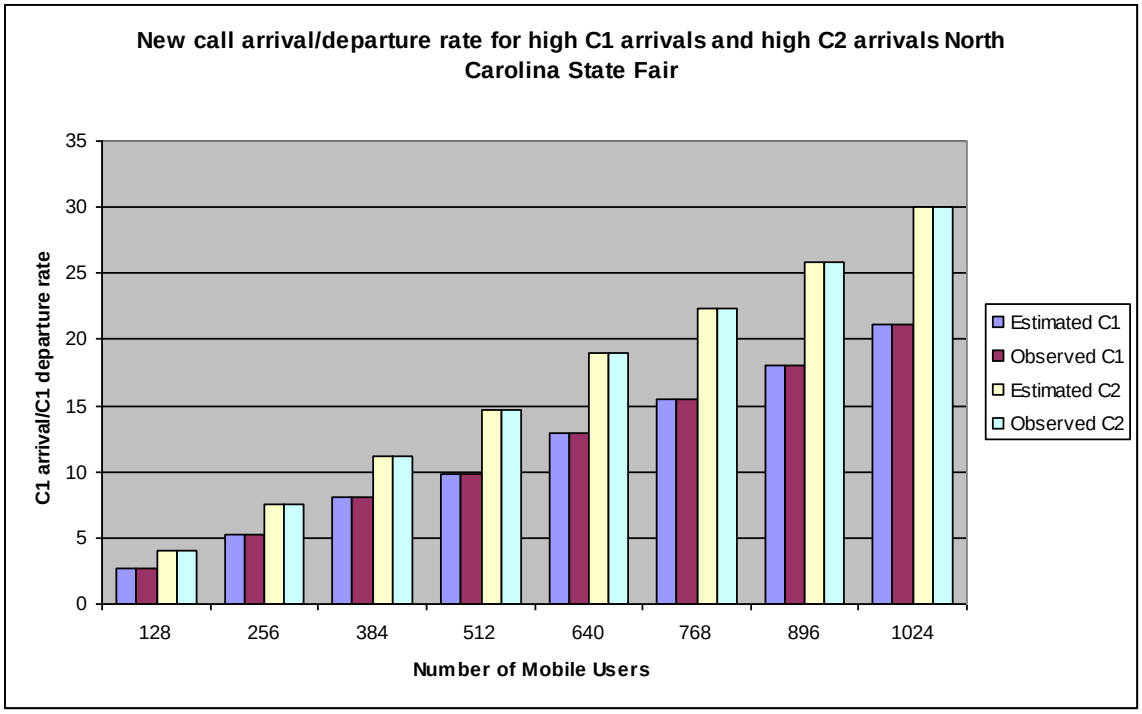

Figure 4-7: Comparison of Calculated vs. Observed New Call Arrival/Departure Rates under High Class 1 Arrivals and High Class 2 Arrivals in North Carolina State Fair.

To further illustrate the computational accuracy, we exemplify three scenarios out of 27 scenarios tested. Figure 4-5, Figure 4-6, and Figure 4-7 compare calculated vs. observed new call arrival/departure rates for these three example scenarios. Figure 4-5 is for the case in which class 1 call arrival/departure rates (at "medium") are higher than class 2 call arrival/departure rates (at "low") in New York City. Figure 4-6 is for the case in which class 1 arrival/departure rates (at "low") are lower than class 2 arrival/departure rates (at "medium") in KAIST. Figure 4-7 is for the case in which class 1 arrival/departure rates (at "high") are about the same as class 2 arrival/departure rates (at "high") in North Carolina State Fair. In all three cases, we see that the calculated arrival/departure rates from the workload characterization algorithm developed are virtually identical to the observed rates for both service classes.

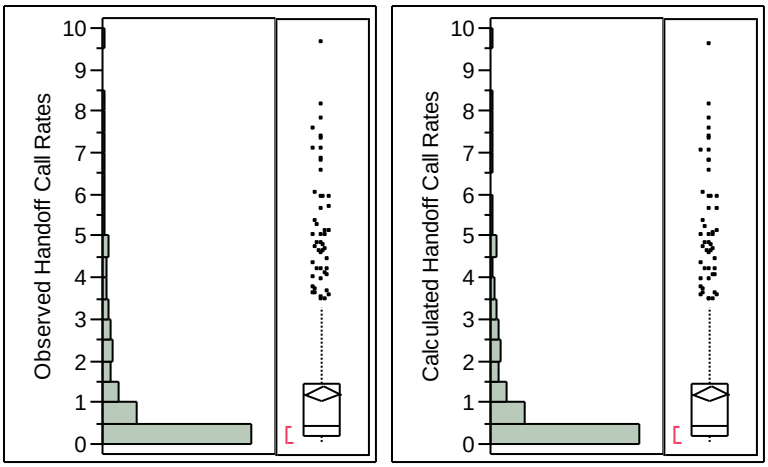

Figure 4-8: Histogram and Box Plot Diagrams of Calculated vs. Observed Handoff Call Arrival/Departure rates.

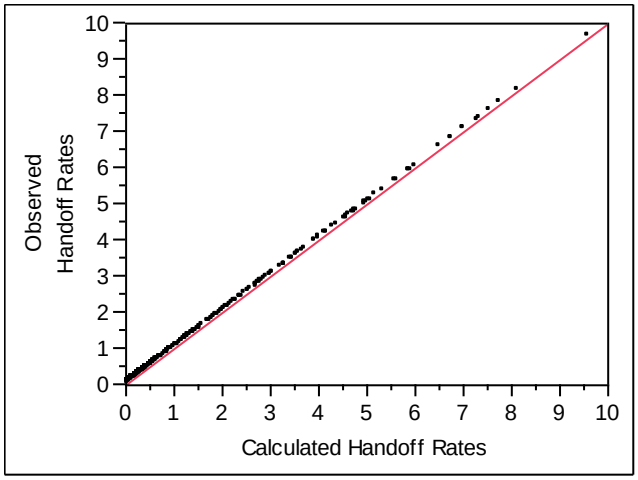

Figure 4-9: Bivariate Fit of Observed New Call arrival/Departure Rates vs. Calculated Handoff Call Arrival/Departure Rates.

Table 4-3: Comparison of Calculated vs. Observed Handoff Call Arrival/Departure Rates.

Handoff Call Arrival/Departure	Calculated	Observed	Absolute Difference	Percentage Difference
Number of observations	432	432	0	0%
Mean	1.217	1.215	0.20E-02	0.056%
Median	0.081	0.081	0.23E-03	0.020%
Standard Deviation	1.688	1.684	0.47E-02	0.020%
Range	9.574	9.554	3.45E-02	4.782%
Midspread	1.262	1.259	3.1E-03	0.45%

4.3.4.2 Supporting Hypothesis 1 for Handoff Call Workload

We also perform a nonparametric correlation analysis to support Hypothesis 1 for handoff call workloads. Figure 4-8 presents the histogram and box plot diagrams of calculated vs. observed arrival/departure rates. Again both the histogram and box plot show very similar characteristics. Most of the handoff call arrival/departure rates are observed to be less than 0.5 calls per unit time. This was due to the low user mobility rate in the trace data. Figure 4-9 shows a bivariate fit line of calculated vs. observed handoff call arrival/departure rates. The fit line passes through the origin with a slope of 1 indicating a strong correlation. Table 4-3 compares the calculated vs. observed handoff call arrival/departure rates. As the effect size, the Spearman's ρ and probability>$|\rho|$ are calculated as 1.0000 and 0, respectively, again strongly supporting our hypothesis and indicating that there is a perfect positive linear relationship between calculated and observed handoff call arrival/departure values. The Spearman's probability>$|\rho|$ being 0 demonstrates a highly significant correlation between calculated and observed handoff call arrival/departure rates.

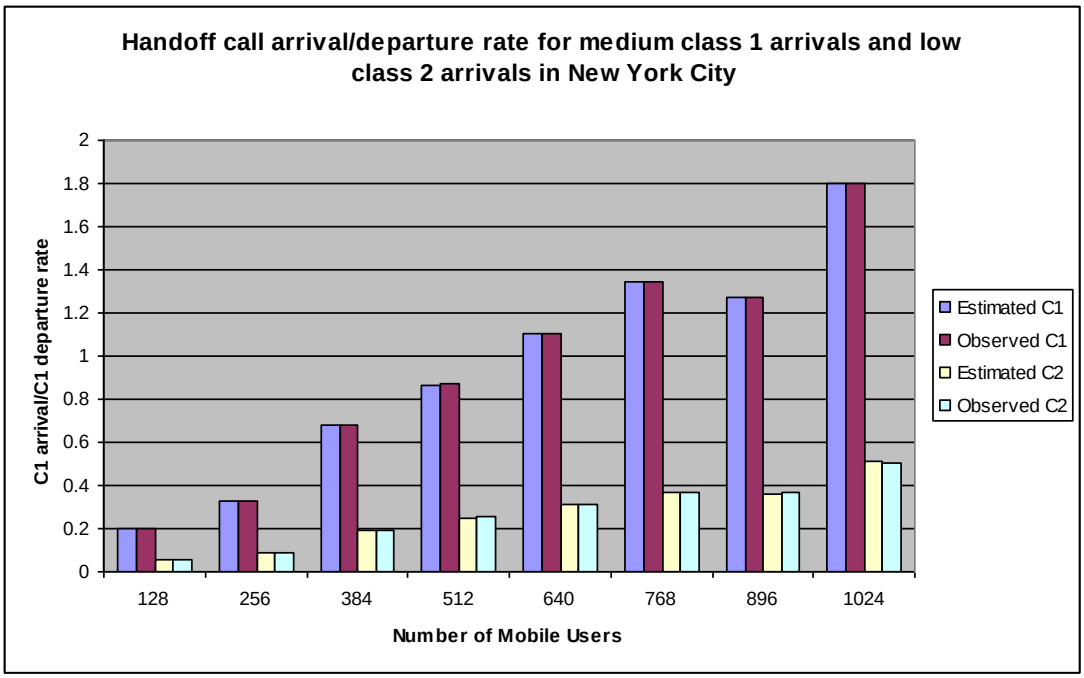

Figure 4-10: Calculated vs. Observed Handoff Call Arrival/ Departure Rates under Medium Class 1 Arrivals and Low Class 2 Arrivals in New York City.

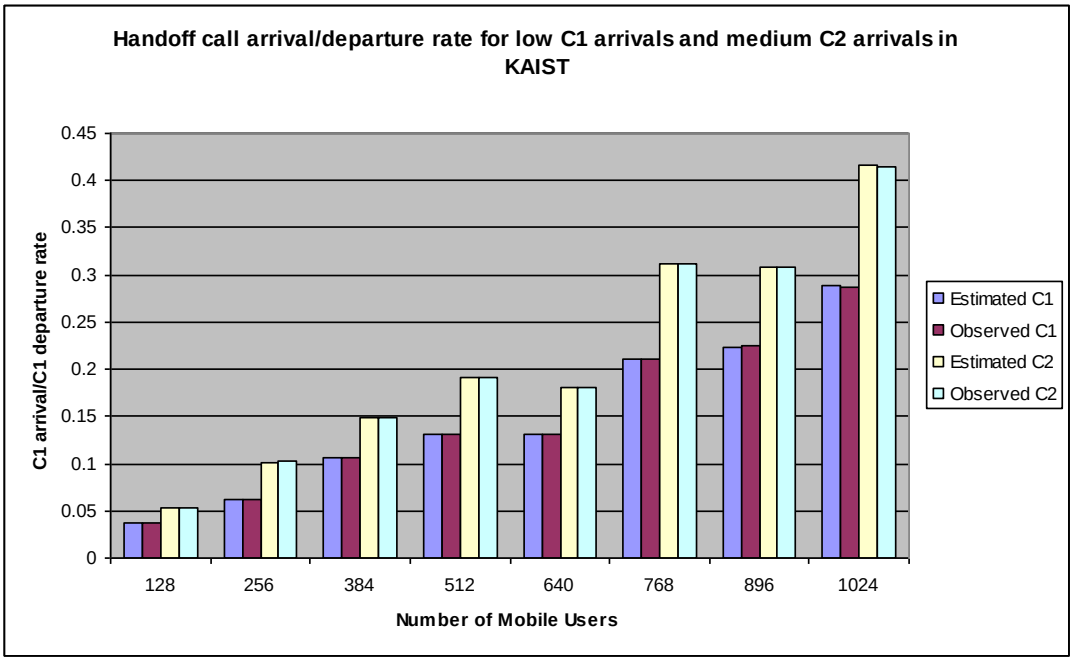

Figure 4-11: Calculated vs. Observed Handoff Call Arrival/ Departure Rates under Low Class 1 Arrivals and Medium Class 2 Arrivals in KAIST.

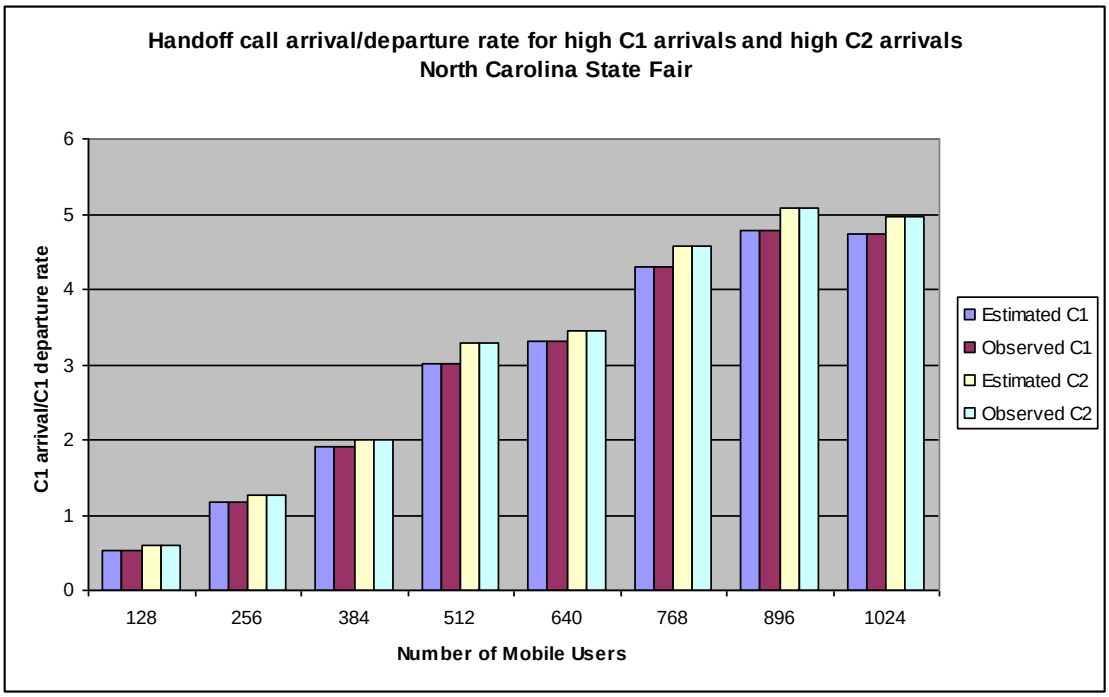

Figure 4-12: Calculated vs. Observed Handoff Call Arrival/Departure Rates under High Class 1 Arrivals and High Class 2 Arrivals in NC State Fair.

Figure 4-10, Figure 4-11 and Figure 4-12 compare calculated vs. observed handoff call arrival/departure rates for the very same three example scenarios as for Figures 4-5, 4-6 and 4-7, respectively. We see that, unlike new call arrival/departure rates, handoff call arrival/departure rates do not increase linearly with mobile user population. This is because handoff arrivals are related not only to user population but also to mobility, so the mobility paths taken by mobile users affect the distribution of handoff arrivals. Nevertheless this is indicative of the effectiveness of the workload characterization algorithm developed as it accurately calculates handoff arrival and departure rates regardless to the mobile paths taken by mobile users. Here again we

observe that in all three example scenarios, the calculated rates are virtually identical to the observed rates for both classes.

4.3.5. Sensitivity Analysis

The simulation results obtained thus far are based on the cell residence time (accounting for mobility patterns) and call duration time being exponentially distributed. Below we perform sensitivity analysis to test the sensitivity of simulation results with respect to normal, hyper-exponential and uniform time distributions for the cell residence time and call duration time. For each time distribution, we repeat the simulation covering the same 27 test scenarios with each test scenario covering 4 mobile populations at 256, 512, 768, and 1024. We again perform a nonparametric correlation analysis to show that calculated vs. observed call arrival rates are highly correlated.

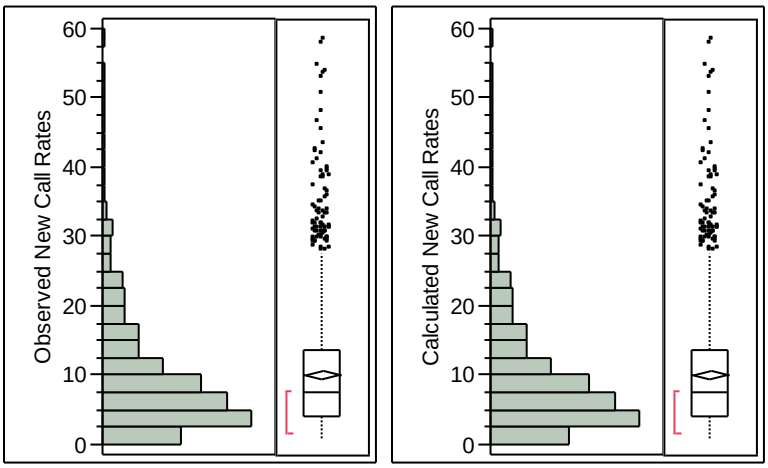

Figure 4-13: Histogram and Box Plot Diagrams of Calculated vs. Observed New Call Arrival/Departure Rates under Normal Distribution.

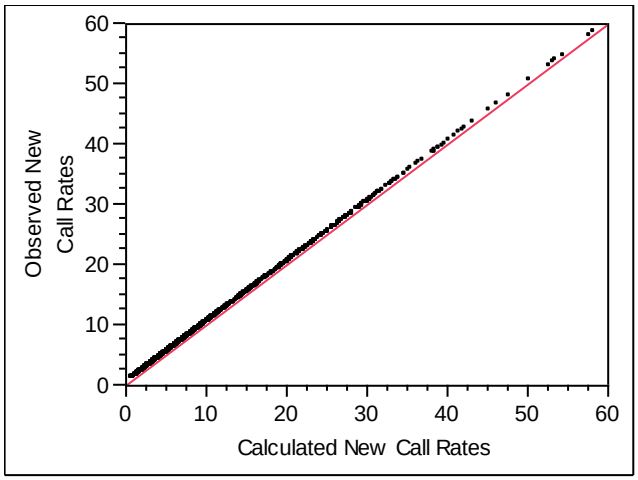

Figure 4-14: Bivariate Fit of Observed New Call arrival/Departure Rates vs. Calculated New Call Arrival/Departure Rates under Normal Distribution.

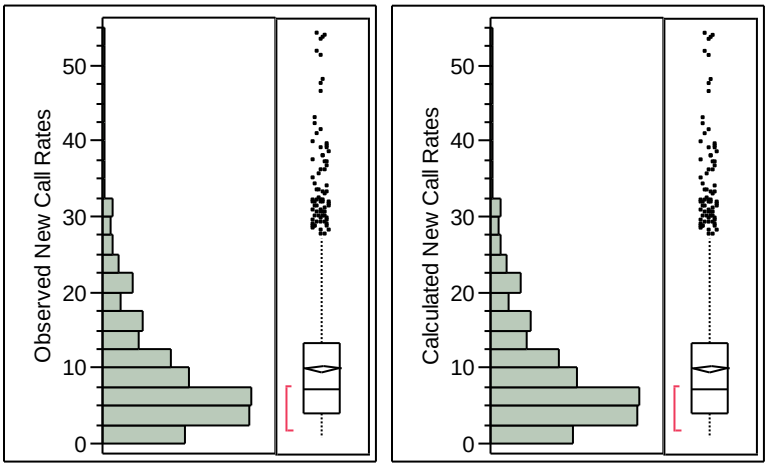

Figure 4-15: Histogram and Box Plot Diagrams of Calculated and Observed New Call Arrival/Departure Rates under Uniform Distribution.

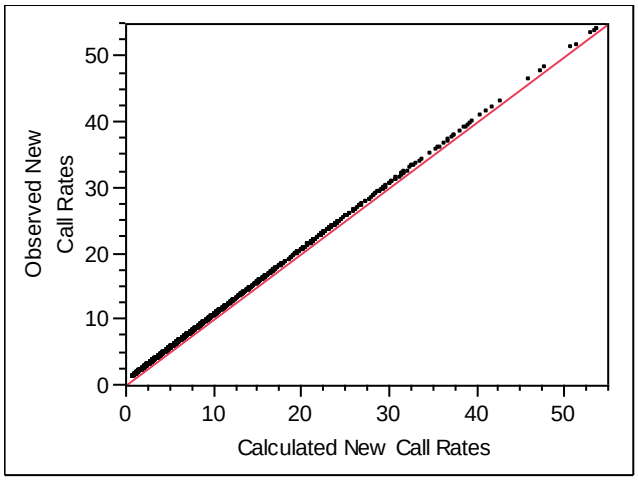

Figure 4-16: Bivariate Fit of Observed New Call arrival/Departure Rates vs. Calculated New Call Arrival/Departure Rates under Uniform Distribution.

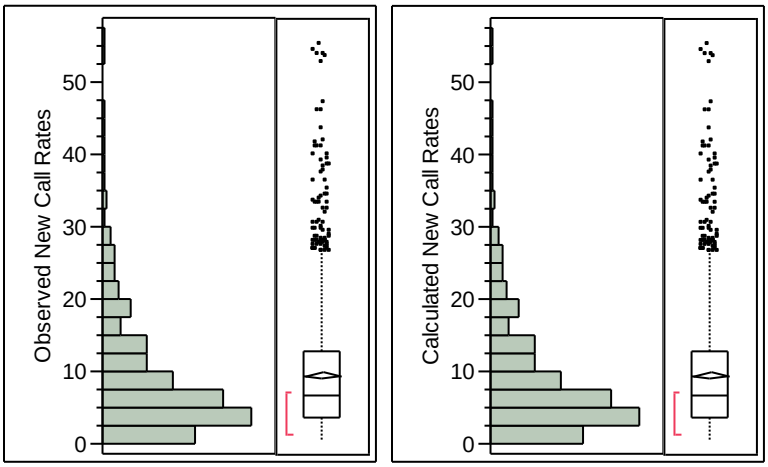

Figure 4-17: Histogram and Box Plot Diagrams of Calculated vs. Observed New Call Arrival/Departure Rates under Hyper-Exponential Distribution.

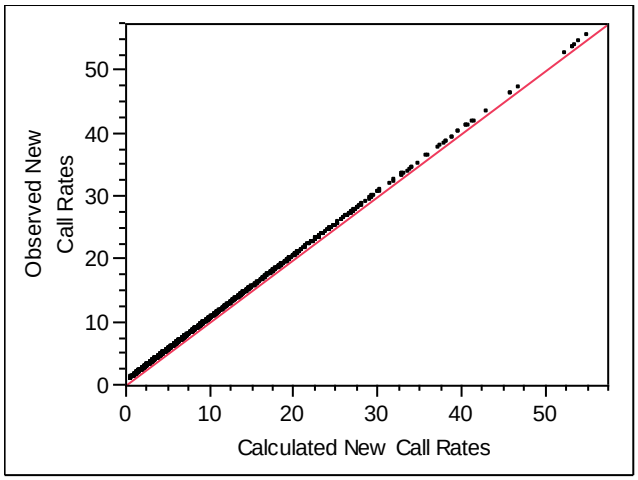

Figure 4-18: Bivariate Fit of Observed vs. Calculated New Call Arrival/Departure Rates under Hyper-Exponential Distribution.

The resulting histogram/box plot and the bivariate fit line for calculated vs. observed *new call* arrival/departure rates are shown in Figure 4-13 and Figure 4-14 for the normal distribution case (correspondingly Figure 4-15 and Figure 4-16 for the uniform distribution case, and Figure 4-17 and Figure 4-18 for the hyper-exponential distribution case). In all three cases, we see that again both the histogram and box plot show very similar characteristics. Also, the fit line again passes through the origin with a slope of 1 indicating a strong correlation. In all three cases, the Spearman's ρ and probability>$|\rho|$ are calculated as 1.0000 and 0, thus significantly supporting a perfect positive linear relationship between calculated and observed new call arrival/departure values. To avoid clutter, the corresponding histogram/box plots and the bivariate fit lines for calculated vs. observed *handoff call* rates under these three time distributions are not reported here. As for the counterpart new call arrival rates, we have observed a strong correlation for calculated vs. observed handoff call rates, In conclusion, the sensitivity analysis conducted provides strong evidences to support Hypothesis 1 and that the results are insensitive to the probability distribution of the cell residence time and call duration.

CHAPTER 5 Partitioning-Threshold Hybrid CAC Algorithms for Reward Optimization with QoS Guarantees

Traditional CAC algorithms make acceptance decisions for new and handoff calls to satisfy certain QoS constraints. We develop and analyze a class of CAC algorithms that make acceptance/rejection decisions not only to satisfy QoS constraints of multiple service classes but also to optimize the reward of the system, taking into account reward rates and arrival/departure information of service classes. In this chapter, we develop and analyze partitioning-threshold hybrid CAC based on our published work in [7][3]. In particular we present evidences to support Hypothesis 2 for partitioning-threshold hybrid CAC. In Chapter 6 we discuss spillover CAC. In Chapter 7 we discuss elastic threshold-based CAC. In chapter 8, we compare and contrast these CAC algorithms head-to-head.

5.1. Baseline CAC: Partitioning and Threshold-based CAC

For ease of presentation, we assume that there are two service types, class 1 (high-priority) and class 2 (low-priority), distinguished primarily by their traffic type, e.g., real-time vs. non-real-time. These algorithms can be easily applied to the case in which more than two service classes exist. The traffic input parameters, i.e., λ^1_n, μ^1_n, λ^1_h and μ^1_h for class 1 and λ^2_n, μ^2_n, λ^2_h and μ^2_h for class 2 are obtained by means of the workload

[3] With kind permission from Springer Science+Business Media: I.R. Chen, O. Yilmaz and I.L. Yen, "Admission control algorithms for reward optimization with QoS guarantees in mobile wireless networks," *Wireless Personal Communications*, Vol. 38, No. 3, August 2006, pp. 357-376.

characterization algorithm described in Chapter 4. The superscript in the notation denotes the class type. The cell id (*X*) in the notation is dropped as we now refer to a general cell in the system.

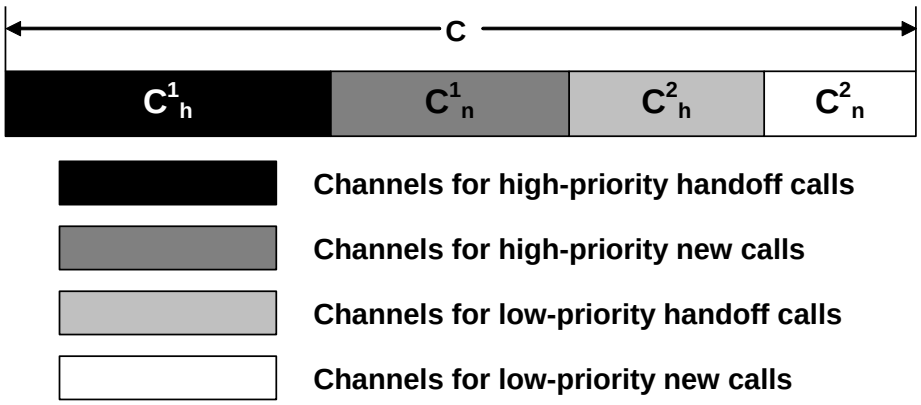

Figure 5-1: Partitioning Admission Control.

5.1.1. Partitioning CAC

A partitioning CAC policy divides the total number of channels in a cell into several fixed partitions with each partition specifically reserved to serve a particular service class (real-time vs. non-real-time) and call type (new vs. handoff). Thus for our example system there exist four partitions: high-priority handoff calls, high-priority new calls, low-priority handoff calls, and low–priority new calls, as illustrated in Figure 5-1.

By "partitioning" here we mean that a fixed number of channels are allocated to a specific service type and a call type and it cannot be used or shared by others. In calculating the expected reward we assume that we have a priori knowledge of the arrival rate of calls and the service class to which it belongs. This knowledge is essential to justify the admission of a call into a cell in order to maximize the reward of a cell at any given point of time. The network service provider normally has specified the desired

threshold blocking probability for both new and handoff calls for different service classes in order to satisfy QoS constraints.

With the scenario of two service types, the following are the input parameters to a cell: C, λ^1_h, μ^1_h, λ^1_n, μ^1_n, λ^2_h, μ^2_h, λ^2_n, μ^2_n, ν^1, ν^2, k^1, k^2, B^1_ht, B^2_ht, B^1_nt, and B^2_nt where k^1 and k^2 are the number of channels used by class 1 and class 2 calls, and B^1_ht, B^2_ht, B^1_nt, and B^2_nt are threshold blocking probability of high-priority handoff, low-priority handoff, high-priority new, and low-priority new calls, respectively.

Under the partitioning admission control algorithm, the total number of channel *C* is divided into C^1_h, C^1_n, C^2_h, and C^2_n channels for high-priority handoff calls, high-priority new calls, low-priority handoff calls, and low-priority new calls, respectively, as shown in Figure 5-1. These parameters are subject to the constraint:

$$C_h^1, C_n^1, C_h^2, C_h^2 \le C \tag{8}$$

Let (n^1_h, n^1_n, n^2_h, n^2_n) be the numbers of calls corresponding to the four fixed partitions denoted by (C^1_h, C^1_n, C^2_h, C^2_n). Then $n^1_hk^1 = C^1_h$, $n^1_nk^1 = C^1_n$, $n^2_hk^2 = C^2_h$, and $n^2_nk^2 = C^2_n$ subject to the constraint that:

$$C_h^1 + C_n^1 + C_h^2 + C_h^2 = C \tag{9}$$

The QoS constraints to be satisfied are the blocking probability of new and handoff calls for both classes 1 and 2 calls. That is, we would like to partition C channels such that the following QoS constraints are satisfied:

$$B_h^1 < B_h^1 t \tag{10}$$

$$B_n^1 < B_n^1 t \tag{11}$$

$$B_h^2 < B_h^2 t \tag{12}$$

$$B_n^2 < B_n^2 t \tag{13}$$

The reward that a successfully terminated or handed-off call brings to the cell is calculated by the product of the call's price rate parameter v^i with the duration of the call in the cell. Specifically, suppose that the partitioning algorithm reserves (C^1_h, C^1_n, C^2_h, C^2_n) channels for reward optimization, resulting in N^1_h, N^1_n, N^2_h, and N^2_n high-priority handoff calls, high-priority new calls, low-priority handoff calls, and low–priority new calls, respectively, successfully terminated or handed off per unit time in the cell. Then the cell will receive the following reward *per unit time* due to the deployment of the partitioning admission control algorithm:

$$(\frac{N_h^1}{\mu_h^1} + \frac{N_n^1}{\mu_n^1})\nu^1 + (\frac{N_h^2}{\mu_h^2} + \frac{N_n^2}{\mu_n^2})\nu^2 \tag{14}$$

Thus the optimization problem for the partitioning algorithm is to identify the best partition (C^1_h, C^1_n, C^2_h, C^2_n) that would maximize the cell's reward subject to the imposed QoS constraints defined by Conditions (10) through (13).

Under the partitioning algorithm, if a new high-priority (i.e., class 1) call arrives at a cell and all the channels allocated to serve high-priority new calls are used up, then the call is rejected. Similar reasoning applies to other service classes, too. No sharing is allowed among multiple partitions that exist. In this case the system behaves as if it is managing four concurrent queues: an $M/M/n^1_h/n^1_h$ queue to serve high-priority handoff calls with arrival rate λ^1_h, service rate μ^1_h, and the number of call slots allocated being n^1_h (such that $n^1_h k^1 = C^1_h$), an $M/M/n^1_n/n^1_n$ queue to serve new high-priority new calls in a cell with arrival rate λ^1_n, service rate μ^1_n, and the number of call slots allocated being n^1_n (such that $n^1_n k^1 = C^1_n$), an $M/M/n^2_h/n^2_h$ queue to serve low-priority handoff calls with

arrival rate λ^2_h, service rate μ^2_h, and the number of call slots being n^2_h (such that $n^2_h k^2 = C^2_h$), and an $M/M/n^2_n/n^2_n$ queue to serve low-priority new calls with arrival rate λ^2_n, service rate μ^2_n, and the number of call slots being n^2_n (such that $n^2_n k^2 = C^2_n$).

The call dropping probabilities for handoff calls for various service classes (i.e., B^1_h and B^2_h) and the blocking probability for new calls for various service classes (i.e., B^1_n and B^2_n) can be determined easily by calculating the probability of the partition allocated to serve the specific calls being full. We can calculate the reward generated per unit time by the partition reserved to serve only high-priority handoff calls by associating a reward of $i * v^1_h$ for state i in the $M/M/n^1_h/n^1_h$ queue. The same way applies to other partitions. Specifically, we can compute the reward per unit time to the cell by:

$$PR(C, \lambda^1_h, \lambda^1_n, \lambda^2_h, \lambda^2_n) = PR^1_h + PR^1_n + PR^2_h + PR^2_n \quad (15)$$

where the notation $PR(C, \lambda^1_h, \lambda^1_n, \lambda^2_h, \lambda^2_n)$ is used to stand for the reward rate earned by the partitioning algorithm as a function of $C, \lambda^1_h, \lambda^1_n, \lambda^2_h, \lambda^2_n$ (with other parameters not listed), while PR^1_h, PR^1_n, PR^2_h, and PR^2_n stand for the rewards generated per unit time due to high-priority handoff calls, high-priority new calls, low-priority handoff calls, and low-priority new calls, respectively, as given by (only PR^1_h is shown below since expressions for others are similar):

$$PR^1_h = \sum_{i=1}^{n^1_h} iv^1 \frac{\frac{1}{i!}\left(\frac{\lambda^1_h}{\mu^1_h}\right)^i}{1 + \sum_{j=1}^{n^1_h} \frac{1}{j!}\left(\frac{\lambda^1_h}{\mu^1_h}\right)^j} \quad (16)$$

A partitioning solution is "legitimate" if B^1_h, B^2_h, B^1_n and B^2_n obtained satisfy Conditions (10) through (13). A partitioning admission control integrated with pricing for

reward optimization with QoS guarantees aims to find the optimal set (C^1_h, C^1_n, C^2_h, C^2_n) that will yield the maximum reward obtained among all legitimate solutions.

5.1.2. Threshold-Based CAC

In the threshold-based CAC algorithm, we select a threshold C_T to separate class 1 from class 2 based on the service type, i.e., real-time vs. non-real time. The meaning of the threshold is that when the number of channels used in the cell exceeds C_T then new or handoff calls from service class 2 (low-priority) will not be admitted. Within each service class, we further create thresholds to differentiate handoff from new calls such that C^1_{hT} is the threshold for class 1 high-priority handoff calls; C^1_{nT} is the threshold for class 1 high-priority new calls; C^2_{hT} is the threshold for class 2 low-priority handoff calls; and C^2_{nT} is the threshold for class 2 low-priority new calls.

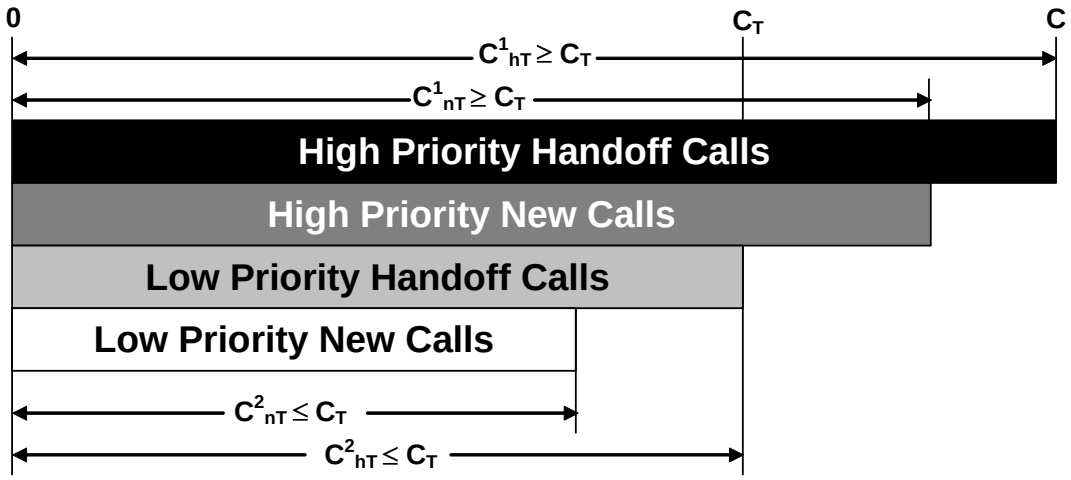

Figure 5-2: Threshold-Based Admission Control.

Figure 5-2 illustrates the threshold-based CAC algorithm. Since we give handoff calls a higher priority than new calls, the following additional conditions must also be satisfied:

$$C^1_{nT} \geq C_T,\ C^1_{hT} \geq C_T \quad (17)$$

$$C^2_{nT} \leq C_T,\ C^2_{hT} \leq C_T \quad (18)$$

A threshold-based admission control for reward optimization with QoS guarantees thus aims to find the optimal set (C^1_{hT}, C^1_{nT}, C^2_{hT}, C^2_{nT}) satisfying Conditions (17) and (18) that would yield the highest reward while satisfying the QoS constraints specified by Conditions (10) through (13).

We analyze the threshold-based admission control algorithm by using a Stochastic Petri Net (SPN) model. An SPN model is used rather than a Markov model because of the interdependency between thresholds assigned to handoff and new calls of various service classes. The SPN model adopts the idea from [39] and is generically applicable to multiple service classes. Figure 5-3 shows an SPN model for the threshold-based admission control with two service classes.

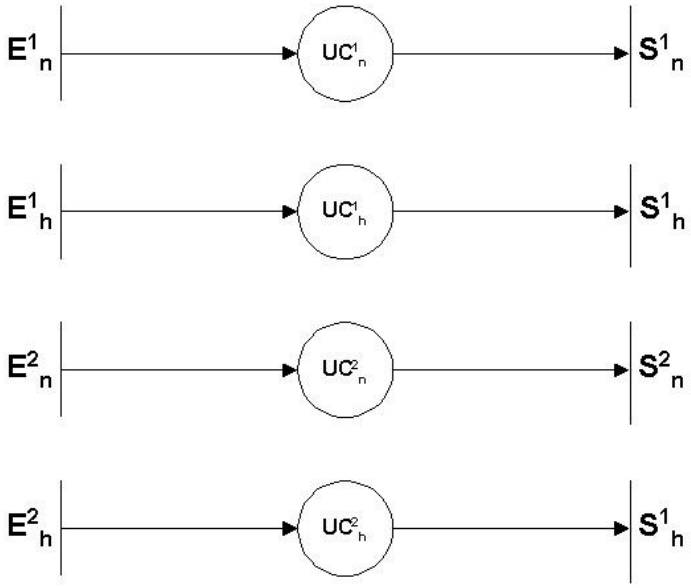

Figure 5-3: An SPN Model for Threshold-Based Admission Control with Two Service Classes.

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. The transitions and places shown in Figure 5-3 are described as follows. For *transitions*, E^i_n models new call arrivals of service class i at rate λ^i_n; E^i_h models handoff call arrivals of service class i at rate λ^i_h; S^i_n models service of new calls of service class i with a service rate of $M(UC^i_n)$ multiplied with μ^i_n where $M(UC^i_n)$ stands for the number of tokens in place UC^i_n; and S^i_h models service of handoff calls of service class i with a service rate of $M(UC^i_h)$ multiplied with μ^i_n where $M(UC^i_h)$ stands for the number of tokens in place UC^i_h. For places, UC^1_n models the execution state of service class 1 new call; UC^1_h models the execution state of service class 1 handoff calls; UC^2_n models the execution state of service class 2 new calls; and UC^2_h models the execution state of service class 2 handoff calls.

A new service request arrival is admitted only if the threshold assigned is not yet reached. Therefore we assign an enabling predicate to guard E^1_n, E^1_h, E^2_n, and E^2_h, with thresholds C^1_n, C^1_h, C^2_n, and C^2_h, respectively. Consequently, the enabling predicate of E^1_n is $[M(UC^1_n) + M(UC^1_h)]\ k^1 + k^1 + [M(UC^2_n) + M(UC^2_h)]\ k^2 \le C^1_n$. The enabling predicate of E^1_h is $[M(UC^1_n) + M(UC^1_h)]\ k^1 + k^1 + [M(UC^2_n) + M(UC^2_h)]\ k^2 \le C^1_h$. The enabling predicate of E^2_n is $[M(UC^1_n) + M(UC^1_h)]\ k^1 + k^2 + [M(UC^2_n) + M(UC^2_h)]\ k^2 \le C^2_n$. Finally, the enabling predicate of E^2_h is $[M(UC^1_n) + M(UC^1_h)]\ k^1 + k^2 + [M(UC^2_n) + M(UC^2_h)]\ k^2 \le C^2_h$.

The blocking probability B^1_n, B^1_h, B^2_n, and B^2_h are calculated from the SPN model by:

$$B_n^1 = \frac{\left(\lambda_n^1 - rate(E_n^1)\right)}{\lambda_n^1} \tag{19}$$

$$B_h^1 = \frac{\left(\lambda_h^1 - rate(E_h^1)\right)}{\lambda_h^1} \tag{20}$$

$$B_n^2 = \frac{\left(\lambda_n^2 - rate(E_n^2)\right)}{\lambda_n^2} \tag{21}$$

$$B_h^2 = \frac{\left(\lambda_h^2 - rate(E_h^2)\right)}{\lambda_h^2} \tag{22}$$

where rate(E^i_c) is calculated by finding the expected value of a random variable X defined as X= λ^i_c if E^i_c is enabled; 0 otherwise. A "legitimate" solution from a threshold admission control algorithm must generate B^1_n, B^1_h, B^2_n, and B^2_h to satisfy the QoS constraints specified by Conditions (10) through (13) discussed earlier.

We compute the reward generated per unit time from the threshold-based admission control algorithm to the cell by:

$$TR(C, \lambda^1_h, \lambda^1_n, \lambda^2_h, \lambda^2_n) = TR^1_h + TR^1_n + TR^2_h + TR^2_n \tag{23}$$

Here TR^1_h, TR^1_n, TR^2_h, and TR^2_n stand for the rewards generated per unit time due to high-priority handoff calls, high-priority new calls, low-priority handoff calls, and low-priority new calls, respectively, given by:

$$TR^i_h = (1 - B^i_h)\, \lambda^i_h\, v^i / \mu^i_h \tag{24}$$

$$TR^i_n = (1 - B^i_n)\, \lambda^i_n\, v^i / \mu^i_n. \tag{25}$$

5.2. Partitioning-Threshold Hybrid CAC

We develop partitioning-threshold hybrid CAC [7] to take advantage of both partitioning and threshold-based. The hybrid algorithm also divides the channels into fixed partitions the same way as the partitioning algorithm does. However, to take advantage of multiplexing, a "shared" partition is reserved to allow calls of all service classes/types to compete for its usage in accordance with threshold-based CAC. Figure 5-4 illustrates partition-threshold hybrid CAC. All service classes will use the shared partition based on threshold-based CAC first before they use their reserved partitions. For example, class 1 handoff calls are allowed to use the partition with channels reserved for class 1 handoff calls in the C^1_h if the number of channels already used in the shared partition (by all classes) is equal to or exceeds the threshold assigned to class 1 handoff calls.

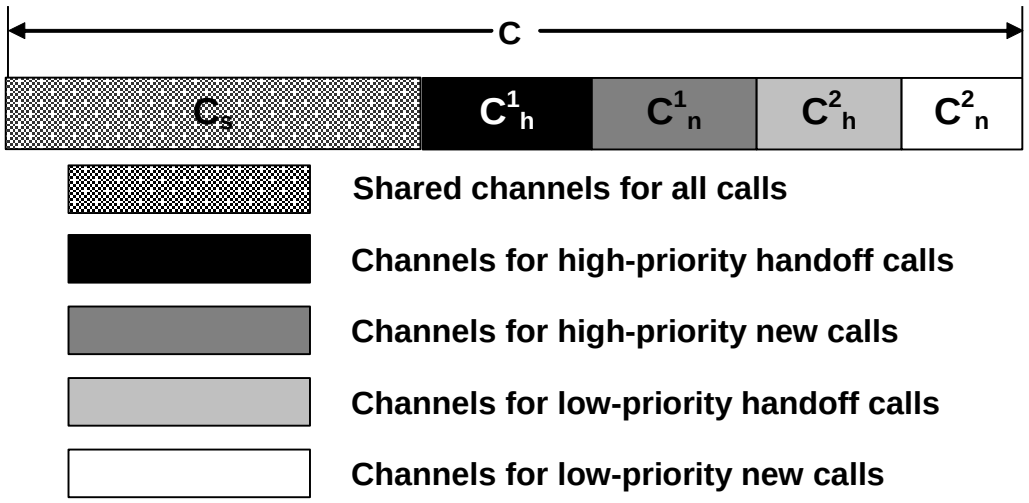

Figure 5-4: Partitioning-Threshold Hybrid CAC.

Let n^1_{hs}, n^1_{ns}, n^2_{hs}, n^2_{ns} be integer variables presenting the numbers of high-priority handoff calls, high-priority new calls, low-priority handoff calls, and low–priority new calls, respectively, in the shared partition. Let C_s be the number of channels allocated to

the shared partition under the hybrid algorithm. Then, the number of calls of various service classes and types admitted into the shared partition are limited by C_s channels allocated to the shared partition, that is,

$$n^1_{hs}k^1 + n^1_{ns}k^1 + n^2_{hs}k^2 + n^2_{ns}k^2 \leq C_s \quad (26)$$

subject to the constraint that:

$$C^1_h + C^1_n + C^2_h + C^2_n + C_s = C \quad (27)$$

The QoS constraints specified by (10) through (13) and the reward earned *per unit time* as specified by Equation (14) remain applicable to partitioning-threshold hybrid CAC.

Note that the hybrid algorithm encompasses the partitioning algorithm as a special case in which $C_s = 0$ and also the threshold-based algorithm as another special case in which C^1_h, C^1_n, C^2_h, and C^2_n are all zero. The optimization problem for the partitioning-threshold hybrid CAC algorithm is to identify the best partition (C^1_h, C^1_n, C^2_h, C^2_n, C_s) that would maximize the cell's reward subject to the imposed QoS constraints defined by Conditions (10) through (13).

5.3. Performance Model

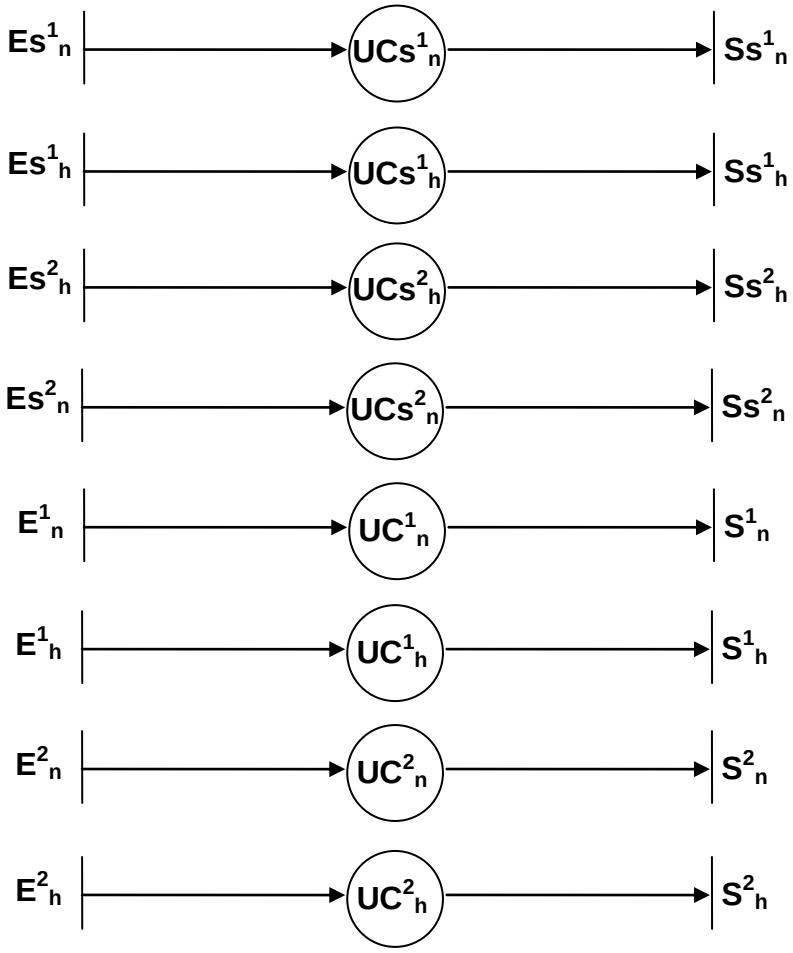

Figure 5-5: The SPN Model for Partitioning-threshold Hybrid CAC.

We develop a SPN model shown in Figure 5-5 which allows us to theoretically compute the reward rate obtainable under partitioning-threshold hybrid CAC for servicing multiple service classes with QoS guarantees. The transitions and places are described as follows. Transitions Es^i_n and E^i_n model arrival events of new calls of service class *i* at rate λ^i_n to the shared partition and to the partition reserved for class *i* new calls. Transitions Es^i_h and E^i_h model arrivals of handoff calls of service class *i* at rate λ^i_h to the shared partition and to the partition reserved for class *i* handoff calls. Transition Ss^i_n

models service of new calls of service class *i* with a service rate of $M(UCs^i_n)$ multiplied with μ^i_n where $M(UCs^i_n)$ stands for the number of tokens in place UCs^i_n; and transition Ss^i_h models service of handoff calls of service class *i* with a service rate of $M(UCs^i_h)$ multiplied with μ^i_n where $M(UCs^i_h)$ stands for the number of tokens in place UCs^i_h. Transition S^i_n models service of new calls of service class *i* with a service rate of $M(UC^i_n)$ multiplied with μ^i_n where $M(UC^i_n)$ stands for the number of tokens in place UC^i_n; and transition S^i_h models service of handoff calls of service class *i* with a service rate of $M(UC^i_h)$ multiplied with μ^i_n where $M(UC^i_h)$ stands for the number of tokens in place UC^i_h. Places UCs^1_n and UC^1_n model the execution state of service class 1 new call; UCs^1_h and UC^1_h model the execution state of service class 1 handoff calls; UCs^2_n and UC^2_n model the execution state of service class 2 new calls; and UCs^2_h and UC^2_h model the execution state of service class 2 handoff calls.

A new service request arrival is admitted to the shared partition only if the threshold assigned to the service class is not yet reached. We assign an enabling predicate to guard Es^1_n, Es^1_h, Es^2_n, and Es^2_h, with thresholds Ts^1_n, Ts^1_h, Ts^2_n, and Ts^2_h, respectively. Consequently, the enabling predicate of Es^1_n is $[M(UCs^1_n) + M(UCs^1_h)]\ k^1 + k^1 + [M(UCs^2_n) + M(UCs^2_h)]\ k^2 \leq Ts^1_n$. The enabling predicate of Es^1_h is $[M(UCs^1_n) + M(UCs^1_h)]\ k^1 + k^1 + [M(UCs^2_n) + M(UCs^2_h)]\ k^2 \leq Ts^1_h$. The enabling predicate of Es^2_n is $[M(UCs^1_n) + M(UCs^1_h)]\ k^1 + k^2 + [M(UCs^2_n) + M(UCs^2_h)]\ k^2 \leq Ts^2_n$. Finally, the enabling predicate of Es^2_h is $[M(UCs^1_n) + M(UCs^1_h)]\ k^1 + k^2 + [M(UCs^2_n) + M(UCs^2_h)]\ k^2 \leq Ts^2_h$.

A service request of class *i* that cannot be handled by the shared partition may be served by the partition reserved for servicing class *i*. Therefore, we assign an enabling

predicate to guard E^1_n, E^1_h, E^2_n, and E^2_h to partitions reserved for service class 1 new, service class 1 handoff, service class 2 new, and service class 2 handoff calls, respectively. Specifically, the enabling predicate of E^1_n is: Es^1_h is false and $M(UC^1_n)\ k^1 \leq C^1_n$. The enabling predicate of E^1_h is: Es^1_h is false and $M(UC^1_h)\ k^1 \leq C^1_h$. The enabling predicate of E^2_n is: Es^2_h is false and $M(UC^2_n)\ k^2 \leq C^2_n$. The enabling predicate of E^2_h is: Es^2_h is false and $M(UC^2_h)\ k^2 \leq C^2_h$.

The blocking probabilities B^1_n, B^1_h, B^2_n, and B^2_h may be easily calculated from the SPN model by:

$$B_n^1 = \frac{\left(\lambda_n^1 - rate\left(Es_n^1\right) - rate\left(E_n^1\right)\right)}{\lambda_n^1} \tag{28}$$

$$B_h^1 = \frac{\left(\lambda_h^1 - rate\left(Es_h^1\right) - rate\left(E_h^1\right)\right)}{\lambda_h^1} \tag{29}$$

$$B_n^2 = \frac{\left(\lambda_n^2 - rate\left(Es_n^2\right) - rate\left(E_n^2\right)\right)}{\lambda_n^2} \tag{30}$$

$$B_h^2 = \frac{\left(\lambda_h^2 - rate\left(Es_h^2\right) - rate\left(E_h^2\right)\right)}{\lambda_h^2} \tag{31}$$

where $rate(E^i_c)$ is calculated by the expected value of a random variable X defined as $X = \lambda^i_c$ if E^i_c is enabled; 0 otherwise. A "legitimate" solution must generate B^1_n, B^1_h, B^2_n, and B^2_h such that the QoS constraints specified by Conditions (10) through (13) discussed are satisfied.

From the perspective of the shared partition, the arrival rates are λ^1_h, λ^1_n, λ^2_h, and λ^2_n and the total number of channels available is C_s. Hence we compute the reward

generated per unit time from the partitioning-threshold hybrid CAC algorithm to the cell by the sum of rewards earned from all service classes, i.e., :

$$HR\left(C,\lambda_h^1,\lambda_n^1,\lambda_h^2,\lambda_n^2\right)=v^1\left[\left(1-B_h^1\right)\left(\frac{\lambda_h^1}{\mu_h^1}\right)+\left(1-B_n^1\right)\left(\frac{\lambda_n^1}{\mu_n^1}\right)\right]+v^2\left[\left(1-B_h^2\right)\left(\frac{\lambda_h^2}{\mu_h^2}\right)+\left(1-B_n^2\right)\left(\frac{\lambda_n^2}{\mu_n^2}\right)\right] \quad (32)$$

5.4. Performance Evaluation of Hybrid CAC vs. Partitioning and Threshold-based CAC

We report numerical data obtained from applying Equations (15), (23), and (32) derived for partitioning, threshold-based and hybrid CAC algorithms for reward optimization with QoS guarantees and compare their performance characteristics with physical interpretations of the results. The input parameters are C, λ^1_h, μ^1_h, λ^1_n, μ^1_n, λ^2_h, μ^2_h, λ^2_n, μ^2_n, v^1, v^2, k^1, k^2, B^1_ht, B^2_ht, B^1_nt, and B^2_nt. We set C=80, k^1=4 and k^2=1 for a typical cell in mobile wireless networks to service real-time and non-real-time traffic such that there are 80 channels in the cell with a class 1 call (real-time) consuming 4 channels and a class 2 call (non-real-time) consuming 1 channel. We vary the values of other model parameters, such as the arrival rates of new/handoff calls for different classes, pricing values (v^1 vs v^2), and threshold blocking probabilities (B^1_ht and B^2_ht) to analyze their effects on the maximum reward obtainable subject to the QoS constraints specified in terms of Conditions (10) through (13) being satisfied.

Table 5-1 compares the optimal reward obtained per unit time while the QoS constraints specified are satisfied, under partitioning, threshold-based and hybrid CAC algorithms at optimal settings as a function of the high-priority handoff call arrival rate λ^1_h, with all other parameter values being listed at the bottom of the table. The

corresponding optimal (C^1_h, C^1_n, C^2_h, C^2_n) settings under partitioning, optimal (C^1_{hT}, C^1_{nT}, C^2_{hT}, C^2_{nT}) settings under threshold-based and (C^1_h, C^1_n, C^2_h, C^2_n, C_s) settings under hybrid CAC algorithms are also listed in the table to reveal the trend exhibited in resource allocation by these algorithms. One should note that a difference of even 1 unit of reward per unit time earned by the system as a result of adopting different admission control algorithms could be considered significant because the reward accumulated over a period of time would be significant.

Table 5-1: Comparing Partitioning, Threshold-based and Hybrid CAC for Maximum Reward Obtainable while Satisfying QoS as a Function of λ^1_h.

λ^1_h	Partitioning		Hybrid		Threshold-based	
	(C^1_h, C^1_n, C^2_h, C^2_n)	Reward/Time	(C^1_h,C^1_n,C^2_h, C^2_n,C_s)	Reward/Time	(C^1_{hT},C^1_{nT}, C^2_{hT}, C^2_{nT})	Reward/Time
1	(16,56,4,4)	577.391	(8, 72,0,0,36)	580.000	(80,80,80,80)	579.95
1.5	(20,52,4,4)	615.486	(12,36,0,0,32)	620.000	(80,80,80,80)	619.88
2	(20,52,4,4)	652.304	(12,32,0,0,36)	659.997	(80,80,80,80)	659.75
2.5	(28,44,4,4)	686.660	(16,32,0,0,32)	699.986	(80,80,80,80)	699.485
3	(32,40,4,4)	717.032	(16,32,0,0,32)	739.949	(80,80,80,80)	739.023
3.5	(32,40,4,4)	754.215	(16,28,0,0,36)	779.842	(80,80,76,76)	778.258
4	None	None	(16,28,0,0,36)	819.565	(80,80,76,76)	817.058
4.5	None	None	(20,24,0,0,36)	858.998	(80,80,76,76)	855.266
5	None	None	(20,24,0,0,36)	897.974	(80,80,76,76)	892.708
5.5	None	None	(20,24,0,0,36)	936.137	(80,80,76,76)	929.203
6	None	None	(20,20,0,0,40)	973.303	(80,80,76,76)	964.569
6.5	None	None	(20,20,0,0,40)	1009.098	(80,76,75,72)	992.917
7	None	None	(24,20,0,0,36)	1043.262	None	None
7.5	None	None	(24,20,0,0,36)	1075.786	None	None

C=80, μ^1_h = 1.0, λ^1_n = 6.0, μ^1_n = 1.0, λ^2_h = 1.0, μ^2_h = 1.0, λ^2_n = 1.0, μ^2_n = 1.0, v^1 = 80, v^2 = 10, k^1 = 4, k^2 = 1, $B^1_h t$ = 0.02, $B^2_h t$ = 0.04, $B^1_n t$ = 0.05, $B^2_n t$ = 0.1.

The data in Table 5-1 indicate that as λ^1_h increases, the reward rate obtainable also increases as long as the QoS constraints can still be satisfied given the amount of resources available (C=80). Nevertheless, as λ^1_h increases further past a threshold value, all algorithms eventually fail to yield a legitimate solution because the workload is too heavy to satisfy the imposed QoS constraints, as indicated in the table by "None". One can see that hybrid CAC is the most tolerant algorithm among all in terms of being able

to yield a solution under high workload situations, followed by threshold-based and partitioning.

We see that in response to a high arrival rate of λ^1_h, hybrid CAC (in the middle column) tends to increase the size of two partitions, that is, it tends to increase C^1_h to satisfy the stringent QoS constraint of $B^1_ht = 0.02$ for class 1 handoff calls, and it also tends to increase Cs to exploit the multiplexing power of the shared partition by means of threshold-based admission control to satisfy QoS constraints of all other service calls. The multiplexing power of the shared partition is clearly demonstrated by the fact that hybrid always significantly outperforms partitioning in terms of reward obtainable over a range of λ^1_h values, while being able to sustain a higher workload of λ^1_h and provide QoS guarantees. As the arrival rate of class 1 handoff calls increases, on the other hand, threshold-based admission (in the last column) control tends to decrease the threshold values of C^1_{nT}, C^2_{hT} and C^2_{nT} while keeping C^1_{hT} as high as possible to satisfy the stringent QoS constraint of $B^1_ht = 0.02$. We observe that the performance of threshold-based admission control is comparable to hybrid CAC until λ^1_h becomes high enough, beyond which threshold-based performs worse and eventually fails to yield a legitimate solution compared with hybrid CAC. We attribute the superiority of hybrid CAC over partitioning and threshold-based admission control to the ability to optimally reserve dedicated resources for high-priority classes through fixed partitioning to reduce interference from low-priority classes, and to optimally allocate resources to the shared partition in accordance with threshold-based admission control to exploit the multiplexing power for all classes.

Table 5-2: Comparing Partitioning, Threshold-based and Hybrid CAC for Maximum Reward Obtainable while Satisfying QoS as a Function of λ^2_h and λ^2_n.

λ^2_h&λ^2_n	Partitioning		Hybrid		Threshold-based	
	(C^1_h, C^1_n, C^2_h, C^2_n)	Reward/Time	(C^1_h,C^1_n,C^2_h, C^2_n,C_s)	Reward/ Time	(C^1_{hT},C^1_{nT}, C^2_{hT}, C^2_{nT})	Reward/Time
1	(48,24,4,4)	576.382	(32,16,0,0,32)	580.000	(80,80,80,80)	579.952
2	(44,24,6,6)	594.268	(28,12,0,0,40)	599.999	(80,80,80,80)	599.917
3	(44,20,8,8)	610.326	(28,12,0,0,40)	619.998	(80,80,80,80)	619.855
4	(44,20,8,8)	628.380	(28,12,1,1,38)	639.993	(80,80,80,80)	639.755
5	(40,20,10,10)	644.636	(28,12,2,2,36)	659.977	(80,80,80,80)	659.593
6	(40,20,11,9)	660.886	(24,8,2,2,44)	679.937	(80,80,80,80)	679.338
7	None	None	(24,8,2,2,44)	699.854	(80,80,80,80)	698.948
8	None	None	(24,8,3,3,42)	719.675	(80,80,80,80)	718.365
9	None	None	(20,8,3,3,46)	739.321	(80,80,76,76)	737.525
10	None	None	(20,8,3,3,46)	758.708	(80,80,76,76)	756.341
11	None	None	(20,8,4,4,44)	777.650	(80,80,76,76)	774.714
12	None	None	(20,4,3,3,50)	795.995	(80,80,76,76)	792.533
13	None	None	(16,4,3,3,54)	813.654	(80,80,76,76)	809.685
14	None	None	(16,4,3,3,54)	830.339	(80,80,76,76)	826.054
15	None	None	(16,4,3,3,54)	845.795	(80,80,76,76)	841.533
16	None	None	(16,4,6,6,48)	859.543	(80,80,76,75)	855.576
17	None	None	(12,0,2,2,64)	872.773	None	None
C=80, λ^1_h = 5.0, μ^1_h = 1.0, λ^1_n = 2.0, μ^1_n = 1.0, μ^2_h = 1.0, μ^2_n = 1.0, v^1 = 80, v^2 = 10, k^1 = 4, k^2 = 1, $B^1_h t$ = 0.02, $B^2_h t$ = 0.04, $B^1_n t$ = 0.05, $B^2_n t$ = 0.1.						

Next we test the sensitivity of the results with respect to the traffic load of class 2 (low-priority) calls. Table 5-2 shows the reward rate earned by the system as a function of λ^2_h and λ^2_n. Here we see that as the arrival rate of the low-priority class increases (shown in the first column), hybrid CAC (shown in the middle column) tends to decrease the number of dedicated channels allocated to high-priority calls, while at the same increasing the number of shared channels to exploit the multiplexing power in the shared partition. The reason is that as the arrival rate of low-priority calls increases the system will gain most of its reward from low-priority calls. Thus hybrid CAC tempts to allocate as much resources to low-priority calls as possible, to the extent that QoS constraints for both high and low priority calls are satisfied. Since the QoS constraint of high priority handoff calls is stringent (2% drop probability), we see from Table 5-2 that even when

the arrival rate of low-priority class is very high, hybrid CAC still allocates some designated channels to the C^1_h partition. In all cases we observe that hybrid CAC performs the best among all over a wide range of arrival rate of low-priority calls.

Two other important parameters affect the behavior of admission control for reward optimization with QoS guarantees. One is the ratio of v^1: v^2; the other is QoS constraints. We present sensitivity analysis of these two parameters below.

Table 5-3: Comparing Partitioning, Threshold-based and Hybrid CAC for Maximum Reward Obtainable while Satisfying QoS as a Function of v^1 to v^2 Ratio.

v^1: v^2	Partitioning		Hybrid		Threshold-based	
	(C^1_h, C^1_n, C^2_h, C^2_n)	Reward/Time	(C^1_h,C^1_n,C^2_h, C^2_n,C_s)	Reward/ Time	(C^1_{hT},C^1_{nT}, C^2_{hT}, C^2_{nT})	Reward/ Time
Low Class 1 Call Arrival Rates (λ^1_h = 1.0, λ^1_n = 1.0)						
1	(20, 16, 22, 22)	219.735	(12,12,5,5,46)	220.000	(80,80,80,80)	220.000
2	(20, 16, 22, 22)	239.550	(12,12,5,5,46)	240.000	(80,80,80,80)	240.000
4	(20, 20, 20, 20)	279.381	(16,12,5,5,42)	280.000	(80,80,80,80)	280.000
8	(20, 20, 20, 20)	359.135	(16,12,5,5,42)	360.000	(80,80,80,80)	360.000
16	(20, 20, 20, 20)	518.645	(16,16,7,5,36)	520.000	(80,80,80,80)	520.000
32	(20, 20, 20, 20)	837.663	(16,16,7,5,36)	840.000	(80,80,76,76)	840.000
64	(24, 20, 18, 18)	1476.281	(16,16,7,5,36)	1480.000	(80,80,76,76)	1480.000
128	(24, 24, 16, 16)	2754.232	(16,16,7,5,36)	2760.000	(80,80,76,76)	2760.000
High Class 1 Call Arrival Rates (λ^1_h = 3.5, λ^1_n = 4.5)						
1	None	None	(8,12,5,5,50)	278.919	(80,80,80,80)	278.280
2	None	None	(12,16,4,4,44)	358.240	(80,80,80,80)	357.129
4	None	None	(12,16,3,3,46)	516.987	(80,80,80,80)	514.828
8	None	None	(12,16,2,2,48)	834.545	(80,80,76,76)	830.611
16	None	None	(12,16,1,1,50)	1469.720	(80,80,72,72)	1464.84
32	None	None	(12,16,0,0,52)	2747.443	(80,80,72,69)	2736.79
64	None	None	(12,16,0,0,52)	5303.173	(80,80,71,66)	5284.15
128	None	None	(12,16,0,0,52)	10416.435	(80,80,71,66)	10380.50
C=80, μ^1_h = 1.0, μ^1_n = 1.0, λ^2_h = 10.0, μ^2_h = 1.0, λ^2_n = 10.0, μ^2_n = 1.0, v^2 = 10, k^1 = 4, k^2 = 1, $B^1_h t$ = 0.02, $B^2_h t$ = 0.04, $B^2_n t$ = 0.05, $B^2_n t$ = 0.1.						

Table 5-3 shows the effect of v^1: v^2 (by varying v^1 while setting v^2 to 10). The results show that hybrid CAC outperforms or is at least as good as partitioning and threshold-based admission control. Moreover, we observe that the difference in reward earned becomes more significant as the v^1: v^2 ratio increases. This effect is especially pronounced when the system is heavily loaded (shown in the bottom half of Table 5-3) under which it is necessary to optimally allocate channels to calls of different priority

types and service charge rates to maximize the reward earned by the system while at the same time satisfying the imposed QoS constraints.

Table 5-4: Comparing Partitioning, Threshold-based and Hybrid CAC for Maximum Reward Obtainable while Satisfying QoS as a Function of QoS Constraints on Class 1 and Class 2 Handoff Calls.

(B^1_ht, B^2_ht)	Partitioning		Hybrid		Threshold-based	
	$(C^1_h, C^1_n, C^2_h, C^2_n)$	Reward/ Time	$(C^1_h, C^1_n, C^2_h, C^2_n, C_s)$	Reward/ Time	$(C^1_{hT}, C^1_{nT}, C^2_{hT}, C^2_{nT})$	Reward/ Time
Low Class 1 Call Arrival Rates (λ^1_h = 1.0, λ^1_n = 1.0)						
(0.02,0.04) x 2^0	(20,20,20,20)	359.135	(16,12,5,5,42)	360.000	(80,80,80,80)	359.999
(0.02,0.04) x 2^{-1}	(20,20,20,20)	359.135	(16,12,5,5,42)	360.000	(80,80,80,80)	359.999
(0.02,0.04) x 2^{-2}	(20,20,20,20)	359.135	(16,12,5,5,42)	360.000	(80,80,80,80)	359.999
(0.02,0.04) x 2^{-3}	(24,20,20,20)	358.345	(16,12,5,5,42)	360.000	(80,80,80,80)	359.999
(0.02,0.04) x 2^{-4}	(24,20,20,20)	358.345	(16,12,5,5,42)	360.000	(80,80,80,80)	359.999
(0.02,0.04) x 2^{-5}	(24,20,21,19)	358.264	(16,12,5,5,42)	360.000	(80,80,80,80)	359.999
(0.02,0.04) x 2^{-6}	(28,16,22,14)	353.041	(16,12,5,5,42)	360.000	(80,80,80,80)	359.999
(0.02,0.04) x 2^{-7}	(28,16,23,13)	350.311	(16,12,5,5,42)	360.000	(80,80,80,80)	359.999
…	None	None	…	…	…	…
(0.02,0.04) x 2^{-21}	None	None	(16,12,5,5,42)	360.000	(80,76,76,61)	359.991
(0.02,0.04) x 2^{-22}	None	None	(16,12,5,5,42)	360.000	(80,76,76,54)	359.904
(0.02,0.04) x 2^{-23}	None	None	(16,12,5,5,42)	360.000	(80,76,76,48)	359.409
(0.02,0.04) x 2^{-24}	None	None	(16,12,5,5,42)	360.000	(80,76,76,42)	357.231
(0.02,0.04) x 2^{-25}	None	None	(16,12,5,5,42)	360.000	None	None
High Class 1 Call Arrival Rates (λ^1_h = 3.5, λ^1_n = 4.5)						
(0.02,0.04) x 2^0	None	None	(12,16,2,2,48)	834.544	(80,80,76,76)	830.610
(0.02,0.04) x 2^{-1}	None	None	(12,16,2,2,48)	834.544	(80,80,76,76)	830.610
(0.02,0.04) x 2^{-2}	None	None	(20,8,1,1,50)	830.078	(80,76,76,76)	826.208
C=80, μ^1_h = 1.0, μ^1_n = 1.0, λ^2_h = 10.0, μ^2_h = 1.0, λ^2_n = 10.0, μ^2_n = 1.0, v^1 = 80, v^2 = 10, k^1 = 4, k^2 = 1, B^1_nt=0.05, B^2_nt=0.1.						

Table 5-4 shows the effect of QoS constraints. We tighten the QoS constraints of handoff calls for both classes 1 and 2 (B^1_ht, and B^2_ht) by a multiplicative factor of 2 successively to see how these admission control algorithms would respond to the change. We observe that under light-load conditions (the first half of Table 5-4) all three algorithms can reasonably adapt to the QoS change in order to satisfy the QoS constraints. However, partitioning admission control generates relatively lower reward because without multiplexing power it needs to trade reward off for QoS satisfaction. When the QoS constraints of handoff calls become extremely tight, both partitioning and

threshold-based admission control algorithms fail to provide a legitimate solution, while hybrid admission is still able to provide a legitimate solution due to its ability to exploit the multiplexing power in the shared partition and to reserve dedicated resources for individual service classes.

The adaptability of hybrid CAC with respect to more stringent QoS constraints is especially pronounced under heavy-load situations (shown at the bottom half of Table 5-4). In response to more stringent QoS constraints on handoff calls under heavy-load conditions, hybrid CAC allocates more channels in the C^1_h partition and conversely fewer channels in the C^1_n partition to satisfy the most stringent QoS constraint imposed on class 1 handoff calls. Further, it also allocates more channels in the shared partition to satisfy the stringent QoS constraint of class 2 handoff calls, which through multiplexing also has the benefit of compensating class 1 and class 2 new calls to satisfy their QoS constraints.

5.5. Supporting Hypothesis 2 for Partitioning-Threshold Hybrid CAC

We see that the analytical results presented in Section 5.2 support Hypothesis 2a and Hypothesis 2b for partitioning-threshold hybrid CAC. The arrival and departure rates of calls, however, were synthetic. In the dissertation research, we further validate analytical results by means of simulation using real human mobility data. We have performed extensive simulation and tested the hybrid CAC algorithm with 27 test scenarios with varying aggregate class 1 and class 2 new/handoff call arrival and departure rates as a function of mobile user population based on the simulation environment setup described in Section 4.3.

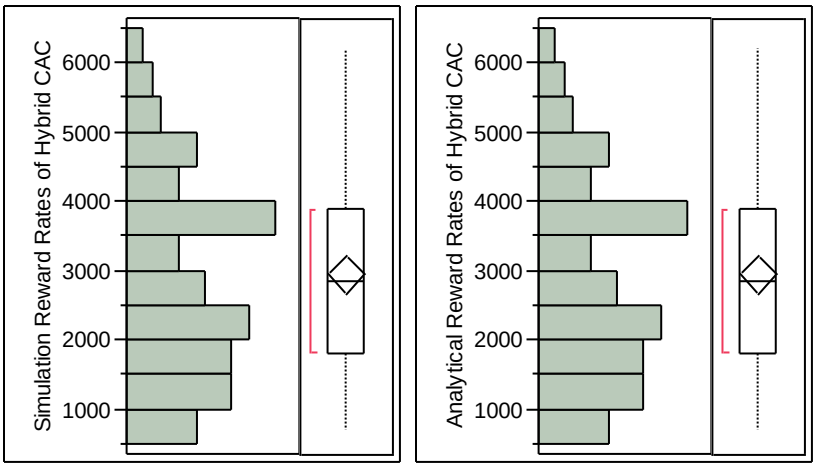

Figure 5-6: Histogram and Box Plot Diagrams of Simulated vs. Analytical Arrival/Departure Rates of Hybrid CAC.

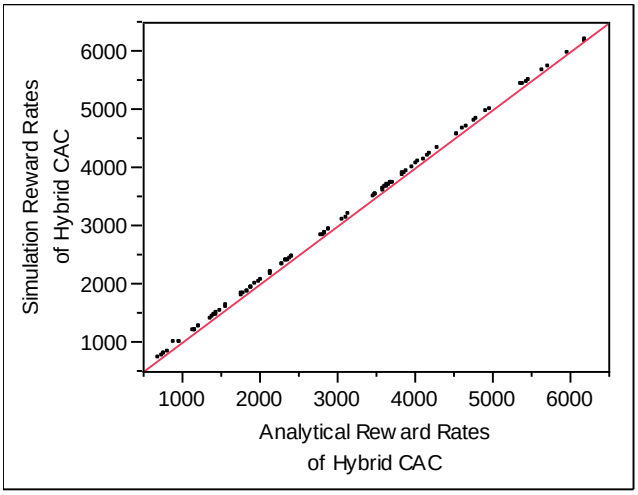

Figure 5-7: Bivariate Fit of Simulated vs. Analytical Reward Rates of Hybrid CAC.

Table 5-5: Comparison of Analytical vs. Simulation Results for Partitioning-Threshold Hybrid CAC.

Hybrid CAC	Simulation	Analytical	Absolute Difference	Percentage Difference
Number of observations	102	102	0	0%
Mean	2943.6	2944.9	1.29	-0.046%
Median	2841.3	2842	0.035	0.001%
Standard Deviation	1430.8	1436.3	11.60	0.819%
Range	5475.6	5506.4	122.28	0.808%
Midspread	2081.8	2083.2	1.36	0.054%

We have performed a nonparametric correlation analysis to validate analytical results with simulation results collected. Figure 5-6 presents the histogram and box plot diagrams of simulation vs. analytical reward rates. Both the histogram and box plot show very similar characteristics. Both histograms have a mode value at 4000 while the mean and median reward rates are around 2900. This is due to the fact that our test cases generate reward rates which do not necessarily follow a specific distribution. Figure 5-7 shows a bivariate fit line of simulation vs. analytical reward rates. The fit line passes through the origin with a slope of 1 indicating a strong correlation. Table 5-5 compares analytical results vs. simulation results for partitioning-threshold hybrid CAC. As for the effect size, the Spearman's ρ and probability>|ρ| are calculated as 0.9998 and 0.001, respectively, thereby supporting our hypothesis strongly and indicating that there is a perfect positive linear relationship significantly between simulation and analytical reward values.

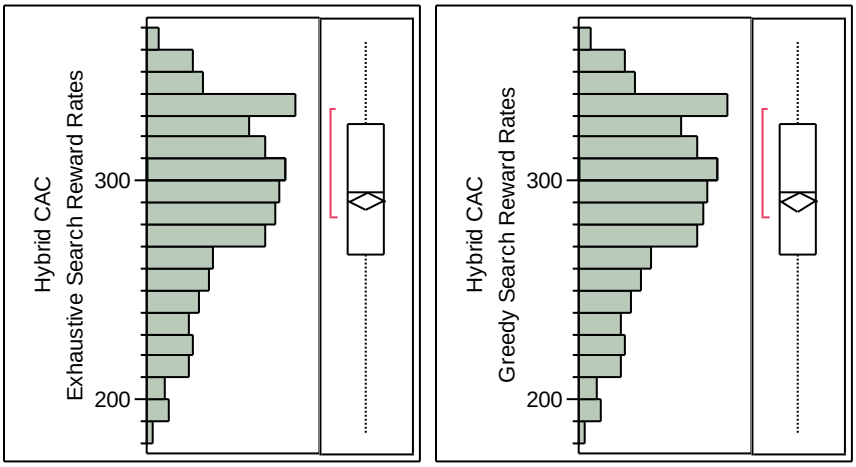

Figure 5-8: Histogram and Box Plot Diagrams of Exhaustive vs. Greedy Search Reward Rates of Hybrid CAC.

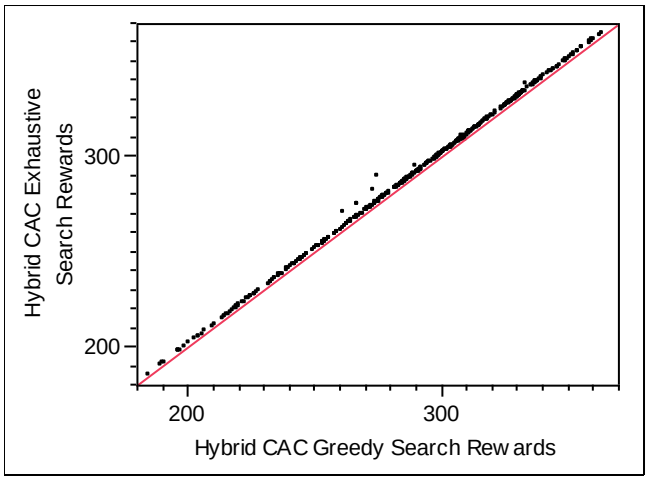

Figure 5-9: Bivariate Fit of Hybrid CAC Exhaustive Search Reward Rates vs. Hybrid CAC Greedy Search Reward Rates.

Table 5-6: Comparison of Reward Rates Obtainable by Greedy vs. Exhaustive Search based Hybrid CAC.

Hybrid CAC	Exhaustive	Greedy	Absolute Difference	Percentage Difference
Number of observations	415	415	0	0%
Mean	291.1	290.9	-0.12	-0.00046%
Median	294.7	294.7	0	0%
Standard Deviation	40.3	40.3	1.05	0.0039%
Range	179.0	179.0	14.17	0.052%
Midspread	59	59.4	0	0%

We have designed partitioning-threshold hybrid CAC with search efficiency considerations without compromising search optimality. To test this hypothesis, we generate a large number of test scenarios (1000) randomly to simulate random call arrival and departure rates of service classes with QoS constraints with the objective to evaluate solution optimality of exhaustive search vs. partitioning-threshold hybrid CAC. Table 5-6 compares reward rates obtainable by heuristic based vs. exhaustive search based hybrid CAC. We observe that both partitioning-threshold hybrid CAC algorithms generate exactly the same number of legitimate solutions (415) out of the 1000 test scenarios.

We further perform a nonparametric correlation analysis to compare the results found by hybrid CAC vs. those by exhaustive search. Figure 5-8 presents the histogram and box plot diagrams of reward rates calculated via exhaustive search vs. greedy search. Both the histogram and box plot present very similar characteristics. Both histograms have a mode value at 330 while the mean and median reward rates are around 290. Figure 5-9 shows a bivariate fit line of rewards calculated via greed search vs. exhaustive search. The fit line passes through the origin with a slope of 1 indicating a strong correlation. As the effect size, the Spearman's ρ and probability>$|\rho|$ are calculated as 0.9998 and 0.001, respectively, thereby supporting our hypothesis strongly and indicating that there is a perfect positive linear relationship significantly between reward rates calculated via exhaustive search vs. greedy search.

CHAPTER 6 Spillover Call Admission Control Algorithms for Reward Optimization with QoS Guarantees

To improve solution efficiency of partitioning-threshold hybrid CAC, we develop and analyze partitioning-based spillover CAC in this chapter. This Chapter is based on the work published in [71][4]. We will compare and contrast spillover CAC vs. partitioning-threshold hybrid CAC later in Chapter 8.

6.1. Spillover CAC

For ease of exposition, we again assume that two service classes exist with class 1 being the high-priority class. This new CAC algorithm developed is based on partitioning and is "spillover" in the sense that if a service call cannot be admitted into P_i (partition i) then it will overflow to P_{i+1} and so on.

[4] © 2008 IEEE. Reprinted, with permission, from O. Yilmaz, I.R. Chen, G. Kulczycki, and B. Frakes "Spillover call admission control for reward optimization with QoS guarantees for multiple service classes in mobile wireless networks," *IEEE 3rd International Workshop on Performance Analysis and Enhancement of Wireless Networks*, AINA Workshops, Okinawa, Japan, March 2008, pp. 1299 - 1304

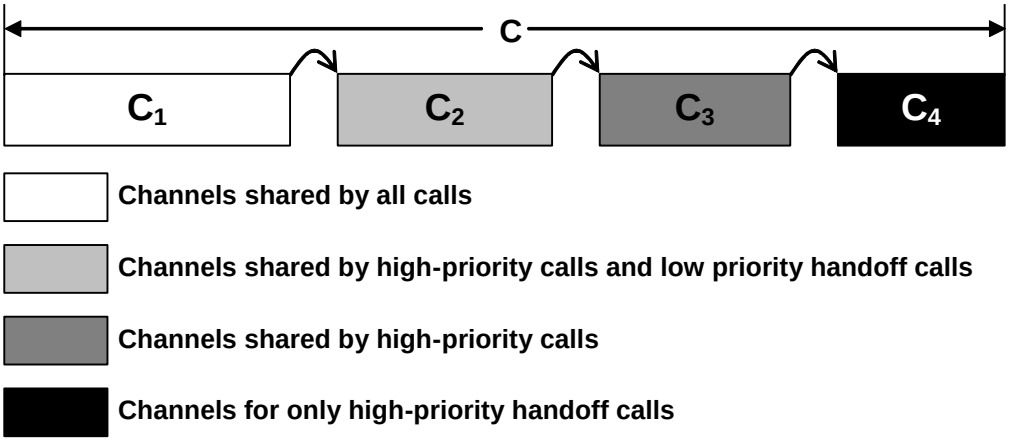

Figure 6-1: Spillover Call Admission Control.

Figure 6-1 illustrates these four partitions allocated by the spillover CAC algorithm. There are four partitions, P_1, P_2, P_3 and P_4 allocated with C_1, C_2, C_3 and C_4 channels, respectively. The system channels are partitioned such that P_1 accepts all service call types; P_2 accepts all service call types except the service call type with the least stringent QoS constraints and so on. When a call is received, it is put into the very first partition that can accept this call. In our example system, we reserve 4 partitions P_1, P_2, P_3 and P_4, with P_1 accepting all calls, P_2 accepting all calls except for class 2 new calls, P_3 accepting all calls except for class 2 new and handoff calls, and P_4 only accepting class 1 handoff calls. When a class 1 handoff call is received, P_1 is used unless this partition is full. If this partition is full then P_2, P_3 or P_4 will be tested to accept the call in this order.

Spillover CAC, by the virtue of allocating *several* partitions to high-priority classes, can satisfy stringent constraints of high-priority calls. It improves utilization by sharing certain partitions among different service call types. (e.g., P_1 is shared by all service call types). Due to the use of partitioning rather than thresholds, it also greatly reduces computational complexity. Finally, by letting multiple service call types share

most of the partitions, it produces optimal solutions. To satisfy the imposed QoS constraints, the spillover partitioning algorithm looks for "legitimate" 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 Conditions (10) through (13).

More formally, let $c\_min^i_{n/h}$ denote the minimum number of channels needed to satisfy the new/handoff call QoS constraints for class *i*. We first sort all service classes by $c\_min^i_{n/h}$ to determine the *service order*. For example if the order is $c\_min^1_h \geq c\_min^1_n \geq c\_min^2_h \geq c\_min^2_n$, 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. In our example system in which there are two classes, we will have 4 partitions P_1, P_2, P_3, and P_4. The first partition would be reserved for class 1 handoff calls only; the second partition would be for class 1 calls including both handoff and new calls; the third partition would be for class 1 calls and class 2 handoff calls; and the last partition would be open to all call types.

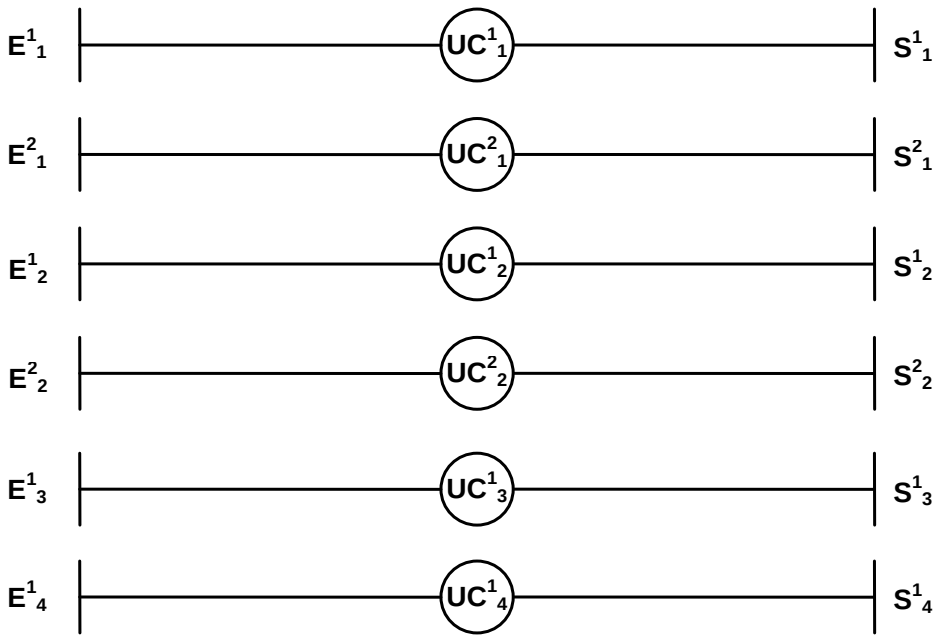

Figure 6-2: SPN Model for Spillover Admission Control.

6.2. Performance Model

We develop a mathematic model based on Stochastic Petri nets as shown in Figure 6-2 which allows us to theoretically calculate the reward obtainable under spillover CAC for servicing multiple service classes with QoS guarantees in terms of the blocking/dropping probabilities of service classes.

In Figure 6-2, place UC^i_1 holds service class *i* new and handoff calls in partition 1, and $M(UC^i_1)$ represents the number of class *i* calls it holds. Transition E^i_1 models arrivals of class *i* new and handoff calls at rate $\lambda_h^i\left(\frac{\mu_n^i}{\mu_h^i}\right)+\lambda_n^i$ to partition 1. S^i_1 models departures of class *i* calls with a service rate of $M(UC^i_1)$ multiplied with per-call service

rate μ^i_n; place UC^1_2 holds service class 1 new and handoff calls in partition 2, and $M(UC^1_2)$ represents the number of class 1 calls it holds.

Transition E^1_2 models arrivals of class 1 new and handoff calls at rate $\lambda_h^1\left(\frac{\mu_n^1}{\mu_h^1}\right)+\lambda_n^1$ to partition 2. S^1_2 models departures of class 1 calls with a service rate of $M(UC^1_2)$ multiplied with per-call service rate μ^1_n; place UC^2_2 holds service class 2 handoff calls in partition 2, and $M(UC^2_2)$ represents the number of class 2 handoff calls it holds. Transition E^2_2 models arrivals of class 2 handoff calls at rate λ_h^2 to partition 2. S^2_2 models departures of class 2 handoff calls with a service rate of $M(UC^2_2)$ multiplied with per-call service rate μ^2_h; place UC^1_3 holds service class 1 new and handoff calls in partition 3, and $M(UC^1_3)$ represents the number of class 1 calls it holds. Transition E^1_3 models arrivals of class 1 new and handoff calls at rate $\lambda_h^1\left(\frac{\mu_n^1}{\mu_h^1}\right)+\lambda_n^1$ to partition 3. S^1_3 models departures of class 1 calls with a service rate of $M(UC^1_3)$ multiplied with per-call service rate μ^1_n; place UC^1_4 holds service class 1 handoff calls in partition 4, and $M(UC^1_4)$ represents the number of class 1 handoff calls it holds. Transition E^1_4 models arrivals of class 1 handoff calls at rate λ_h^1 to partition 4. S^1_4 models departures of class 1 handoff calls with a service rate of $M(UC^1_4)$ multiplied with per-call service rate μ^1_h.

A new service request arrival is admitted to P_1 only if the partition has enough channels to accommodate the new request. Therefore, we assign enabling predicates to guard E^1_1 and E^2_1 with C_1 being the constraint. Specifically, the enabling predicate of E^i_1 is $M(UC^1_1)\ k^1 + M(UC^2_1)\ k^2 + k^i \leq C_1$. A new service request of a class 1 call or a class 2 handoff call is admitted into P_2 only if the P_2 but not P_1 has enough channels to

accommodate the new request. Thus, we assign enabling predicates to guard E^1_2 and E^2_2 with C_2 being the constraint. Specifically, the enabling predicate of E^i_2 is $M(UC^1_2)\ k^1 + M(UC^2_2)\ k^2 + k^i \leq C_2$. A new service request of a class 1 call is admitted to P_3 only if the P_3 but not P_1 or P_2 has enough channels to accommodate the new request. Thus, we assign enabling predicates to guard E^1_3 with C_3 being the constraint. Specifically, the enabling predicate of E^i_3 is $M(UC^1_3)\ k^1 + k^1 \leq C_3$. A new service request of a class 1 handoff call is admitted to P_4 only if the P_4 but not P_1, P_2 or P_3 has enough channels to accommodate the new request. Thus, we assign enabling predicates to guard E^1_4 with C_4 being the constraint. Specifically, the enabling predicate of E^i_4 is $M(UC^1_4)\ k^1 + k^1 \leq C_4$. 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:

$$B_h^1 = \frac{\left(\lambda_h^1 - \left[rate(E_1^1) + rate(E_2^1) - rate(E_3^1) \right] \left(\frac{\lambda_h^1}{\lambda_n^1 \left(\frac{\mu_h^1}{\mu_n^1} \right) + \lambda_h^1} \right) - rate(E_4^1) \right)}{\lambda_h^1} \tag{33}$$

$$B_n^1 = \frac{\left(\lambda_n^1 - \left[rate(E_1^1) + rate(E_2^1) - rate(E_3^1) \right] \left(\frac{\lambda_n^1}{\lambda_n^1 + \lambda_h^1 \left(\frac{\mu_n^1}{\mu_h^1} \right)} \right) \right)}{\lambda_h^1} \tag{34}$$

$$B_h^2 = \frac{\left(\lambda_h^2 - rate(E_1^2) \left(\frac{\lambda_h^2}{\lambda_n^2 \left(\frac{\mu_h^2}{\mu_n^2} \right) + \lambda_h^2} \right) - rate(E_2^2) \right)}{\lambda_h^2} \tag{35}$$

$$B_n^2 = \frac{\left(\lambda_n^2 - rate(E_1^2)\left(\frac{\lambda_n^2}{\lambda_n^2 + \lambda_h^2\left(\frac{\mu_n^2}{\mu_h^2}\right)}\right)\right)}{\lambda_n^2} \quad (36)$$

Here *rate*(E^i_c) is calculated by the expected value of a random variable X defined as X= λ^i_c if E^i_c is enabled; 0 otherwise. We compute the total reward generated per unit time as the sum of reward rate earned from each service class/type:

$$SR\left(C, \lambda_h^1, \lambda_n^1, \lambda_h^2, \lambda_n^2\right) = v^1\left[\left(1 - B_h^1\right)\left(\frac{\lambda_h^1}{\mu_h^1}\right) + \left(1 - B_n^1\right)\left(\frac{\lambda_n^1}{\mu_n^1}\right)\right] + v^2\left[\left(1 - B_h^2\right)\left(\frac{\lambda_h^2}{\mu_h^2}\right) + \left(1 - B_n^2\right)\left(\frac{\lambda_n^2}{\mu_n^2}\right)\right] \quad (37)$$

The optimization problem for the spillover algorithm is to identify the best partition (C_1, C_2, C_3, C_4) that would maximize the reward obtainable subject to the imposed QoS constraints defined by Conditions (10) through (13).

6.3. Pure vs. Fast Spillover CAC

We consider two search heuristics methods to find the best partition (C_1, C_2, C_3, C_4) for spillover CAC. The *pure* spillover CAC algorithm finds the optimal partition allocation that would maximize the reward earned per unit time while satisfying QoS constraints specified in Conditions (10) through (13). This method essentially is exhaustive search with heuristics being applied to improve search performance by eliminating combinations that would not generate a legitimate solution. A second variation is the *fast* spillover CAC algorithm that applies a greedy search method to find a near optimal solution fast at the expense of search optimality. However as we shall see,

the solution found by the *fast* spillover algorithm is very close to that found by the *pure* spillover algorithm with search efficiency greatly improved.

Table 6-1: Pure Spillover CAC.

```
Optimal_t search(){
  // Determine the minimum number of channels needed by each service
  call type
  λ1 = [(λ1n/μ1n)+(λ1h/μ1h)];
  λ2 = λ1 k1 + [(λ1n/μ1n)+(λ1h/μ1h)] k2 ;

  c_min1h = c_min1n = get_min(C/k1, λ1, Bt1n) * k1
  c_min2h = c_min2n = get_min(C, λ2, Bt2n)
  for(C4=0, B1H=0; C4<=C; C4=C4+k1) {
    for(C3=0, B1N=0; C3<=C; C3=C3+k1) {
      for(C2=0, B2H=0; C2<=C; C3=C2+k2) {
        C1 = C - (C2+C3+C4)
          // check if at least min number of channels are reserved for
          each service call type
          if((c_min1h<=C)&&(c_min1n<=(C1+C2+C3))  &&
          (c_min2h<=C1+C2)&&(c_min2n<=C1))) {
             optimal= check_optimality(C1,C2,C3,C4)
        }
        B2N = B2N || (B2n <= Bt2n)
        B2H = B2H || (B2h <= Bt2h)
        B1N = B1N || (B1n <= Bt1n)
        // decreasing C1 won't generate a legitimate solution
        if (B2n > Bt2n) break;
      }
      // increasing C2 won't satisfy Bt2n
      if((B2h > Bt2h) && !B2N) break;
    }
    // increasing C1 won't satisfy Bt2h
    if((B1n > Bt1n) && !B2H) break;
  }
  return (optimal)
}
```

Table 6-1 shows a pseudo code listing the *pure* spillover CAC algorithm utilizing exhaustive search for finding optimal resource allocation. This algorithm first determines the minimum number of channels needed to satisfy the QoS constraints by each service call type. This can be done by modeling each service call type as an M/M/n/n queue and determining the minimum number of channels that would satisfy the QoS constraints. 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 type in the rest of the search. For a (C_1, C_2, C_3, C_4) combination that satisfies the minimum-channel requirement, we check if the QoS constraint of the service call type that is allowed only in P_1 is satisfied (that is, class 2 new calls in Table 6-1). If it cannot be satisfied, we stop increasing the number of channels allocated to P_2 because otherwise it would result in fewer channels allocated to P_1. Similarly, if we see that the QoS constraints for class 2 new calls could not be satisfied in P_1 and P_2, we eliminate cases in which P_3 is allocated with more channels. We apply the same logic to guide the channel allocation to P_4. After we find all legitimate solutions, the algorithm concludes by determining the optimal solution.

The fast spillover algorithm employs a heuristic search method to guide the search of (C_1, C_2, C_3, C_4) to satisfy Condition (10) through (13), which may lead to a near optimal solution. This search consists of three stages. In the first stage, we check a special case in which all channels are assigned to the first partition. If we can find a legitimate solution, we return it as the optimal solution. Otherwise, we continue with the second and the third stages. In the second stage, we look for a base partition allocation which will generate a legitimate solution, and in the third stage we improve the base partition allocation to a near-optimal or optimal solution.

Table 6-2: Fast Spillover CAC.

```
Optimal_t spilover_search() {
  legitimate = find_legitimate()
  optimal = find_optimal(legitimate)
  return(optimal)
}
Optimal_t find_legitimate() {
   legitimate = check_optimality(C,0,0,0);
   // if QoS are satisfied when P1 has all channels, return this as
   legitimate
   if(legitimate != NULL)
      return(legitimate);
  // Determine the minimum number of channels needed by each service
```

```
call type
λ1 = [(λ1n/μ1n)+(λ1h/μ1h)];
λ2 = λ1 k1 + [(λ1n/μ1n)+(λ1h/μ1h)] k2 ;

c_min1 = c_min1h = c_min1n = get_min(C/k1, λ1, Bt1n) * k1
c_min2h = c_min2n = get_min(C, λ2, Bt2n)
C1 = c_min2n
C2 = C3 = C4 = 0

for (legitimate = NULL, part = P1;;) {  // do until break

   total = C1 + C2 + C3 + C4
   // check if the total allocated channels exceeds C
   if ((total <=C)&&(part != NULL)) {
      legitimate = check_optimality(C1,C2,C3,C4)
      if(legitimate == NULL){
         switch(part) {
         case P1:
            // increase the minimum channel needed for C2 new calls
            if QoS not satisfied
            if(B2n > Bt2n) {
               C1 = c_min2n = c_min2n + k2
            }
            else { // determine channel allocation in 2nd partition
               part = P2;
               if(c_min2h < c_min2n) c_min2h = C1;
               c_min2h += (C - c_min2h) % k1;
               C2 = c_min2h - c_min2n;
            }
            break;
         case P2:
             if(c_min2h >= C) part = NULL;
             if(B2h > Bt2h) {
               C2 = C2 + k2;
               c_min2h = C1 + C2;
               c_min2h += (C - c_min2h) % k1;
               C2 = c_min2h - C1;
             }
             else { // determine channel allocation in 3rd partition
               part = P3;
               if(c_min1n < c_min2h) c_min1n = c_min2h;
               C3 = (c_min1n - C2 - C1);
             }
             break;
           case P3:
             if(c_min1n >= C) part = NULL;
             if(B1n > Bt1n) {
               C3 = C3 + k1;
               c_min1n = C1 + C2+ C3;
             }
             else { // determine channel allocation in 4th partition
               part = P4;
               if(c_min1h < c_min1n) c_min1h = c_min1n;
               C4 = (c_min1h - C1 - C2 - C3);
               if((C4>0)&&( B1h > Bt1h))
                   C4 = C4 + k1;
             }
```

```
                break;
              case P4:
                if(c_min^1_h >= C) part = NULL;
                if(B^1_h > Bt^1_h) {
                  C_4 = C_4 + k^1;
                  c_min^1_h = C_1 + C_2 + C_3 + C_4;
                }
                break;
              default:
                break;
              }
           }
           else break; // found a legitimate solution
      }
      else { // if ((total <=C)&&(part != NULL)) {
         if(part == P3) C_3 = C_3 - k^1;
         else if(part == P4) C_4 = C_4 - k^1;

         if(C_2 > 0){ // move one channel from C_2 to C_1 and continue
              c_min^2_n = C_1 = C_1 + 1;
              C_2 = C_2 - 1;
              part = P1;
              c_min^1_h = c_min^1_n = c_min^1;
              if(c_min^1_n < c_min^2_h) c_min^1_n = c_min^2_h;
              C_3 = C_4 = 0;
         }
         else if(C_3 > 0){ // move k^1 channels from C_3 to C_2 and continue
              C_2 = C_2 + k^1;
              C_3 = C_3 - k^1;
              Part = P2;
              c_min^2_h = C_1 + C_2;
              c_min^1_h = c_min^1_n = c_min^1;
              if(c_min^1_n < c_min^2_h) c_min^1_n = c_min^2_h;
              C_3 = C_4 = 0;
         }
         else break; // no more combination to search so break
      }
   }
   return (legitimate)
}
```

```
Optimal_t find_optimal(cur) {
  c_min^2_h = cur.C1 + cur.C2;
  c_min^1_n = cur.C1 + cur.C2 + cur.C3;
  // search for a local maximum
  for(C1=C; C1 ≥ cur.C1; C1=C1-k^2) {
     if(c_min^2_h > C1) C2 = c_min^2_h - C1;
     else C2 = 0;

     C2 = C2 + ((C - C2 - C1) % k^1);
     for(; C2 ≤ C - C1; C2=C2+k^1) {
        if(c_min^1_n > (C1 + C2)) C3 = c_min^1_n - C1 - C2;
        else C3 = 0;

        for(; C3 ≤ C - C1 - C2; C3=C3+k^1) {
           C4 = C - C3 - C2 - C1;
           optimal = check_optimality(C1,C2,C3,C4)
        }
     }
     if(C1 < optimal.C1 -1) break;
  }
  return(optimal)
}
```

Table 6-2 shows a pseudo code listing of the fast spillover CAC algorithm. The `spillover_search` method first determines a legitimate solution via `find_legitimate`, and later uses this solution as the starting point to determine a legitimate 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_legitimate` works: It first determines the minimum number of channels needed by each service call type to satisfy the QoS constraints. Later it repeats the following greedy search steps until it finds a partition allocation that leads to a legitimate solution. In each step we focus on a single partition stored in parameter *part*. We start searching for the minimum number of channels that should be allocated in P_1 to satisfy QoS constraints of class 2 new calls. Since class 2 new calls are only allowed in this partition, we cannot satisfy QoS constraints unless we assign enough channels to accommodate class 2 new calls in this partition. After reserving minimum number of channels in P_1, we focus on P_2. Since class 2 handoff calls are not allowed in P_3 and P_4

we assign enough number of channels to P_2 to satisfy QoS constraints of class 2 handoff calls. Similarly, we assign enough number of channels to P_3 to such that the QoS constraints of class 1 new calls are satisfied. Finally we assign channels to P_4 to satisfy the most stringent QoS constraints of class 1 handoff calls. During the channel assignment if we don't have enough channels to satisfy the QoS constraints, as a backup measure we first assign more channels to P_1 and continue our search with channel assignment to P_2. If P_2 becomes empty we start moving channels from P_3 to P_2 as the secondary backup measure. We do these to take the advantage of multiplexing power of P_1 and P_2 and to look for other solutions that may satisfy QoS constraints of all service classes. In partitions P_3 and P_4 only class 1 calls are allowed, so we set the number of channels in these partitions as multiple of k^1. In order to achieve this we pad extra channels to the P_2 if necessary. If none of the backup measures help, `find_legitimate` function returns no legitimate solution.

For the current partition allocation (C_1, C_2, C_3, C_4) leading a legitimate solution, `find_optimal` evaluates all combinations of (C_1', C_2', C_3', C_4') where $C_1 \leq C_1' \leq C$, $C_2 \leq C_2' \leq C\text{-}C_1'$, $C_3 \leq C_3' \leq C\text{-}C_1'\text{-}C_2'$, and $C_4' = C\text{-}C_1'\text{-}C_2'\text{-}C_3'$. We use the following heuristic during the optimal solution search: if decrementing C_1 does not improve the optimality there is no need to evaluate lower C_1 values. Before evaluating a new C_1, we check if the last C_1 that we evaluated has 2 or more channels less than the C_1 partition of the current optimal channel allocation. We stop the search and return the current optimal if this is the case.

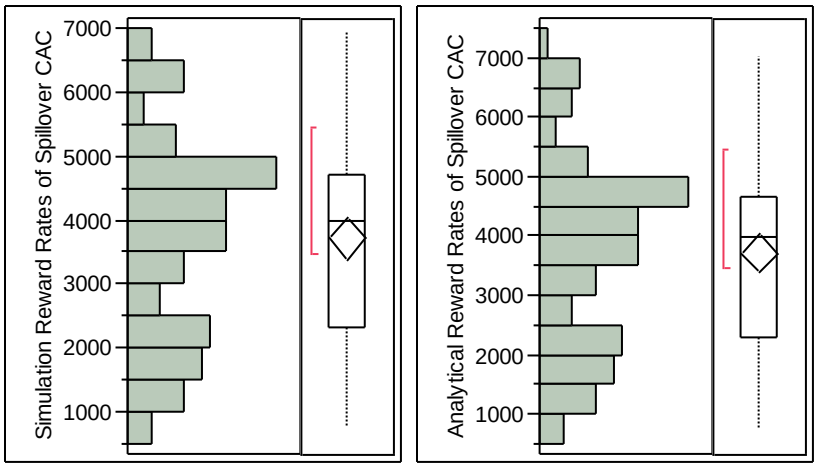

Figure 6-3: Histogram and Box Plot Diagrams of Simulated vs. Analytical Reward Rates of Spillover CAC.

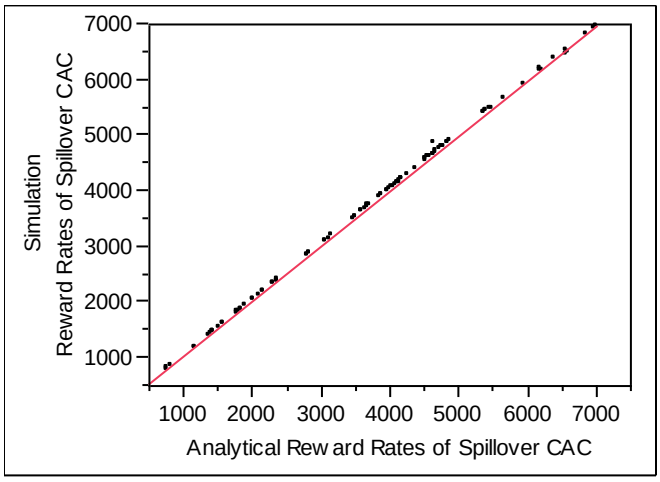

Figure 6-4: Bivariate Fit of Simulated vs. Analytical Reward Rates of Spillover CAC.

Table 6-3: Comparison of Analytical vs. Simulation Results for Spillover CAC.

Spillover CAC	Simulation	Analytical	Absolute Difference	Percentage Difference
Number of observations	101	101	0	0%
Mean	3690.2	3698.7	8.48	0.133%
Median	3976.8	3978.7	0.32	0.013%
Standard Deviation	1610.3	1625.1	32.63	0.583%
Range	6201.9	6271.6	318.10	5.881%
Midspread	2467.7	2443.6	6.09	0.135%

6.4. Supporting Hypothesis 2 for Spillover CAC

We have compared reward rates generated from spillover CAC with these generated from partitioning and threshold-based CAC. The results (which will be presented in Chapter 8 and is not reported here to save space) support Hypothesis 2. We further support Hypothesis 2 for spillover CAC with simulation validation.

We perform a nonparametric correlation analysis to validate analytical results obtained from theoretical analysis. Figure 6-3 presents the histogram and box plot diagrams of analytical vs. simulated reward rates. Both the histogram and box plot show very similar characteristics. Figure 6-4 shows a bivariate fit line of calculated vs. observed new call arrival/departure rates. The fit line passes through the origin with a slope of 1 indicating a strong correlation. Table 6-3 compares analytical results vs. simulation results for spillover CAC over 27 test scenarios with the simulator developed in Chapter 4. As the effect size, the Spearman's ρ and probability>$|\rho|$ are calculated as 0.9996 and 0.0001, respectively, thereby supporting our hypothesis strongly and indicating that there is a perfect positive linear relationship between analytical and simulated reward rates significantly.

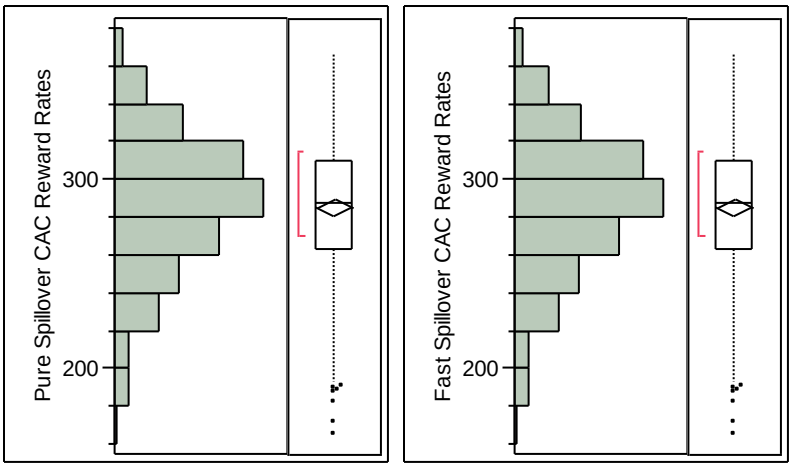

Figure 6-5: Histogram and Box Plot Diagrams of Pure vs. Fast Spillover Reward Rates.

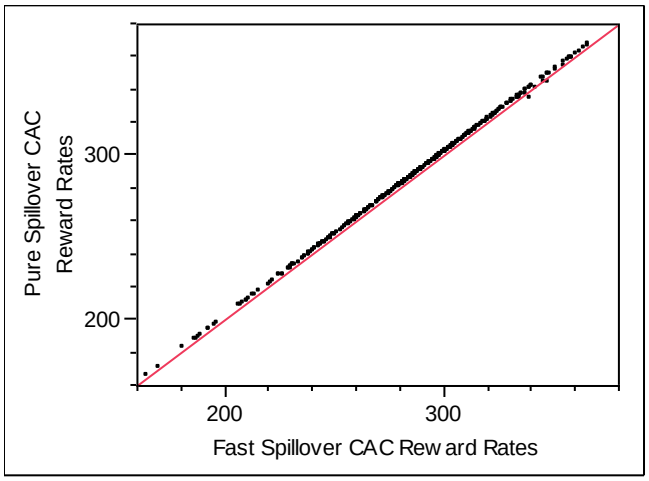

Figure 6-6: Bivariate Fit of Pure Spillover CAC Reward Rates vs. Fast Spillover CAC Reward Rates.

Table 6-4: Comparison of Reward Rates Obtainable by Pure vs. Fast Spillover CAC.

Spillover CAC	Pure	Fast	Absolute Difference	Percentage Difference
Number of observations	368	368	0	0%
Mean	285.3	285.4	-0.0699	-0.021%
Median	288	288	0	0%
Standard Deviation	38.1	38.2	0.482	0.140%
Range	202.0	202.0	6.024	1.775%
Midspread	47.1	47.1	0	0%

Fast spillover CAC is designed with search efficiency in mind without compromising search optimality with respect to pure spillover CAC. To test this hypothesis, we generate a large number of test scenarios (1000) randomly to simulate random call arrival and departure rates of service classes with QoS constraints with the objective to evaluate solution optimality of pure spillover CAC vs. fast spillover CAC.

We observe that both spillover CAC algorithms generate exactly the same number of legitimate solutions (368) out of the 1000 test scenarios. We perform a nonparametric correlation analysis to validate analytical results obtained from theoretical analysis. Figure 6-5 presents the histogram and box plot diagrams of pure vs. fast spillover CAC reward rates. Both the histogram and box plot show very similar characteristics. Figure 6-6 shows a bivariate fit line of reward rates calculated via pure vs. fast spillover CAC algorithms. The fit line passes through the origin with a slope of 1 indicating a strong correlation. Table 6-4 compares reward rates obtainable by pure vs. fast spillover CAC. As the effect size, the Spearman's ρ and probability>$|\rho|$ are calculated as 1.0000 and 0, respectively, thereby supporting our hypothesis strongly and indicating that there is a perfect positive linear relationship between reward rates calculated via pure spillover vs. fast spillover CAC significantly.

In Chapter 8, we will report how spillover CAC fairs compared against other CAC algorithms in the class of CAC algorithms designed for reward optimization and QoS satisfaction.

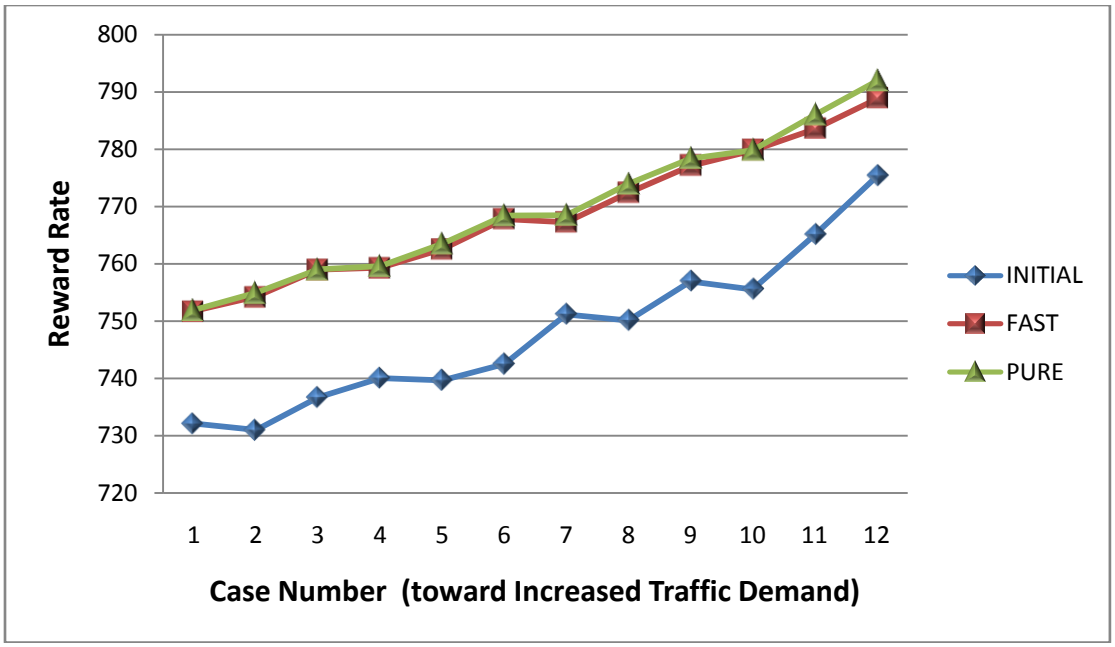

Figure 6-7: Sensitivity of Initial Solution Quality with respect to Traffic Demand.

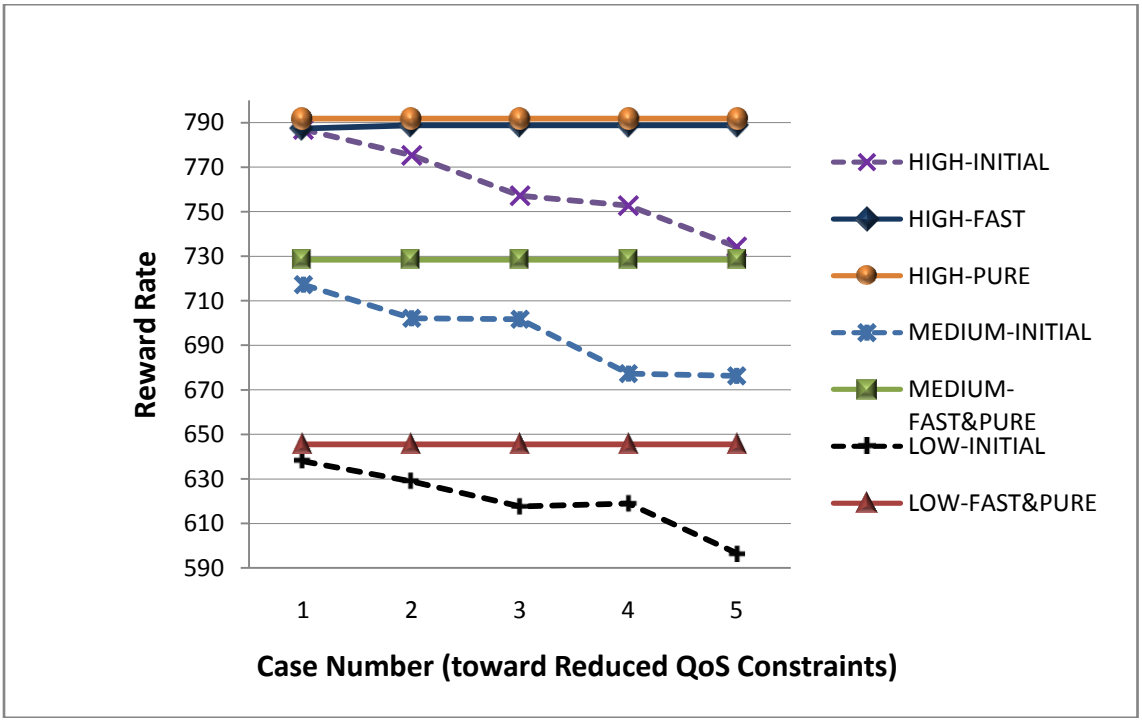

Figure 6-8: Sensitivity of Initial Solution Quality with respect to QoS Constraints.

6.5. Sensitivity Analysis of Solution Quality

Recall that fast spillover CAC consists of two phases. The first phase searches for a legitimate 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 6-7 shows the solution quality of the initial solution as the traffic demand increases, going from case 1 with ($\lambda^1_h = 3.66$, $\lambda^1_n = 7.32$, $\lambda^2_h = 2.69$, and $\lambda^2_n = 3.59$) to case 11 with ($\lambda^1_h = 3.66$, $\lambda^1_n = 7.32$, $\lambda^2_h = 5.99$, and $\lambda^2_n = 7.98$) covering low to high traffic demands. We observe that the solution quality of the initial solution is largely insensitive to the traffic demand. Figure 6-7 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 6-8 shows the solution quality of the initial solution as we reduce QoS constraints, going from case 1 with $B^1_ht = 0.01$, $B^1_nt = 0.025$, $B^2_ht = 0.02$ and $B^2_nt = 0.05$ to case 5 with $B^1_ht = 0.05$, $B^1_nt = 0.125$, $B^2_ht = 0.1$ and $B^2_nt = 0.25$. We see that when QoS constraints are reduced (made less stringent), the solution quality of the initial solution found by fast spillover CAC decreases because less stringent QoS constraints allows the `find_legitimate` function find a solution earlier. Therefore, the initial

solution is sensitive to the QoS constraints. However, the final solution result found by the greedy search remains the same. We conclude that the solution quality of the final solution found by the fast spillover CAC algorithm is insensitive to the solution quality of the initial solution.

CHAPTER 7 Elastic Threshold-Based Call Admission Control for Reward Optimization with QoS Guarantees

Both hybrid and spillover CAC algorithms developed utilize the concept of resource partitioning. Partitioning-based CAC, however, loses the power of multiplexing since resources are reserved to serve only certain service classes. In this Chapter, we develop a CAC algorithm called *elastic threshold-based CAC* based on thresholds only. This CAC algorithm improves solution optimality at the expense of solution efficiency compared with hybrid and spillover CAC. This Chapter is based on the work published in [73][5]. In Chapter 8, we will compare and contrast elastic CAC vs. hybrid and spillover CAC.

7.1. Elastic Threshold-Based CAC

Elastic threshold-based CAC utilizes the concept of threshold-based CAC which applies a separate and distinct threshold to each service type. By sharing all channels available among all service classes, threshold-based CAC can establish high channel utilizations. It applies thresholds to limit the traffic from low-priority calls and thus reserves more bandwidth for high-priority calls. Although threshold-based CAC has great

[5] © 2008 IEEE. Reprinted, with permission, from O. Yilmaz, and I.R. Chen, "On QoS guarantees with reward optimization for servicing multiple priority classes in wireless networks," *IEEE 17th International Conference on Computer Communications and Networks*, St. Thomas, US Virgin Islands, August 2008, pp 1 - 6.

power for channel sharing, it suffers from the use of discrete thresholds which cut traffic from service classes abruptly and reject all further traffic. We develop elastic threshold-based CAC to improve the effectiveness of threshold-based CAC in terms of QoS satisfaction and thus higher rewards generated by applying elastic thresholds.

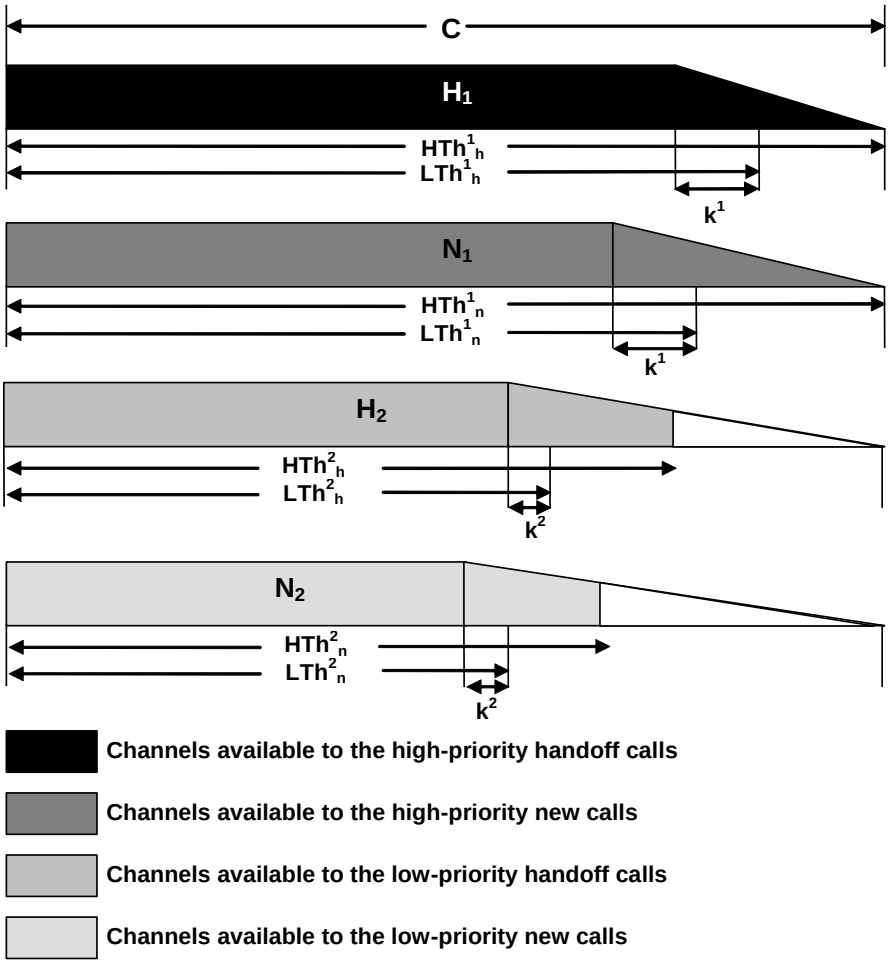

Figure 7-1: Elastic Threshold-Based Admission Control.

For ease of reference, 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. Elastic threshold-based CAC uses two thresholds, a high (*Hth)* threshold

and a low (*Lth)* threshold, for each service classes in the system. The CAC algorithm rejects a fraction of class *i* new calls when LTh^i_n is reached and rejects all class *i* new calls when HTh^i_n is reached. Similarly, the CAC algorithm starts blocking a fraction of class *i* handoff calls when LTh^i_h is reached and blocks all class *i* handoff calls when HTh^i_h is passed. For each threshold value, elastic threshold-based CAC assumes that the threshold is reached if accepting an incoming call will increase the number of channels above the threshold value. The probabilities P^i_n and P^i_h of accepting a new and handoff call of service class *i* are given by:

$$P_n^i = \begin{cases} 1 & n + k^i \le LTh_n^i \\ 1 - \left[\left(n + k^i - LTh_n^i \right) \Big/ \left(C - LTh_n^i \right) \right] & LTh_n^i < n + k^i \le HTh_n^i \\ 0 & HTh_n^i < n + k^i \end{cases} \tag{38}$$

$$P_h^i = \begin{cases} 1 & n + k^i \le LTh_h^i \\ 1 - \left[\left(n + k^i - LTh_h^i \right) \Big/ \left(C - LTh_h^i \right) \right] & LTh_h^i < n + k^i \le HTh_h^i \\ 0 & HTh_h^i < n + k^i \end{cases} \tag{39}$$

Here *C* and *n* represent the total number of channels and the number of channels used in the system, respectively. Figure 7-1 illustrates the thresholds and call arrivals in elastic threshold-based CAC running in a system with *C* number of channels. Each color represents the call arrival rate of a different service call type accepted by the system depending on the total number of channels used by all service types. For example, LTh^2_n (a low threshold) is triggered if a new low-priority class 2 call arrives when the number of channels used by the system is greater than LTh^2_n-k^2. After this, elastic CAC starts rejecting a fraction of class 2 call arrivals until a class 2 new call arrives when the number of channels used is greater than HTh^2_n –k^2. After the higher threshold is reached, the system stops accepting class 2 new calls.

Elastic threshold-based CAC for reward optimization with QoS guarantees thus aims to find the optimal set (LTh^1_h, LTh^1_n, LTh^2_h, LTh^2_n, Hth^1_h, HTh^1_n, HTh^2_h, HTh^2_n) that would yield the highest reward while satisfying the QoS constraints specified by Conditions (10) through (13).

7.2. Performance Model

We analyze the elastic threshold-based CAC algorithm by using an SPN model. An SPN model is used rather than a Markov model because of the interdependency between thresholds assigned to handoff and new calls of various service classes. Figure 7-2 shows an SPN model for the elastic threshold-based admission control with two service classes.

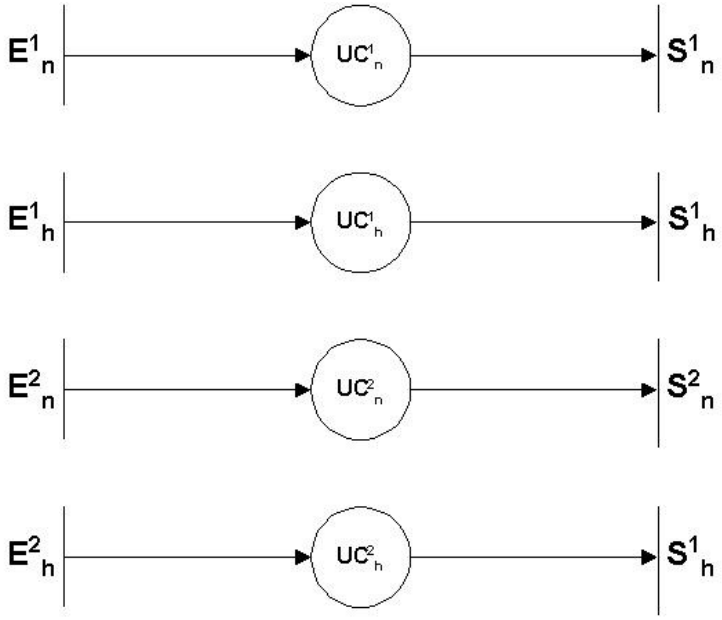

Figure 7-2: Performance Model for Elastic Threshold-Based Admission Control.

The transitions and places shown in Figure 7-2 are described as follows. For *transitions*, E^i_n models new call arrivals of service class *i* with a variable rate $f(\lambda^i_n)$ given as:

$$f\left(\lambda_n^i\right)=\begin{cases}\lambda_n^i & n+k^i \le LTh_n^i \\ \lambda_n^i\left(1-\left[\left(n+k^i-LTh_n^i\right)\Big/\left(C-LTh_n^i\right)\right]\right) & LTh_n^i < n+k^i \le HTh_n^i \\ 0 & HTh_n^i < n+k^i\end{cases} \tag{40}$$

E^i_h models handoff call arrivals of service class *i* at rate $f(\lambda^i_h)$ given as:

$$f\left(\lambda_h^i\right)=\begin{cases}\lambda_h^i & n+k^i \le LTh_h^i \\ \lambda_h^i\left(1-\left[\left(n+k^i-LTh_h^i\right)\Big/\left(C-LTh_h^i\right)\right]\right) & LTh_h^i < n+k^i \le HTh_h^i \\ 0 & HTh_h^i < n+k^i\end{cases} \tag{41}$$

S^i_n models service of new calls of service class *i* with a service rate of $M(UC^i_n)$ multiplied with μ^i_n where $M(UC^i_n)$ stands for the number of tokens in place UC^i_n. S^i_h models service of handoff calls of service class i with a service rate of $M(UC^i_h)$ multiplied with μ^i_n where $M(UC^i_h)$ stands for the number of tokens in place UC^i_h. For places, UC^1_n models the execution state of service class 1 new calls; UC^1_h models the execution state of service class 1 handoff calls; UC^2_n models the execution state of service class 2 new calls; and UC^2_h models the execution state of service class 2 handoff calls.

A new service request arrival is admitted only if the threshold assigned is not yet reached. Therefore we assign an enabling predicate to guard E^1_n, E^1_h, E^2_n, and E^2_h, with thresholds C^1_n, C^1_h, C^2_n, and C^2_h, respectively. Consequently, the enabling predicate of E^1_n is $[M(UC^1_n) + M(UC^1_h)]\ k^1 + k^1 + [M(UC^2_n) + M(UC^2_h)]\ k^2 \le HTh^1_n$. The enabling predicate of E^1_h is $[M(UC^1_n) + M(UC^1_h)]\ k^1 + k^1 + [M(UC^2_n) + M(UC^2_h)]\ k^2 \le HTh^1_h$. The enabling predicate of E^2_n is $[M(UC^1_n) + M(UC^1_h)]\ k^1 + k^2 + [M(UC^2_n) +$

$M(UC^2_h)]\ k^2 \leq HTh^2_n$. Finally, the enabling predicate of E^2_h is $[M(UC^1_n) + M(UC^1_h)]\ k^1 + k^2 + [M(UC^2_n) + M(UC^2_h)]\ k^2 \leq HTh^2_h$.

The blocking probability B^1_n, B^1_h, B^2_n, and B^2_h are calculated from the SPN model by:

$$B_n^1 = \frac{\left(\lambda_n^1 - rate(E_n^1)\right)}{\lambda_n^1} \tag{42}$$

$$B_h^1 = \frac{\left(\lambda_h^1 - rate(E_h^1)\right)}{\lambda_h^1} \tag{43}$$

$$B_n^2 = \frac{\left(\lambda_n^2 - rate(E_n^2)\right)}{\lambda_n^2} \tag{44}$$

$$B_h^2 = \frac{\left(\lambda_h^2 - rate(E_h^2)\right)}{\lambda_h^2} \tag{45}$$

where $rate(E^i_c)$ is calculated by finding the expected value of a random variable X defined as $X= \lambda^i_c$ if E^i_c is enabled; 0 otherwise. A "legitimate" solution from the elastic threshold-based admission control algorithm must generate B^1_n, B^1_h, B^2_n, and B^2_h to satisfy the QoS constraints specified by Conditions (10) through (13) discussed earlier.

We compute the reward generated per unit time from the elastic threshold-based admission control algorithm to the cell as the sum of rewards earned by each service class:

$$ETR(C, \lambda^1_h, \lambda^1_n, \lambda^2_h, \lambda^2_n) = ETR^1_h + ETR^1_n + ETR^2_h + ETR^2_n \tag{46}$$

Here ETR^1_h, ETR^1_n, ETR^2_h, and ETR^2_n stand for the rewards generated per unit time due to high-priority handoff calls, high-priority new calls, low-priority handoff calls, and low-priority new calls, respectively, given by:

$$ETR^i_h = (1- B^i_h)\ \lambda^i_h\ v^i / \mu^i_h \tag{47}$$

$$ETR^{i}_{n} = (1- B^{i}_{n})\, \lambda^{i}_{n}\, v^{i} / \mu^{i}_{n.} \quad (48)$$

Thus, the total reward rate earned in the cell is:

$$ETR\left(C, \lambda_h^1, \lambda_n^1, \lambda_h^2, \lambda_n^2\right) = v^1\left[\left(1-B_h^1\right)\left(\frac{\lambda_h^1}{\mu_h^1}\right)+\left(1-B_n^1\right)\left(\frac{\lambda_n^1}{\mu_n^1}\right)\right]+v^2\left[\left(1-B_h^2\right)\left(\frac{\lambda_h^2}{\mu_h^2}\right)+\left(1-B_n^2\right)\left(\frac{\lambda_n^2}{\mu_n^2}\right)\right] \quad (49)$$

7.3. Greedy Elastic CAC

We develop a heuristic-based search method to determine the optimal threshold combinations that would maximize the reward earned per unit time while satisfying QoS constraints specified in Conditions (10) through (13). This heuristic method is essentially a greedy search method for solution efficiency at the expense of search optimality. However, as we shall see, the solution found by this heuristic search method is at least as good as or better than the solutions found by the base partitioning and threshold-based CAC algorithms.

Table 7-1: Elastic Threshold-Based CAC.

```
Optimal_t greedy_search(Δ) {
  legitimate = find_legitimate()
  if legitimate is not null
     optimal = find_optimal(legitimate, Δ)
  return(optimal)
}

Optimal_t find_legitimate() {
  legitimate = search_hightolow()
  if legitimate is null
     legitimate = search_lowtohigh()
  return(legitimate)
}

Optimal_t search_hightolow() {
  // Determine the minimum number of channels needed by each service
  call type
  λ1 = [(λ1n/μ1n)+(λ1h/μ1h)];
  λ2 = λ1 k1 + [(λ1n/μ1n)+(λ1h/μ1h)] k2 ;

  c_min1 = get_min(C/k1, λ1, Bt1n) * k1
```

```
  c_min^2 = get_min(C, λ^2, Bt^2_n)
  legitimate = NULL
  if((c_min^1 > C) or (c_min^2 > C)) {
     // no legitimate solution, so return null
     return legitimate;
  }
  LTh^1_h = HTh^1_h = LTh^1_n = HTh^1_n = LTh^2_h = HTh^2_h = LTh^2_n = HTh^2_n = C
  repeat until legitimate is not null {
     check_optimality of the current threshold combination
     if legitimate is null {
          if (B^2_n < Bt^2_n){
               --LTh^2_n;
               --HTh^2_n;
          }
          if (B^2_h < Bt^2_h){
               --LTh^2_h;
               --HTh^2_h;
          }
          if ((B^2_n > Bt^2_n) and (B^2_h > Bt^2_h)) {
             if (B^1_n < Bt^1_n)){
                  --LTh^1_n;
                  --HTh^1_n;
             }
             else {
                  break the loop
             }
          }
     }
  }
  return (legitimate)
}

Optimal_t search_lowtohigh() {

  repeat legitimate is null {
     HTh^1_n = C
     check_optimality of the current threshold combination
     if legitimate is null
          if (B^2_n > Bt^2_n){
               if ((LTh^2_n ≥ C) or (HTh^2_n ≥ C)) break;
               else if LTh^2_n < HTh^2_n
                    ++LTh^2_n
               else if HTh^2_n < C
                    ++HTh^2_n
          }

          if (B^2_h > Bt^2_h){
               if ((LTh^2_h ≥ C) or (HTh^2_h ≥ C)) break;
               else if LTh^2_h < HTh^2_h
                    ++LTh^2_h
               else if HTh^2_h < C
                    ++HTh^2_h
          }

          if (B^1_n > Bt^1_n) {
               if ((LTh^1_n ≥ C) or (HTh^1_n ≥ C)) break;
               else if LTh^1_n < HTh^1_n
```

```
                    ++LTh²n
          }
          if ((B²n ≤ Bt²n) and (B²h ≤ Bt²h) and (B¹n ≤ Bt¹n) and (B¹h > Bt¹h)){
               if LTh²n < HTh²n
                    --HTh²n
               if LTh²h < HTh²h
                    --HTh²h
               if (B¹n ≤ Bt¹n)
                    --LTh¹n
               if the new threshold combination has already checked
                    break the loop
          }
          if no threshold is changed
               break
     }
  }
  else break the loop
  if the new threshold combination is already checked
     check the neighbors and break
  return (legitimate)
}

Optimal_t find_optimal(optimal, Δ){
  repeat forever {
     // try higher thresholds
     for all HTh²n in (HTh²n - Δ, HTh²n) {
        for all HTh²h in (HTh²h - Δ, HTh²h) {
           check_optimality of the current threshold combination
           if optimal is changed
                break loops
        }
     }
     // try lower thresholds
     for all LTh²n in LTh²n ± Δ {
        for all LTh²h in LTh²h ± Δ {
           for all LTh¹n in LTh¹n ± Δ {
                check_optimality of the current threshold combination
                if optimal is changed
                     break loops
           }
        }
     }
     if optimal is not changed
        break
  }
  return (optimal)
```

Table 7-1 shows a pseudo code listing of the elastic threshold-based CAC algorithm utilizing a greedy search method. This algorithm determines the optimal threshold combination in two steps preformed in the *greedy_search* function. This

function accepts a delta (Δ) value as input to determine the set of threshold combinations in the range of current thresholds $\pm \Delta$. The search stops when the current optimal reward is higher than the reward generated for all threshold combinations in this set.

In the *greedy_search* function, we first invoke the *find_legitimate* function to determine a legitimate solution. As the next step, we call *find_optimal* to perform a greedy search starting from a legitimate solution to determine an optimal solution which satisfies QoS constraints while maximizing the reward rate.

In the *find_legitimate* function we try two independent methods to determine a legitimate solution which satisfies the QoS constraints of all service call types. In the first method, *search_hightolow*, we set all thresholds to *C* and check if this combination is legitimate. We reduce the bandwidth used by low priority classes until we find a legitimate solution or we start missing QoS constraints of low priority classes. When we cannot limit the arrivals of low priority classes anymore we start reducing arrivals of high priority new calls as well by lowering the lower threshold of this service call type. If both approaches do not generate a legitimate solution, this method returns no legitimate solution.

The second method, *search_lowtohigh* is only called if *search_hightolow* cannot find a legitimate solution. In this method we first determine the minimum number of channels needed to satisfy the QoS constraints of each service call type. This can be done by modeling each service call type as an M/M/n/n queue and determining the minimum number of channels that would have satisfied the QoS constraints. This method helps to eliminate all threshold combinations that do not provide at least the minimum number of channels to each service call type in the rest of the search. In subsequent iterations, we

increase the lower thresholds of low priority calls if lower threshold is less than the higher threshold; otherwise we increase the higher threshold until higher threshold reaches to *C* or we find a legitimate solution. Similarly, we increase the lower threshold of high priority new calls to satisfy the QoS constraints of this service call type.

If we do not change any of these thresholds and we still cannot satisfy the QoS constraints of high priority handoff calls, we decrease the higher thresholds of low priority calls and higher threshold of high priority new calls. This step is done by rolling back to a previous state when the new thresholds cannot find a solution. At any point if we determine that we encounter a threshold combination that has already been evaluated, we break looping and instead check the threshold values which are one less or one more than the current threshold values. This last check is essential because we may come across with a previously evaluated threshold in the borderline cases, and retrying this threshold value may cause an infinite loop. After finding a legitimate solution, we attempt adjacent threshold values to determine a legitimate solution with a higher reward rate. When no adjacent threshold returns a higher reward rate, the CAC algorithm returns the optimal solution found.

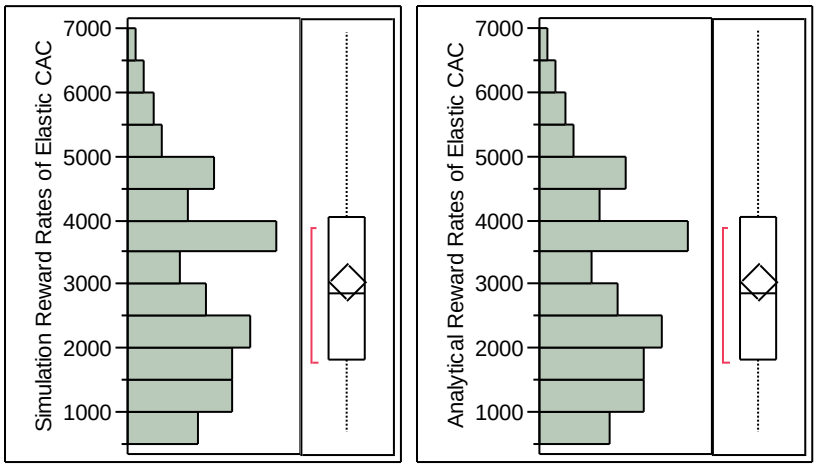

Figure 7-3: Histogram and Box Plot Diagrams of Simulation vs. Analytical Reward Rates of Elastic CAC.

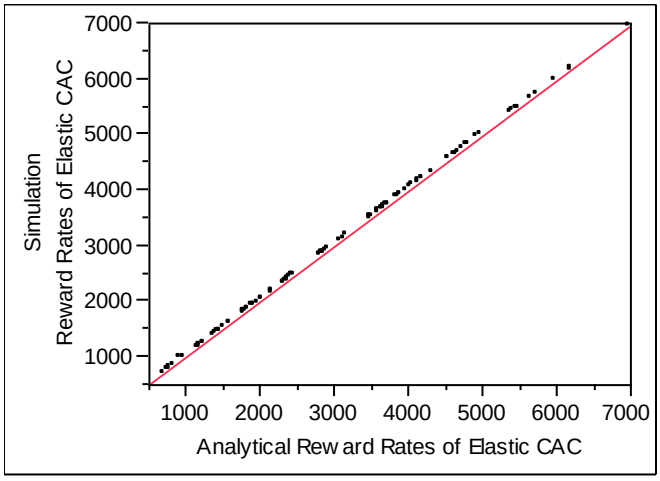

Figure 7-4: Bivariate Fit of Simulation vs. Analytical Reward Rates of Elastic CAC.

Table 7-2: Comparison of Analytical vs. Simulation Results for Elastic CAC.

Elastic CAC	Simulation	Analytical	Absolute Difference	Percentage Difference
Number of observations	106	106	0	0%
Mean	3025.3	3026.8	1.50	-0.0427%
Median	2845.8	2844.6	0.01	0.00046%
Standard Deviation	1478.1	1484.3	12.66	0.808%
Range	6235	6289.4	132.25	8.812%
Midspread	2209.1	2209.8	1.43	0.056%

7.4. Supporting Hypothesis 2 for Elastic Threshold-Based CAC

We have tested solution optimality of elastic CAC by comparing reward rates generated from elastic CAC with these generated from partitioning and threshold-based CAC covering all 27 test scenarios with the simulator developed in Chapter 4. The results (to be presented in Chapter 8 along with others) support both Hypothesis 2a and Hypothesis 2b. We further support Hypothesis 2 for elastic threshold-based CAC with simulation validation. Figure 7-3 presents the histogram and box plot diagrams of simulation vs. analytical reward rates. Both the histogram and box plot show very similar characteristics. Figure 7-4 shows a bivariate fit line of simulation vs. analytical reward rates. The fit line passes through the origin with a slope of 1 indicating a strong correlation. Table 7-2 compares analytical results vs. simulation results for elastic CAC with the simulator developed in Chapter 4, also covering all 27 test scenarios. We perform a nonparametric correlation analysis to validate analytical results with simulation results collected. As for the effect size, the Spearman's ρ and probability>$|\rho|$ are calculated as 0.9998 and 0.001, respectively, thereby supporting our hypothesis strongly and indicating that there is a perfect positive linear relationship between simulation and analytical reward values significantly.

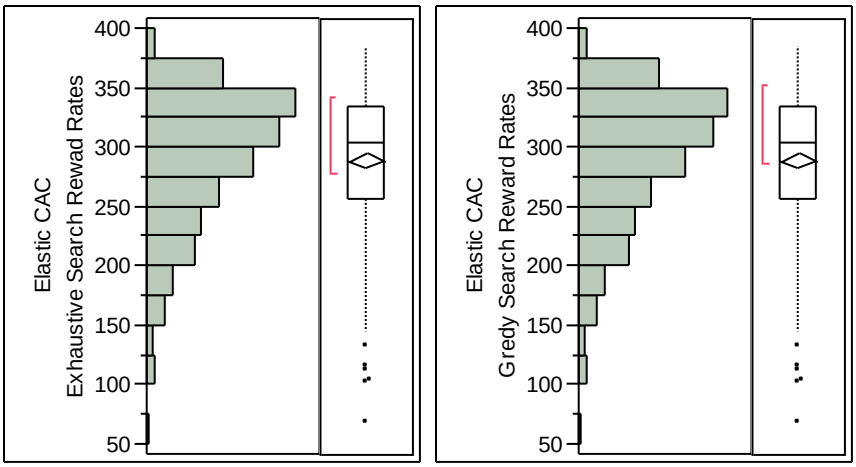

Figure 7-5: Histogram and Box Plot Diagrams of Exhaustive vs. Greedy Search Reward Rates of Elastic CAC.

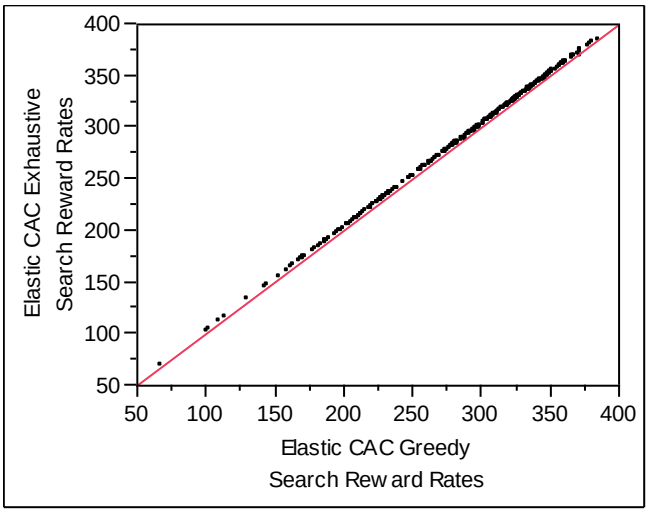

Figure 7-6: Bivariate Fit of Exhaustive Reward Rates vs. Greedy Reward Rates of Elastic CAC.

Table 7-3: Comparison of Reward Rates Obtained by Greedy vs. Exhaustive-Search Elastic CAC.

Elastic CAC	Greedy Search	Exhaustive Search	Absolute Difference	Percentage Difference
Number of observations	346	346	0	0%
Mean	288.80	289.02	-0.226	-0.0687%
Median	303.89	304.07	0	0%
Standard Deviation	59.11	59.27	0.584	0.1748%
Range	315.81	317.45	5.715	1.536%
Midspread	78.16	78.18	0.134	0.043%

To test solution optimality further, we randomly generate 1614 workload conditions as input to greedy elastic CAC and compare its performance in terms of reward rate obtainable with QoS guarantees vs. exhaustive-search elastic CAC which essentially applies exhaustive search for finding optimal solutions. Table 7-3 compares reward rates obtainable by greedy vs. exhaustive-search elastic threshold-based CAC as well as statistical data in terms of the absolute and percentage differences between these results. We see that both can generate legitimate solutions for 346 out of 1614 test scenarios.

We perform a nonparametric correlation analysis to determine the correlation between reward rates obtained via exhaustive search vs. heuristic search. Figure 7-5 presents the histogram and box plot diagrams of heuristic search vs. exhaustive search reward rates. Both the histogram and box plot show very similar characteristics. The histograms show negatively skewed characteristics. Figure 7-6 shows a bivariate fit line of reward rates obtained via exhaustive search vs. heuristic search. The fit line passes through the origin with a slope of 1 indicating a strong correlation. As for the effect size, the Spearman's ρ and probability>$|\rho|$ are calculated as 0.9999 and 0.0000, respectively, thereby supporting our hypothesis strongly and indicating that there is a perfect positive linear relationship between reward rates calculated via exhaustive search vs. heuristic search.

7.4.1 Sensitivity Analysis

Our simulation results presented in this chapter are based on exponentially distributed cell residence time and call duration time. Below we test the sensitivity of simulation results with respect to normal, hyper-exponential and uniform time

distributions for the cell residence time and call duration time. For each time distribution, we repeat the simulation covering the same 27 test scenarios with each test scenario covering 2 mobile populations at 256 and 512. We again perform a nonparametric correlation analysis to show that analytical vs. simulation reward rates are highly correlated.

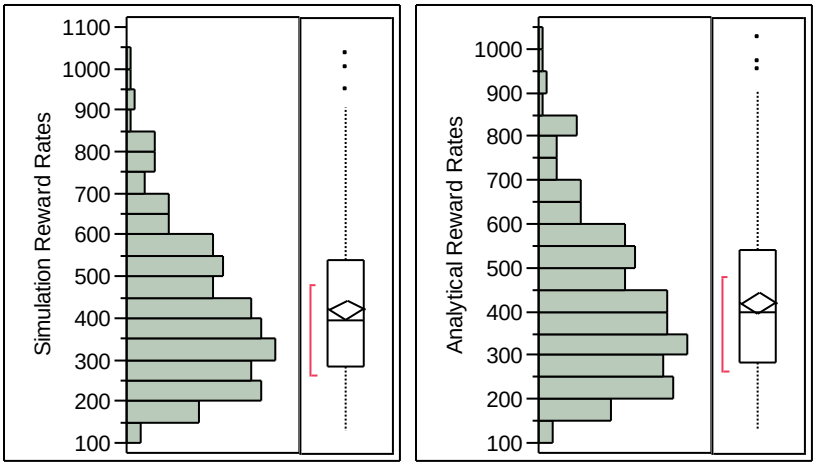

Figure 7-7: Histogram and Box Plot Diagrams of Analytical vs. Simulation reward Rates under Normal Distribution.

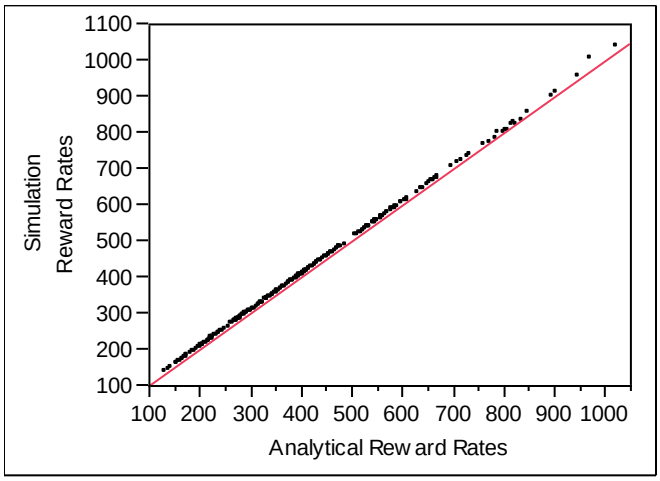

Figure 7-8: Bivariate Fit of Analytical vs. Simulation Reward Rates under Normal Distribution.

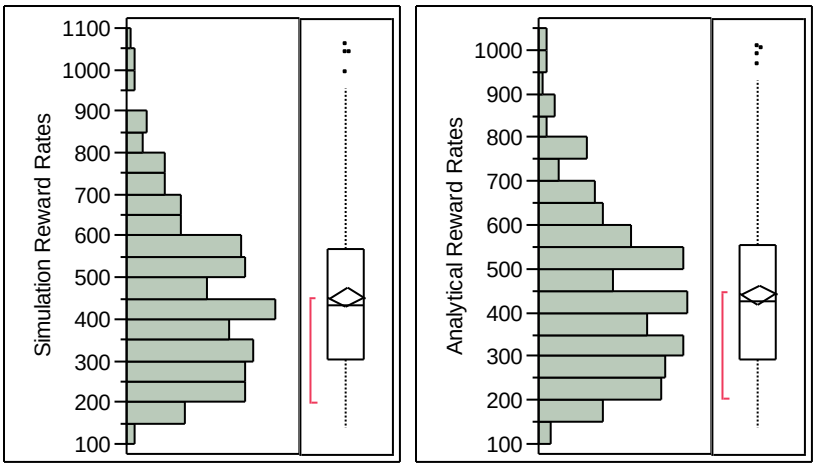

Figure 7-9: Histogram and Box Plot Diagrams of Analytical vs. Simulation Reward Rates under Uniform Distribution.

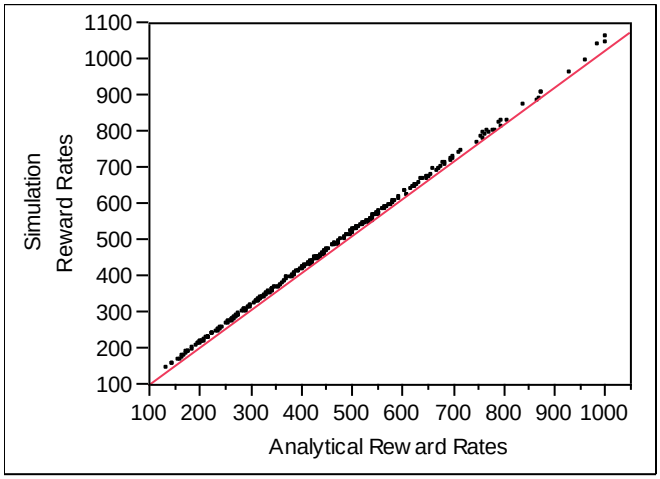

Figure 7-10: Bivariate Fit of Analytical vs. Simulation Reward Rates under Uniform Distribution.

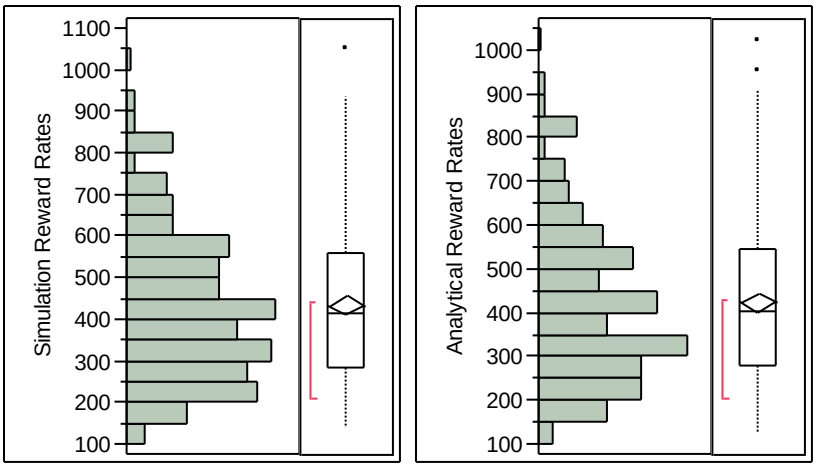

Figure 7-11: Histogram and Box Plot Diagrams of Analytical vs. Simulation Reward Rates under Hyper-Exponential Distribution.

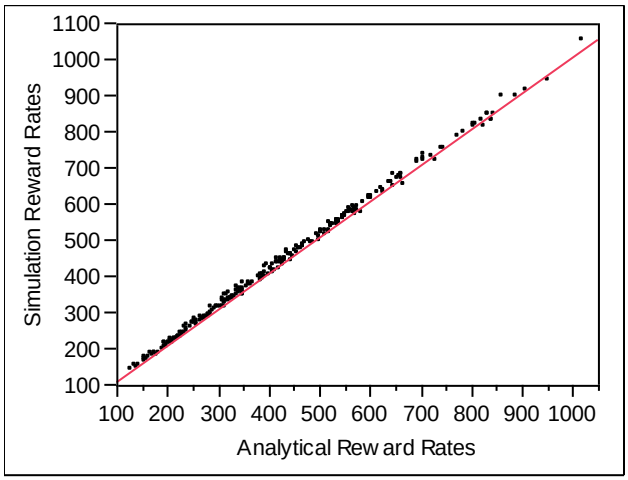

Figure 7-12: Bivariate Fit of Analytical vs. Simulation Reward Rates under Hyper-Exponential Distribution.

The resulting histogram/box plot and the bivariate fit line for simulation vs. analytical rewards rates are shown in Figure 7-6 and Figure 7-8 for the normal distribution case (correspondingly Figure 7-9 and Figure 7-10 for the uniform distribution case, and Figure 7-10 and Figure 7-12 for the hyper-exponential distribution case). In all three cases, we see that again both the histogram and box plot show similar characteristics. Also, the fit line again passes through the origin with a slope of 1 indicating a strong correlation. In

all three cases, the Spearman's ρ and probability>$|\rho|$ are calculated as 1 and 0, 0.9998 and 0, 0.9985 and 0, thus supporting a perfect positive linear relationship between simulation and analytical reward rates for normal, uniform, and hyper-exponential distribution cases respectively. We have observed a strong and significant correlation for simulation vs. analytical reward rates. In conclusion, the sensitivity analysis conducted provides strong evidences to support Hypothesis 1 and that the results are insensitive to the probability distribution of the cell residence time and call duration.

CHAPTER 8 Comparative Analysis of a Class of Call Admission Control Algorithms for Reward Optimization with QoS Guarantees

In this Chapter, we compare and contrast partitioning-threshold hybrid CAC (discussed in Chapter 5), spillover CAC (discussed in Chapter 6), and elastic threshold-based CAC (discussed in Chapter 7). We report their performances in terms of solution optimality and solution efficiency with respect to baseline CAC algorithms, including partitioning CAC and threshold-based CAC. We also provide a tradeoff analysis of solution optimality vs. solution efficiency and provide physical explanations to the analysis results to guide research on CAC algorithm design and analysis.

8.1. Test Environment

The input parameters are C, λ^1_h, μ^1_h, λ^1_n, μ^1_n, λ^2_h, μ^2_h, λ^2_n, μ^2_n, ν^1, ν^2, k^1, k^2, B^1_ht, B^2_ht, B^1_nt, and B^2_nt. We set C=80, k^1=4 and k^2=1. Analytical solutions are obtained by evaluating the performance models developed (discussed in Chapters 5-7) for these CAC algorithms with test scenarios (developed in Chapter 4) given as input. Simulation results are obtained by running the simulator (developed in Chapter 4) through the same set of test scenarios. Recall that there are altogether 27 test scenarios generated resulting from a combination of 3 arrival and departure rates for high priority calls (high, medium, low), 3

arrival and departure rates for low priority calls (high, medium, low), and 3 separate locations. We repeat each test scenario for several mobile user populations to generate varying aggregate arrival and average departure rates and to test the effect of workload on these CAC algorithms. For lower populations all CAC schemes are able to generate a legitimate solution in which QoS constraints are satisfied. However, when the workload increases, certain CAC algorithms may fail to generate a legitimate solution as they fail to allocate channel resources that will satisfy QoS constraints.

Table 8-1: Performance improvement obtained by Elastic Threshold CAC.

Comparison vs.	Partitioning	Spillover	Hybrid	Threshold-based
Num. of supported MUs	(32.9% - 41.4%)	(1.8% - 3.7%)	(2.0% - 4.5%)	(2.4% - 5.1%)
Max Reward Rate	(31.3% - 40.7%)	(1.8% - 3.5%)	(1.9% - 4.1%)	(2.1% - 4.8%)

8.2. Solution Optimality Comparative Analysis

We compare solution optimality characteristics of CAC algorithms developed in terms of two metrics: (a) the maximum number of mobile users (representing workload) that the 7-cell system is able to support with QoS satisfaction, and (b) the maximum "reward rate" obtainable with QoS satisfaction. In all 27 test scenarios, elastic CAC performs the best while partitioning CAC performs the worst. On the other hand, both spillover CAC and hybrid CAC perform as well as elastic CAC, but in general perform the second or the third. Hybrid CAC always performs better than threshold-based CAC because it encompasses both partitioning and threshold-based CAC characteristics. Table 8-1 summarizes the performance gain (effect size) of elastic CAC when compared with other CAC algorithms.

The performance gain is expressed in terms of the percentage range over which

elastic CAC performs better than others with 95% confidence level based on the 27 test scenarios. With a 95% confidence level, elastic CAC performs about 37.2% ± 4.24 better than partitioning CAC, 3.7% ± 1.35 better than threshold-based CAC, 3.3% ± 1.35 better than hybrid CAC and 2.7% ± 0.95 better than spillover CAC in the number of mobile users supported. Similarly, with a 95% confidence level, elastic CAC performs about 36.0% ± 4.69 better than partitioning CAC, 3.4% ± 1.36 better than threshold CAC, 3.0% ± 1.13 better than hybrid CAC, and 2.6% ± 0.85 better than spillover CAC in the maximum reward rate obtainable. In summary, we observe that elastic, spillover, hybrid, and threshold-based CAC perform significantly better than partitioning CAC and that elastic, spillover, hybrid, and threshold-based CAC are able to support high arrival rates by better utilizing channel resources via sharing.

To better visualize these results, we draw two notched-box plot diagrams as shown in Figure 8-1 and Figure 8-2 for the maximum number of mobile users supported and the maximum reward rate obtained by various CAC algorithms, respectively. The thick line in a box plot represents the median value in all test cases. The bottom and top lines of the shaded box represent reward rate values below 25% and 75%, respectively. The horizontal bar at the top represents the largest non-outlier observation. Notches around medians indicate the uncertainty about the medians obtained. We see that these notched-box plots confirm the above observation. In both notched-box plots we see that threshold, spillover, hybrid and elastic CAC algorithms perform better than the partitioning CAC algorithm in terms of both solution optimality metrics. We also see that the performance difference among threshold, spillover, hybrid and elastic CAC algorithms is less than 5% in both plots.

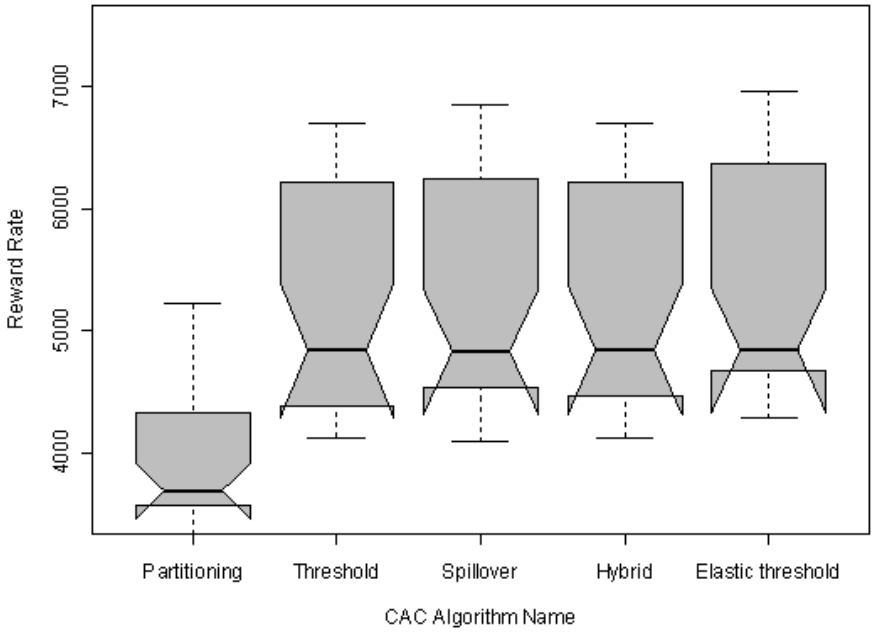

Figure 8-1: Notched-Box Plot Diagram for the Maximum Number of Mobile Users Supportable by various CAC algorithms.

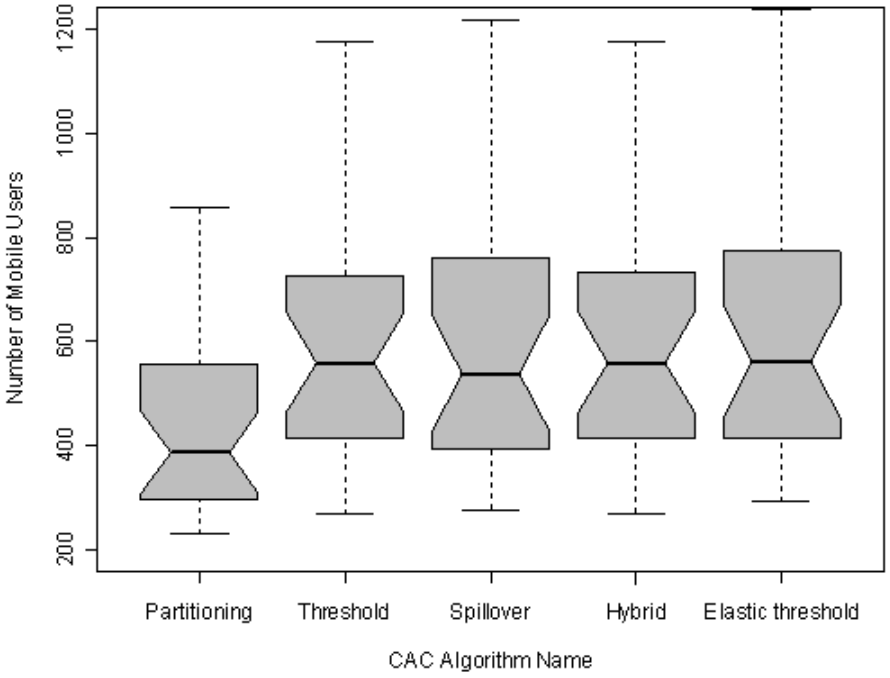

Figure 8-2: Notched-Box Plot Diagram for the Maximum Reward Rate Obtainable by Various CAC Algorithms.

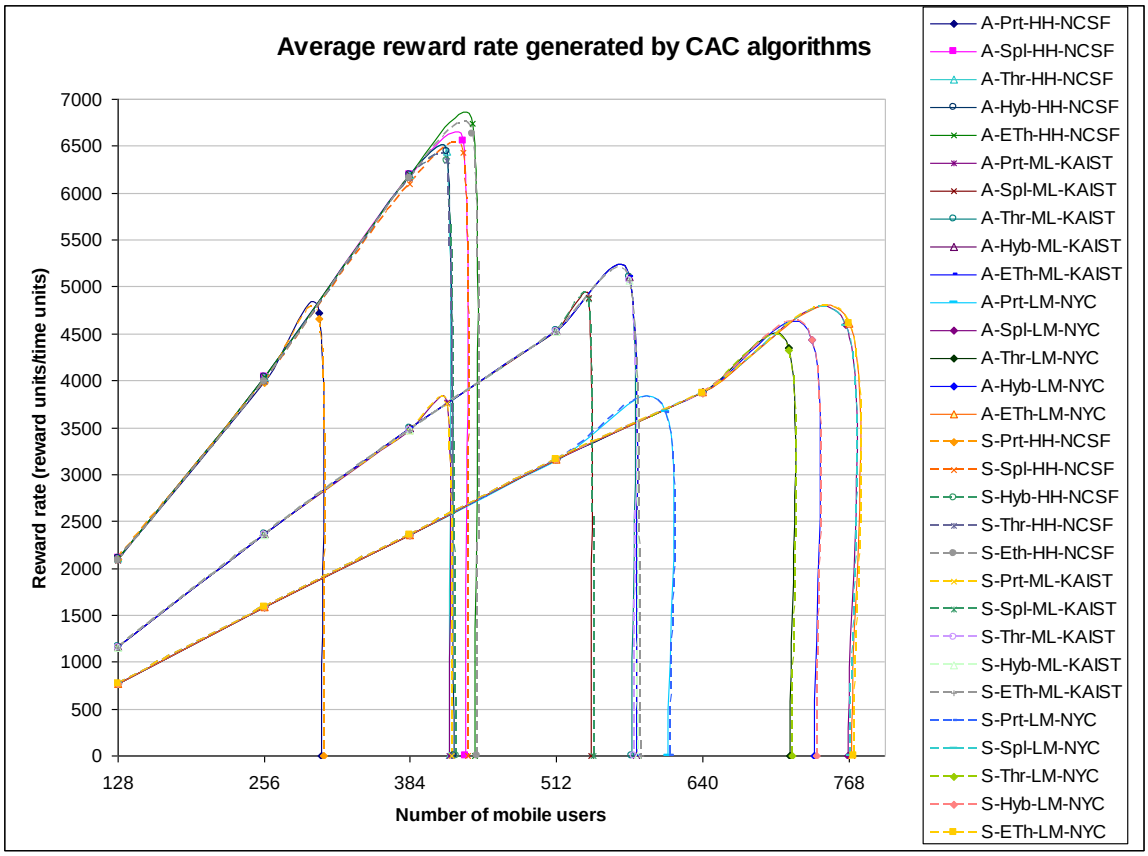

Figure 8-3: Reward Rate Generated by CAC Algorithms.

We summarize the results further in Figure 8-3 for 3 test scenarios selected to illustrate solution optimality. The legend used is in X-YYY-ZZ-L format where X indicates the type of the result, viz., A – analytical, S – simulation; YYY stands for the abbreviation of the CAC; ZZ stands for the test scenario used, e.g., HH means high class 1 and high class 2 arrival/departure rates, LM means low class 1 and medium class 2 arrival/departure rates, ML means medium class 1 and low class 2 arrival/departure rates; the last symbol L indicates the location of the mobility trace. We obtain the highest system reward when the arrival/departure rates are high for both class 1 and class 2. In this test scenario, spillover CAC performs better than hybrid and threshold-based CAC and the highest reward rate is obtained despite a lower number of mobile units is

supported compared with the other two test scenarios. When the system experiences medium class 1 and low class 2 arrival/departure rates, it generates the second highest reward rates and elastic, hybrid and threshold-based CAC perform better than spillover CAC.

In terms of the number of mobile users (representing workload) the system is able to support with QoS satisfaction where scenario 2 performs better than scenario 1 but is worse than scenario 3. In test scenario 3, spillover CAC performs comparable to elastic CAC while hybrid CAC performs relatively poorly compared to these two algorithms. Threshold-based CAC performs the worst among these four comparable algorithms. This behavior shows that the existence of a pair of thresholds introduced in elastic CAC greatly improves the performance especially when the low priority call arrival rate is high. Similarly having additional partitions serving a single call type in hybrid CAC improves the performance of threshold-based CAC. We see from the figure in all cases tested, the generated reward rates from simulation are very close to the reward rates obtained analytically. The simulation results validate our analytical results.

Figure 8-3 also presents the correlation between two solution optimality metrics which we have used to evaluate CAC algorithms. For each test case the maximum reward rate obtainable by each CAC algorithm increases with the increase of the maximum number of mobile unit supported. Again we see that under both metrics partitioning CAC performs the worst in terms of solution optimality. On the other hand, elastic threshold-based CAC perform best among all CAC algorithms, followed by spillover, hybrid and partitioning CAC algorithms.

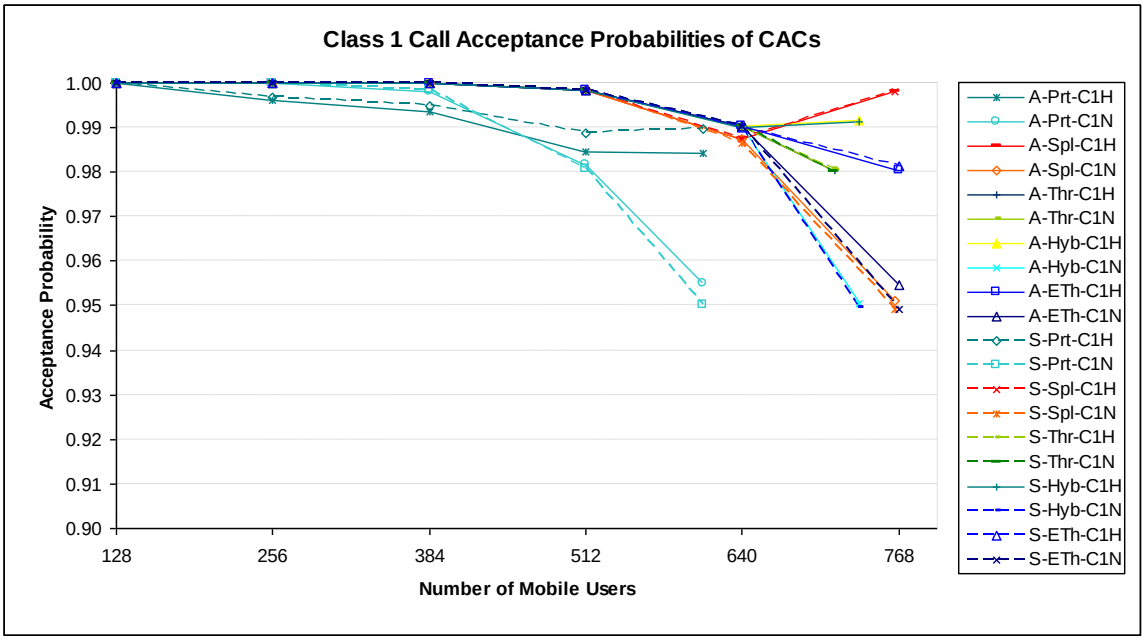

Figure 8-4: Class 1 Call Acceptance Probabilities of CAC algorithms.

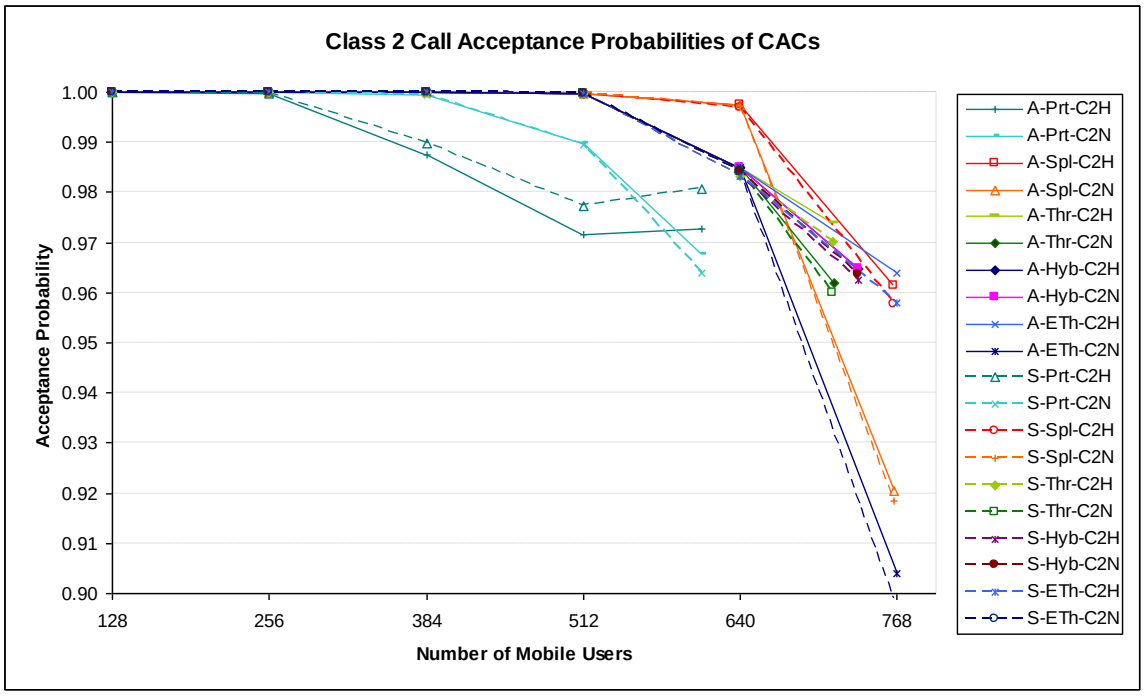

Figure 8-5: Class 2 Call Acceptance Probabilities of CAC algorithms.

Another performance metric of interest is the acceptance probability. Figure 8-4 and Figure 8-5 show the class 1 and class 2 call acceptance probabilities under various

CAC algorithms in the third test scenario. Both spillover CAC and elastic CAC are able to satisfy the stringent QoS constraint of class 1 handoff calls by rejecting class 1 new calls, and class 2 calls as much as necessary as required by their QoS constraints. Threshold CAC fails to find proper threshold values or partition allocations to reject less than 4% of class 2 calls, and thus fails to satisfy class 2 QoS constraints after the system has more than 715 mobile users. On the other hand, elastic CAC is able to accommodate 53 more mobile users by adjusting high and low thresholds of class 1 and class 2 new and handoff calls. Similarly, hybrid CAC improves the performance of threshold-based algorithm and supports 736 mobile users by rejecting 5% of class 1 new calls. Finally, spillover CAC can support up to 765 mobile users by reserving enough channels in the second, third and fourth partitions. In Figure 8-4 and Figure 8-5, we also show simulation results to verify the accuracy of our analytical results. Again, the acceptance rates observed via simulation validate our analytical results.

The last performance metric of interest is *channel efficiency* represented by the channel utilization metric which measures the degree to which channel resources are efficiently utilized. Figure 8-6 shows the notched-box plot diagram for channel utilization by various CAC algorithms. Overall we see that CAC algorithms based on resource partitioning, namely, pure partitioning CAC and spillover CAC, perform worse than CAC algorithms based on setting thresholds, namely, threshold-based CAC, hybrid CAC, and elastic CAC. In particular, partitioning CAC performs very poorly compared with other CAC algorithms in terms of channel utilization. It fails to utilize channel resources to its maximum capacity because resources are not shared by means of strict resource partitioning. Spillover CAC is also based on resource partitioning. However since it

allows partitions to be shared among service classes, it performs much better than the partitioning CAC algorithm in terms of channel utilization. Among algorithms based on setting thresholds, elastic CAC performs slightly better than threshold-based and hybrid CAC algorithms. We attribute this to the use of double thresholds to better limit the traffic generated by each service call type. We conclude that resource sharing through either partitioning or setting thresholds greatly improves channel utilization.

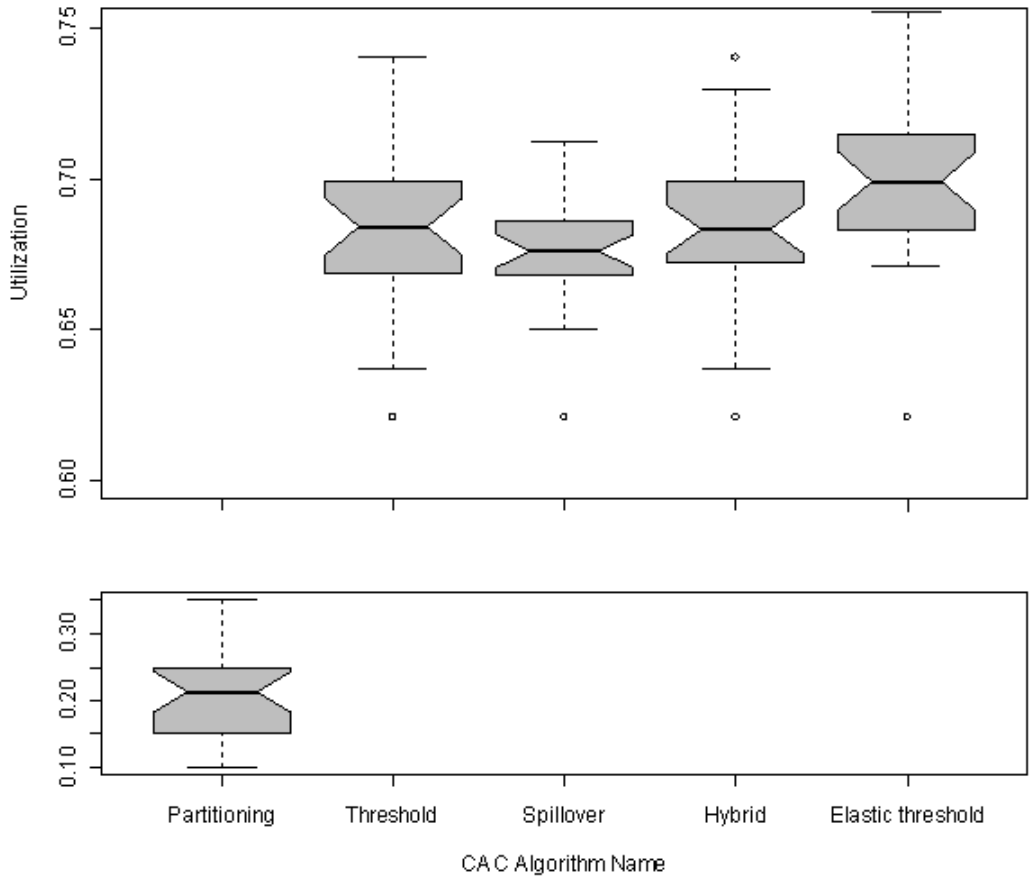

Figure 8-6: Notched-Box Plot Diagram for Channel Utilization.

We examine the relation between channel utilization and reward rate obtainable with QoS guarantees by means of an X-Y relation graph as shown in Figure 8-7. Partitioning CAC is not on the graph because the channel utilization and reward rate

generated are far too small compared with others. We see that the ranking order of CAC algorithms that produce the best reward rate with QoS guarantees is about the same as that for producing the best channel utilization, so there is a positive relation between channel utilization and reward rate. We attribute this positive relation to the resource sharing capabilities of CAC algorithms, the effect of which is especially pronounced in the elastic CAC algorithms which has the best channel utilization as well as the best reward rate obtainable with QoS guarantees. In general the performance order of CAC algorithms in terms of solution optimality is preserved when the objective function is positively and linearly related to the reward objective function considered in the dissertation research.

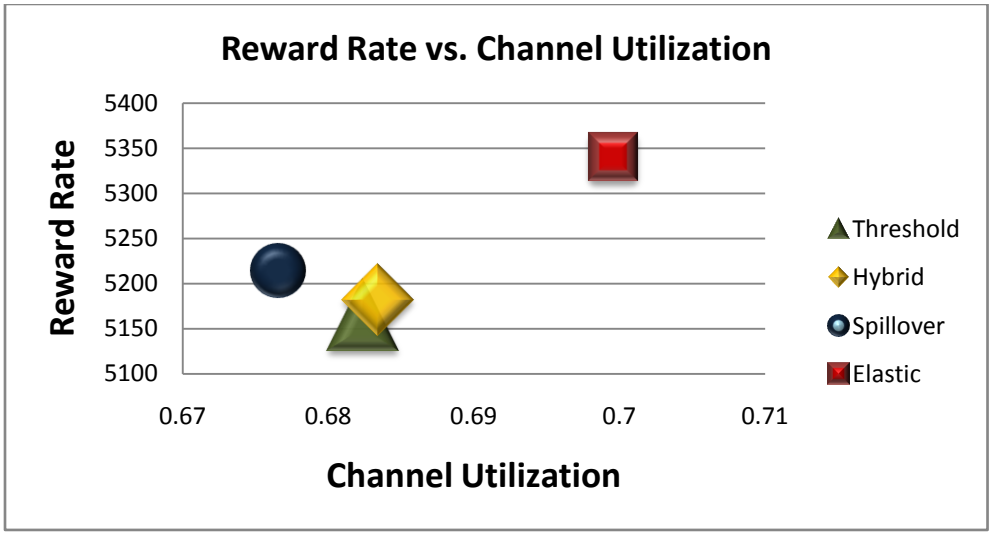

Figure 8-7: Relation of Reward Rate vs. Channel Utilization.

8.3. Solution Efficiency Comparative Analysis

We compare solution efficiency characteristics of partitioning-threshold hybrid CAC vs. spillover CAC and elastic threshold-based CAC in terms of the time elapsed for a CAC algorithm to find the optimal solution that yields the maximum reward rate

obtainable with QoS guarantees. We perform the comparative analysis in two ways: theoretical complexity analysis and experimental evaluation based on the 27 test scenarios. Note that *solution efficiency* refers to the efficiency of CAC algorithms to find optimal solutions, as opposed to *channel efficiency* which is a performance metric as a result of applying CAC algorithms. We have evaluated channel utilization for channel efficiency considerations in Section 8.2.

8.3.1. Solution Efficiency Complexity Analysis

Here we perform complexity analysis of CAC algorithms developed and compare their solution complexity in terms of the number of combinations evaluated before an optimal solution is found. Note that each combination represents a potential solution which needs to be evaluated to see if it satisfies the QoS constraints. Also note that for all threshold-based CAC algorithms, the evaluation of a combination is done by executing the associated SPN performance model. Also all CAC algorithms except partitioning CAC are based on heuristic search which first determines the minimum number of channels needed by each service class, so in case the minimum number of channels needed by a service class is greater than C, the search fails and exits. Thus, the best solution complexity is simply O(0). The worst case solution complexities of partitioning, threshold-based, spillover, hybrid, and elastic CAC algorithms are given below.

Partitioning CAC performs exhaustive search to find a solution which maximizes the reward rate while satisfying QoS constraints. It evaluates $C(C+N-1, N-1)$ combinations where C is the combination operator, C is the number of channels and N is the number of service call types (N=4 in our case). Therefore, the solution complexity of partitioning CAC is $O((C+N)^N)$.

Threshold-based CAC searches for an optimal solution in two phases. In the first phase, it searches for a legitimate solution. It initially sets the threshold of every service class to C and then it decreases the thresholds of all service classes except for the one with most stringent QoS requirement until it finds a solution or it can no longer satisfy the QoS requirement of the service class with most stringent QoS constraints. This phase requires no more than C(N-1) combinations to be evaluated. In the second phase, threshold-based CAC performs greedy search based on the solution found in the first phase until the current solution cannot be improved further. In the worst case the number of combinations to be evaluated is proportional to C^N. Therefore this algorithm has a worst case solution complexity of $O(C^N)$.

Spillover CAC also has two phases where the first phase searches for a legitimate solution and the second phase optimizes the solution. In the first phase, it determines the minimum number of channels to satisfy the QoS constraints of each service class. This phase most of the time generates a solution after at most C combinations have been evaluated. However, in case of failing to satisfy the QoS constraints, it needs to rollback and modify the number of channels allocated to the previous partition to continue the search. Therefore, in the worst case the solution complexity of the first phase is $O(C^N)$. In the second phase spillover CAC performs greedy search. The worst case complexity of the 2^{nd} phase is also $O(C^N)$. Consequently, the worst case solution complexity of spillover CAC is $O(C^N)$.

The hybrid CAC algorithm is a hybrid of partitioning and threshold-based CAC. It performs two searches to determine an optimal solution. In the first search it looks for partition combinations that would maximize the reward rate. In this step it also tries Δ

threshold values around the current solution found to see if it can further improve the quality of the solution. Thus, this step has a worst case solution complexity of $O(C^N\Delta^N)$. The second search performs exactly the same computation as in threshold-based CAC which has a computational complexity of $O(C^N)$. Therefore, the worst case computational complexity of hybrid CAC is $O(C^N\Delta^N)$.

Finally, elastic CAC consists of three phases. The first phase is similar to the first phase of threshold-based CAC. Consequently, the first phase has a solution complexity of O(CN). The second phase is executed when the first phase fails to find a legitimate solution. This phase evaluates Δ high threshold combinations for each low threshold combination. Thus it has a worst case solution complexity of $O(C^N\Delta^N)$. The third phase of elastic is called only if the first two phases yield a legitimate solution. This phase performs greedy search to improve the solution quality. For each new optimal solution it first tries Δ lower thresholds and then Δ higher thresholds of each service type. In the worst case it tries $O(C^N)$ solutions and for each new solution it tries $O(\Delta^N)$ combinations. Therefore, elastic CAC also has a worst case solution complexity of $O(C^N\Delta^N)$.

In conclusion, the worst case solution complexities of partitioning, threshold-based, spillover, hybrid, and elastic CAC algorithms are $O((C+N)^N)$, $O(C^N)$, $O(C^N)$, $O(C^N\Delta^N)$, and $O(C^N\Delta^N)$.

Table 8-2: Solution Efficiency of CAC Algorithms.

Solution Efficiency Metric	Partitioning	Threshold-based	Hybrid	Spillover	Elastic Threshold-based
Elapsed Time (sec)	(0.31 - 0.36)	(892 - 1724)	(426 - 1496)	(11.71 – 71.26)	(1443 – 5429)

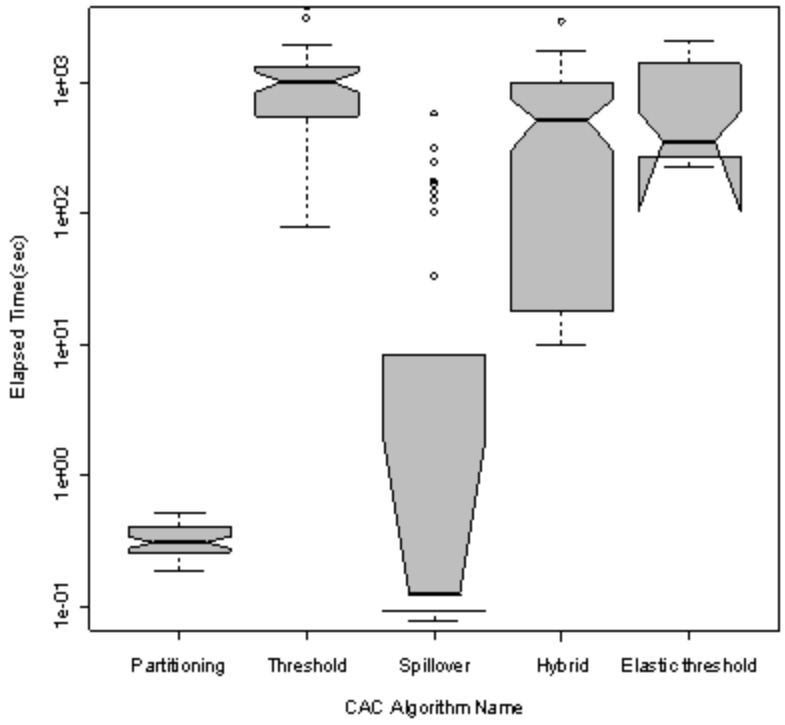

Figure 8-8: Notched-Box Plot Diagram for Elapsed Time.

8.3.2. Solution Efficiency Experimental Evaluation

Below we evaluate search efficiency experimentally using the 27 test scenarios. Table 8-2 shows a range of actual time elapsed to find optimal solutions by various CAC algorithms. Figure 8-8 shows a notched-box plot diagram for the elapsed time. The notched-box plot clearly indicates that CAC algorithms based on resource partitioning principles (namely partitioning CAC and spillover CAC) perform significantly better than CAC algorithms based on threshold settings (namely threshold-based CAC and elastic CAC), as well as better than hybrid CAC algorithms. Pure partitioning CAC performs the best among all and also performs better than spillover CAC because of a smaller solution space to search for the optimal solution than spillover CAC. Elastic CAC performs the

worst because of double thresholds being used, instead of a single threshold, in admitting service calls in each call type.

8.4. Solution Efficiency vs. Solution Optimality

From the analyses in Sections 8.2 and Section 8.3, we see that while the use of thresholds greatly improves solution optimality because of resource multiplexing, solution efficiency is sacrificed. Consequently, there is an intrinsic tradeoff in solution optimality vs. solution efficiency. The analyses performed thus far do not answer the question of which CAC algorithm to choose when given a set of input conditions. For example, if the speed of the algorithm is the highest priority, partitioning CAC might be an option since it performs the best in solution efficiency despite it performs the worst in solution optimality.

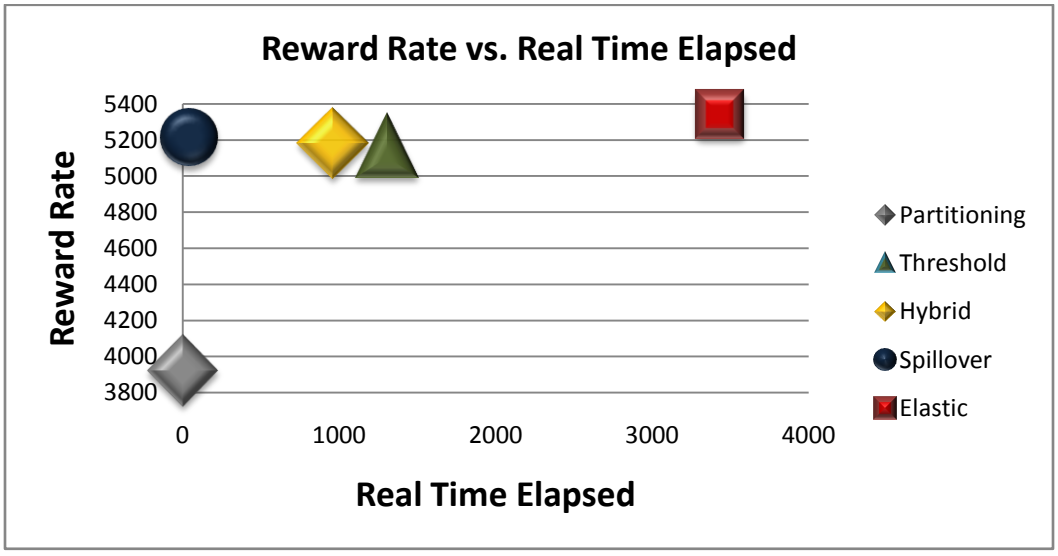

Figure 8-9: A Relation Graph between Solution Optimality (Reward Rate) vs. Solution Efficiency (Elapsed Time).

Figure 8-9 is an X-Y graph showing the relation between solution optimality (the reward rate obtainable with QoS Guarantees) and solution efficiency (the elapsed time to

find the optimal solution) for various CAC algorithms. Elastic CAC performs the worst in solution efficiency while it performs the best in solution optimality. The use of double thresholds increases the search space and also improves solution optimality at the expense of solution efficiency. Spillover CAC performs well in both solution optimality and solution efficiency since it allows resource sharing based on resource partitioning. Thus, it is a reasonable choice if both solution optimality and solution efficiency are important. Hybrid CAC and threshold-based CAC are in the next group with about the same comparable performance in both solution optimality and solution efficiency, with hybrid CAC performs slightly better. Finally, partitioning CAC performs the best in solution efficiency because of strict resource partitioning, but performs significantly poorly compared to all other CAC algorithms in solution optimality because of failing to allow resource sharing.

8.5. Summary

In this chapter we have performed and presented a comparative analysis to compare and contrast a class of CAC reward optimization algorithms in terms of solution optimality metrics including the maximum number of mobile users that the system is able to support and the maximum reward rate obtainable with QoS guarantees. We have shown that elastic-threshold CAC performs the best, while spillover CAC, hybrid CAC and threshold-based CAC in general perform the second, the third, and the fourth, respectively, and partitioning CAC performs the worst. In all 27 test scenarios that we have evaluated, spillover, hybrid and threshold-based CAC algorithms perform reasonably well when the mobile user population is moderate. However, as the user population increases and thus the aggregate traffic increases, they fail to generate a

legitimate solution to satisfy QoS constraints of service classes. We reported that elastic CAC performs about 2.7%, 3.0%, 3.6%, and 36.5% better than spillover, hybrid, threshold, and partitioning CAC algorithms, respectively, in both metrics. We attribute the superiority of elastic CAC to its "elastic-threshold" functionality which is capable of leveraging the pair of threshold values to reject traffic generated by lower priority service calls. We also showed that spillover CAC performs almost as well as elastic CAC when class 1 and class 2 arrival rates are low and medium. This indicates that the spillover technique is still reasonably powerful especially when class 2 call arrival rates are relatively higher than those of class 1.

We have also performed solution efficiency analysis for the elapsed time to find optimal solutions by various CAC algorithms. We conducted theoretical complexity analysis as well as experimental evaluation. The results indicate that partitioning CAC and spillover CAC based on resource partitioning principles perform significantly better than threshold-based CAC, hybrid CAC and elastic CAC based on threshold settings. In particular elastic CAC, while performing the best in solution optimality, performs the worst in solution efficiency due to the use of double thresholds and a very large solution space to find optimal solutions. Partitioning CAC is exactly the opposite.

Finally we investigated the tradeoff between solution optimality vs. solution efficiency of various CAC algorithms. We presented an X-Y relation graph which clearly dictates this tradeoff and allows the system designer to select the best CAC algorithm to be used depending on the importance of solution optimality vs. solution efficiency.

CHAPTER 9 **Applications**

The CAC algorithms and the application framework developed can be applied to any application that concerns reward optimization and QoS satisfaction. In this chapter we demonstrate the applicability with two applications. The first application is the "optimal pricing" application typically performed by service providers, treating "revenue" as a form of the "reward" objective function. Utilizing a simple demand-pricing formula to predict service demands as prices vary, we apply CAC algorithms developed for reward optimization to determine optimal pricing for multiple service classes such that the overall reward obtained by the system is maximized while QoS constraints of users in multiple service classes are satisfied. This application is described in Section 9.1. The second application is the intelligent switch routing application performed by a router to determine the most economical way to select differently priced trunks in multiple paths for satisfying routing requests from clients with varying QoS requirements and constraints while maximizing the net revenue received by the switch. This application is described in Section 9.2. This chapter is largely based on our work published in [68] [6] and [69] [7].

[6] © 2006 IEEE. Reprinted, with permission, from O. Yilmaz and I.R. Chen, "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*, Minneapolis, MN, July 2006, pp. 605 – 612.
[7] Reprinted from O. Yilmaz and I.R. Chen, "Utilizing call admission control for pricing optimization of multiple service classes in wireless cellular networks," accepted to *Computer Communications*, Nov. 2008. Copyright 2008, with permission from Elsevier.

9.1. Optimal Pricing Determination Application

A service provider typically adjusts pricing periodically. Once a "global" optimal pricing is derived, it would stay static for a period of time, allowing users to be charged with the same rate while roaming. We utilize CAC algorithms developed to analyze a pricing scheme that correlates service demand with pricing, allowing service providers to periodically determine optimal pricing under which the system "revenue" is maximized while guaranteeing that QoS constraints of multiple service classes are satisfied. In order to determine best pricing, we adopt the following demand-pricing relation function from empirical studies [1], [37], [55] to predict demand changes when we vary pricing:

$$\lambda^i = a^i (v^i)^{-\varepsilon^i} \tag{50}$$

Here λ^i and v^i denote the arrival rate and pricing of service *i*, respectively, while a^i and ε^i are constants correlating λ^i and v^i. The elasticity constant ε^i determines the effect of pricing change. A value greater than 1 predicts that decreasing the price would generate higher reward, while a value less than 1 predicts that increasing the price would generate higher reward. The elasticity ε^i value can be determined by analyzing statistical data collected. However, ε^i should be greater than 1 for network services as suggested by [1], [37], reflecting the fact that lower pricing would stimulate higher demand for the service provided to possibly result in higher reward. The proportionality constant a^i differs from cell to cell and can be calculated from Equation (50) once the reference arrival rates (λ^i), the current price (v^i) and the elasticity constant ε^i are known through statistical analysis.

For ease of exposition, we consider a case in which the system supports two service classes, 1, and 2. Thus, the total reward R_T generated by each cell is the sum of the reward generated by each service class, that is,

$$R_T = R^1_h + R^1_n + R^2_h + R^2_n \quad (51)$$

Here R^1_h represents reward earned from servicing class 1 handoff calls, R^1_n represents reward earned from servicing class 1 new calls, and so on. Here we note that pricing of a service class implicitly influences the arrival rate of that service class based on Equation (50), which in turn affects the reward obtainable by the system. If we decreased the price of a service class, the arrival rate of that service class would increase, and vice versa. Since all service classes in a cell share a limited set of bandwidth channels, if we lowered pricing of all service classes with the intent to increase reward, at some point QoS constraints would be violated because of system overload. Thus, it is a combinatorial problem to search for optimal pricing to maximize the system reward while satisfying QoS of all service classes.

In determining best pricing, we adopt the demand-pricing relation given in Equation (50) to predict demand changes as we change pricing. We calculate the maximum reward obtainable as a result of applying a CAC algorithm, when given a set of parameter values characterizing the workload of each service class i, namely, λ^i_n, μ^i_n, λ^i_h, μ^i_h, v^i determined by each cell by applying the workload characterization algorithm developed in Chapter 4. The reward obtainable is calculated based on Equation (51), which will be used as an objective function to guide the search of optimal pricing.

More specifically for each service class S^i we obtain a 5-tuple "reference" parameter values (λ^i_n, μ^i_n, λ^i_h, μ^i_h, v^i) as the initial state for class S^i. We determine a price range [$v^{i,min}$, $v^{i,max}$] for S^i such that $v^{i,min}$ and $v^{i,max}$ are minimum and maximum rewards received by the system per time unit acceptable by the service provider and customer

base. Then, for each service class S^i, we determine β^{i+1} reward rates in increment of δ^i between $v^{i,min}$ and $v^{i,max}$ by using the following formulas:

$$v^{i,j} = v^{i,min} + j \cdot \delta^i \qquad i \in \{1, \ldots, n\} \& \; j \in \{0, \ldots, \beta^i\} \tag{52}$$

$$\delta^i = \left(v^{i,max} - v^{i,min}\right) / \beta^i \tag{53}$$

The number of possible combinations to search for optimal pricing, denoted by η, is thus equal to:

$$\eta = (\beta^1+1)\,(\beta^2+1)\,\ldots\,(\beta^n+1) \tag{54}$$

When the price of class S^i is changed, the new call arrival rate of S^i is changed as predicted by (50). That is, the new call arrival rate of S^i for the j^{th} price increment would be given by:

$$\lambda_n^{i,j} = a^i \cdot \left(v^{i,j}\right)^{-\varepsilon^i} \tag{55}$$

Here we note that ε^i for class S^i remains the same while it differs from one cell to another. The predicted handoff call arrival rate of S^i for the j^{th} price increment is given by:

$$\lambda_h^{i,j} = f^i \cdot \lambda_n^{i,j} \tag{56}$$

where f^i denotes the ratio between the current handoff call arrival rate and new call arrival rate of service class S^i, given by:

$$f^i = \lambda_h^{i,\,current} \Big/ \lambda_n^{i,\,current} \tag{57}$$

Note that f^i is a constant because handoff requests into a reference cell are from the neighboring cells. Equation (56) holds as long as an increase or decrease of the new call arrival rate in neighboring cells generates a proportional increase or decrease of the handoff rate into the reference cell.

For each possible price of class S^i we represent it with a 5-tuple ($\lambda^{i,j}_n$, μ^i_n, $\lambda^{i,j}_h$, μ^i_h, $v^{i,j}$). Suppose there are *n* service classes. Then, a set of *n* 5-tuples, one for each server class, would present one potential price combination solution to explore. The evaluation of each potential solution involves the execution a CAC algorithm to calculate maximum reward obtainable while satisfying QoS constraints of multiple classes. Logistically, the service provider can have each cell independently collect statistical data periodically to estimate reference arrival and departure rates of new/handoff calls of various service classes in the cell based on the workload characterization algorithm described in Chapter 4. Each cell then determines the proportionality constant a^i for each service class by applying Equation (51) based on current pricing, the arrival rate, and the elasticity constant ε^i of each service class. Later each cell determines new/handoff call arrival rates for a range of "future" pricing as described above for each service class also based on Equation (50). Finally, for *each* candidate price combination, each cell calculates optimal reward obtainable with QoS guarantees utilizing a CAC algorithm developed in Chapter 4. The optimal settings for all future price combinations are then summarized in a reward table and reported to a central entity which collects and analyzes reward tables from all the cells in the system. To guarantee global QoS across all cells, the particular future price combination that satisfies QoS constraints in all of the cells while maximizing the aggregate reward would be chosen as the winner for optimal pricing.

The system provider can have each cell generate such a reward table only periodically, e.g., every 3 months, as deemed economically feasible by the service provider for changing pricing to service classes. When the end of the current period is approaching, a new set of reference arrival and departure rates of new/handoff calls of

various service classes that have been collected over the last period will be used as input to build the reward table by each cell. Thus, the reward table can be built by each cell in the background periodically. The system-wide optimal pricing per service class then can be determined by the service provider using the methodology introduced here. Such optimal pricing determined will then remain static across cells for all service classes till the next period.

The above approach is distributed in nature and will require a central entity to collect reward tables reported by individual cells, and just pick the price combination that satisfies QoS constraints in all of the cells while maximizing the aggregate reward. Another approach is based on centralized control by having the central entity guide the search such that cells will be instructed to evaluate a particular price combination only when needed until the best price combination is found. This approach has the advantage that each cell does not regenerate the entire revenue table.

The service provider may elect to have different pricing in different days of the week (e.g., weekday vs. weekend rate) or even in different times of the day (e.g., day-time vs. night-time rate). In this case the same methodology developed in the paper applies, except that it is being applied separately to each time segment. For example, if there is a distinction of weekend vs. weekday rate, then two separate sets of reference arrival and departure rates of new/handoff calls would be collected for weekend and weekday time segments, and used as input to generate optimal pricing separately for these two time segments.

Without loss of generality, we illustrate how to utilize a CAC algorithm in the application framework for the optimal pricing determination application with the

partitioning-threshold hybrid CAC algorithm (developed in Chapter 5). We also apply the baseline CAC algorithms, including partitioning and threshold-based CAC algorithms for comparison purposes. The same procedure can be applied with spillover CAC and elastic CAC algorithms.

We present numerical data for η=6x8=48 possible future price combinations by applying the revenue formulas derived for partitioning, threshold-based and partitioning-threshold hybrid CAC algorithms in Chapter 5. These possible future price combinations are relative to current pricing such that the price increment/decrement is considered acceptable to the service provider. Again we consider two classes: class 1 (real-time) and class 2 (non-real-time) with class 1 demanding more resources with higher QoS constraints than class 2. Thus, class 1 has more stringent call blocking probabilities than class 2, as well as higher pricing.

The input parameters are C, λ^1_h, μ^1_h, λ^1_n, μ^1_n, λ^2_h, μ^2_h, λ^2_n, μ^2_n, ν^1, ν^2, a^1, a^2, ε^1, ε^2, k^1, k^2, B^1_ht, B^2_ht, B^1_nt, and B^2_nt. We set C=80 channels, k^1=4 and k^2=1 for a typical cell in mobile wireless networks to service real-time and non-real-time traffic such that there are 80 channels in the cell with a class 1 call (real-time) consuming 4 channels and a class 2 call (non-real-time) consuming 1 channel. We assume that the statistical data collected for the reference cell would provide λ^1_h=5.0, μ^1_h=1.0, λ^1_n=2.0, μ^1_n=1.0, λ^2_h=4.4, μ^2_h=1.0, λ^2_n=4.4, μ^2_n=1.0 per minute, and the current pricing is ν^1=80 cents/min, ν^2=12 cents/min. Similarly, we set elasticity constants to values greater than 1, ε^1=1.3 and ε^2=1.7 for class 1 and class 2 calls, respectively. We apply Equation (56) to calculate proportionality constants a^1=600 and a^2=300 for class 1 and class 2 calls, respectively for our reference cell. We vary service prices in the range [50, 100] for ν^1 and [6, 20] for ν^2 and the

resulting call arrival rates are calculated by using Equations (56) and (57).

Figure 9-1, Figure 9-2, and Figure 9-3 show the maximum revenue obtained by partitioning, threshold-based and hybrid CAC algorithms. Class 1 and class 2 pricings are shown on the Y and X coordinates, respectively, in the unit of cents/min. The maximum revenue obtainable by a legitimate solution is shown on the Z coordinate also in the unit of cents/min.

Figure 9-1 indicates that the revenue obtainable increases as the anticipated arrival rate increases as a result of lowering the prices, as long as the QoS constraints can still be satisfied. Nevertheless, as λ^1_h and λ^1_n increase to 2.4 and 6.0 for v^1=70.0, the partitioning CAC algorithm fails to yield a legitimate solution because the workload is too heavy to satisfy the imposed QoS constraints. Likewise as λ^2_h and λ^2_n increase to 8.7 when v^2=8.0, the algorithm fails to yield a legitimate solution. The maximum revenue is established at v^1=80.0 and v^2=10.0 as these prices result in the highest arrival rate that can be handled with QoS guarantees. The best partition reserved to handle the traffic generated for (v^1=80.0, v^2=10.0) is (C^1_h=10, C^1_n=5, C^2_h=11, C^2_n=9) while the best partition reserved to handle the traffic generated in the current system for (v^1=80.0, v^2=12.0) is (C^1_h=10, C^1_n=5, C^2_h=10, C^2_n=10). As the arrival rate of class 2 becomes higher due to the price cut of v^2 from 12.0 to 10.0, partitioning CAC in this case reduces C^2_n and increases C^2_h to satisfy the higher QoS constraint of class 2 handoff calls.

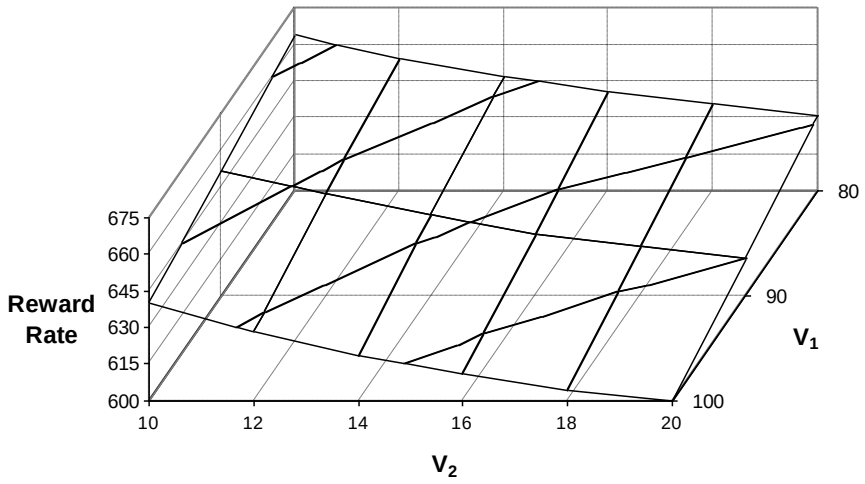

Figure 9-1: Maximum Revenue obtainable by Partitioning CAC.

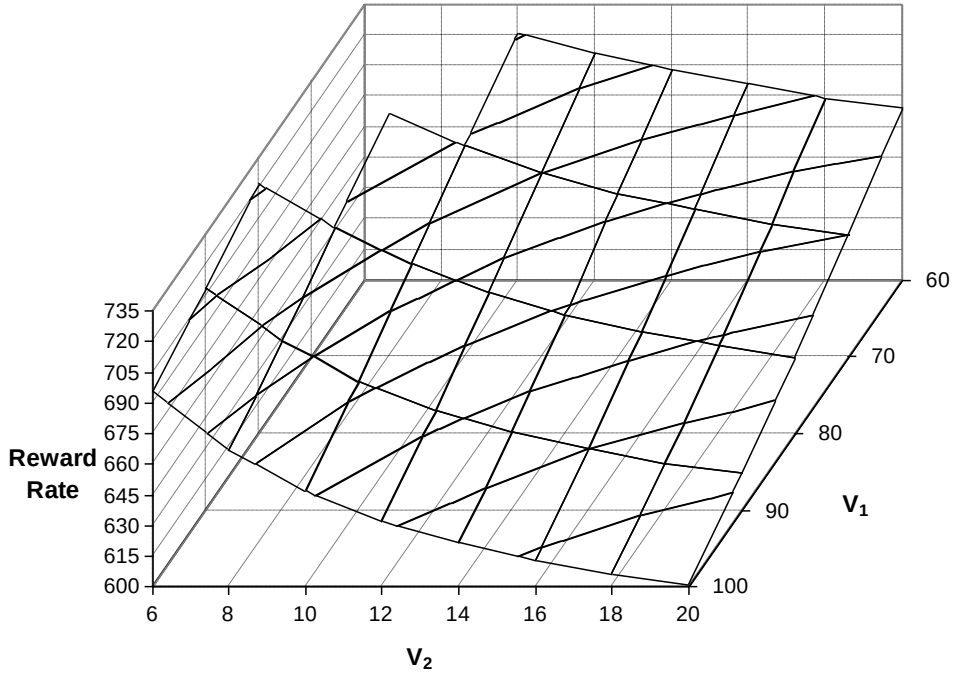

Figure 9-2: Maximum Revenue Rate obtainable by Threshold-based CAC.

Figure 9-2 shows that the highest revenue is achieved when v^1=80.0 and v^2=6.0. To satisfy QoS constraints of class 1 calls, the system applies thresholds C^1_{nT}=80, C^1_{hT}=80, C^2_{nT}=76, C^2_{hT}=76 to handle higher class 2 traffic generated for (v^1=80.0, v^2=6.0), as

opposed to all thresholds being set to 80 for the current system with (v^1=80.0, v^2=12.0). By sharing resources among service classes and controlling the effect of higher class 2 arrival rates by applying lower threshold values, threshold-based CAC performs better than partitioning CAC.

Figure 9-3 illustrates the maximum revenues obtainable with hybrid CAC with QoS guarantees as a function of v^1 and v^2. Hybrid CAC reserves ($C^1{}_h$=7, $C^1{}_n$=3, $C^2{}_h$=1, $C^2{}_n$=1, C_s=38) to handle the traffic generated for the current system with (v^1=80.0, v^2=12.0). To handle higher class 1 and class 2 arrival rates due to optimal pricing at (v_1=60 and v_2=8), it reserves ($C^1{}_h$=6, $C^1{}_n$=1, $C^2{}_h$=1, $C^2{}_n$=0, C_s=51) and applies a lower threshold to class 2 calls in the common partition. In response to a higher class 1 and class 2 arrival rates, hybrid CAC tends to increase the size of C_s partition.

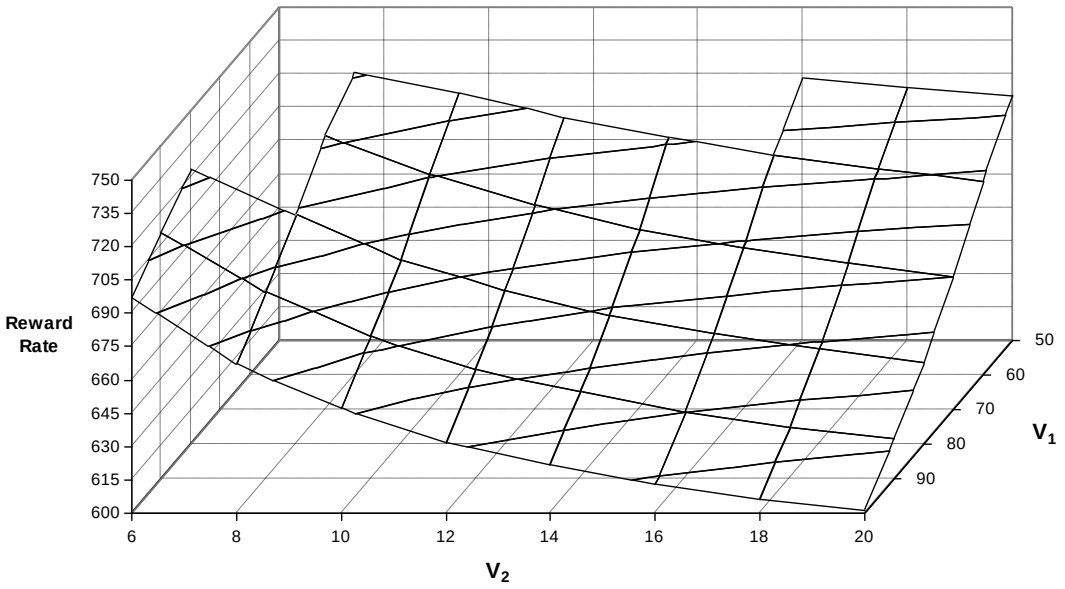

Figure 9-3: Maximum Revenue obtainable by Partitioning-Threshold Hybrid CAC.

Comparing Figure 9-3 with Figures 9-1 and 9-2, we observe that the performance of threshold-based CAC is comparable to hybrid CAC until both class 1 and class 2 arrivals

become very high (λ^1_h=7.3, λ^1_n=2.9, λ^2_h=8.7, λ^2_n=8.7 anticipated when v_1=60 and v_2=8). At these high arrival rates, threshold-based CAC fails to yield a legitimate solution compared with hybrid CAC. Again we attribute the superiority of hybrid CAC over partitioning and threshold-based CAC to the ability to optimally reserve dedicated resources for high-priority classes through fixed partitioning to reduce interference from low-priority classes, and to optimally allocate resources to the shared partition in accordance with threshold-based CAC to exploit the multiplexing power for all classes.

In order to optimize the total system revenue, each cell first follows the procedure described above to calculate the "local" maximum revenue obtainable with QoS guarantees for each possible future price combination (e.g., v^1=80.0, v^2=10.0) under consideration compared to current reference pricing (e.g., v^1=80.0, v^2=12.0). A table is generated by each cell that lists the "local" maximum revenue obtainable for each future pricing that yields a legitimate solution (i.e., the QoS constraints are satisfied). Then a merging process is used to merge tables generated by individual cells to determine "global" optimal pricing and the associated maximum revenue obtainable by the system. This process first eliminates any future price combination that fails to yield a legitimate solution for any of the cells. Then for other future price combinations for which a legitimate solution exists in all cells, the maximum revenue obtainable by the system is calculated by summing all "local" maximum revenues earned by all the cells. The future price combination that yields the maximum aggregate revenue is declared as "global" optimal pricing among all such future price combinations. Since our system excludes price combinations that cannot satisfy QoS constraints, it never generates a solution that maximizes the reward but violates QoS constraints. This optimal pricing determination

application falls within a general class of applications which concern reward optimization and QoS satisfaction for servicing multiple classes in wireless networks. These applications can be easily applied to the application framework developed in the dissertation research.

9.2. Intelligent Switch Routing Application

In this section, we apply the CAC algorithms developed to a routing application in Internet switches/routers which must route calls via trunks (channels essentially) along several candidate paths with different charge rates to reach the destination. As illustrated in Figure 9-4, switch A is connected to switch B through m paths, P_1 through P_m, each with c_{pi} trunks and a charge rate of d_{pi} per trunk usage to route a packet from A to B where $0 < i \leq m$. Routing requests come from n service classes, each with its own arrival and departure rates, QoS requirements represented by k^j (in terms of the number of trunks needed) and QoS constraint represented by B^j (in terms of the blocking probability) where $1 < j \leq n$. Suppose that the reward obtained by switch A is v^j for routing a class j call. If the switch decides to route a class j call through a trunk on path P_i then the switch will receive a reward of $v^j - k^j d_{pi}$ when the call is successfully routed through path P_i with QoS constraints satisfied. The reward optimization problem is to find the best way to allocate calls to trunks on the m paths such that QoS constraints are satisfied and the reward rate received by the switch is maximized.

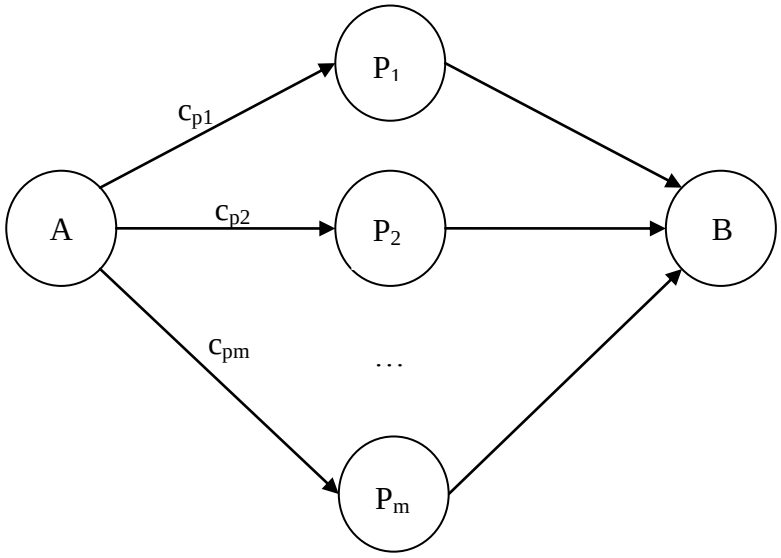

Figure 9-4: Intelligent Switch Routing Application.

Below we provide a solution to this problem by applying design principles of CAC algorithms designed for reward optimization with QoS guarantees. Certainly we will first sort all *m* paths in the increasing order of d_{pi}, say, from P_1 through P_m, and admit calls to these *m* paths in the same order to maximize the reward obtainable by the switch as long as QoS constraints are satisfied. Thus in principle each physical path P_i with c_{pi} trunks can be considered as a separate system resource and we could apply a CAC algorithm developed to each path, one at a time, starting from P_1 and then P_2 etc. until all paths are allocated.

While all CAC algorithms can be applied, spillover CAC is especially well-situated to solve this problem because calls that cannot be accommodated in path P_i can spill into P_{i+1}. Three resource allocation policies under spillover CAC can be applied. The first policy is to allocate trunks based on paths, as illustrate by the top diagram in Figure 9-5 for a system with 2 paths and 4 service classes. This path-based resource

allocation policy essentially treats each path as a separate resource and applies spillover CAC separately to P_1 and P_2 with spillover calls going from P_1 to P_2. The second policy is to allocate trunks based on classes, as illustrated by the middle diagram in Figure 9-5. This class-based policy breaks the physical boundary of paths and their associated trunks and simply allocates all trunks into 4 partitions for the 4 service classes. Spillover calls will flow from one partition to the next partition as in spillover CAC.

The third policy is based on the observation that there are two "reward" dimensions to the problem, namely, a reward of v^j for routing a class j call is received once the routing service is rendered and a charge of $k^j d_{pi}$ is paid once k^j trunks on path P_i is selected to route the call. Thus the third policy divides all trunks sorted in the order of increasing pricing into $m+n$-1 partitions with n classes and m paths, as illustrated by the bottom diagram in Figure 9-5. This path-class based policy also partitions all trunks into 4 partitions for 4 service classes. However, it observes the physical path boundary. That is, if a class partition cuts across the path resource boundary, two physically separate partitions in two physically separate paths will be used to compose the class partition. We assert that the third policy will perform the best among all policies.

The optimization problem is to find the best way to partition trunks in physically separate paths with distinct prices to service calls such that QoS constraints of calls are satisfied while maximizing the net reward obtained by the switch. The optimal solution can be found by running fast spillover CAC discussed in Chapter 6.

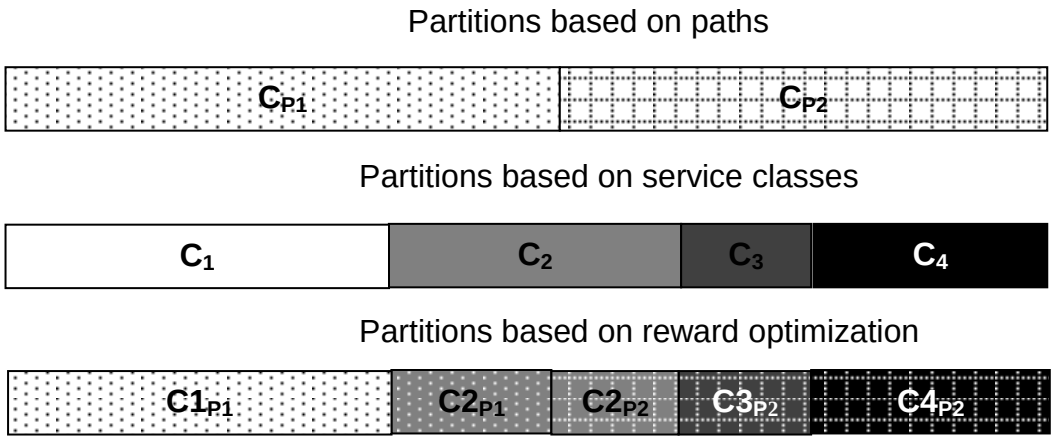

Figure 9-5: Partitioning Trunks with 2 Paths and 4 Service Classes.

9.3. Summary

In this Chapter we exemplified with two applications for the applicability of the dissertation research, namely, the optimal pricing determination application and the intelligent switch routing application. We demonstrated how one can easily apply CAC algorithms developed for these applications to determine optimal settings that will yield the best reward obtainable by the system while satisfying QoS constraints. These two applications fall within a general class of applications which concern reward optimization and QoS satisfaction for servicing multiple classes in computer networks. These applications can be easily applied to the application framework developed in the dissertation research.

CHAPTER 10 **Conclusions**

This dissertation research concerns resource management for reward optimization with Quality of Service (QoS) guarantees in wireless networks that provide multiple priority service classes to roaming mobile users. The principles developed in the dissertation research, i.e., reward-optimization CAC algorithm design and application principles, are generally applicable despite rapidly evolving wireless network technologies. To apply the CAC algorithms developed, we develop an application framework consisting of three stages: workload characterization, CAC algorithm development for reward optimization with QoS guarantees, and application of CAC algorithms developed illustrated with the optimal pricing determination application and the intelligent switch routing application. Research hypotheses have been formed and supported with data and evidences.

10.1. CAC Design Techniques and Principles

Our analysis of CAC algorithms for reward optimization and QoS guarantees hinges on several design principles and techniques. These design principles and techniques help the design of better CAC algorithms in the future.

The CAC algorithms developed are designed to manage channel resources such that the reward obtained by the system is maximized while QoS constraints for distinct service call types are satisfied. Two main techniques have been used to allocate channel

resources, namely, resource partitioning and setting thresholds. Another technique is probabilistic call admission by which a call will be accepted probabilistically after the low threshold is reached. With resource partitioning, we divide channel resources into partitions such that each partition accepts a distinct set of service call types. The idea behind strict resource partitioning is to satisfy stringent QoS constraints of some service call types. While this technique is very powerful to satisfy stringent QoS constraints, it reduces channel utilization by reducing the degree of resource sharing. For example, if a partition is reserved only for class 1 handoff calls, when a class 1 new call arrives even though there are idle channels in this partition, the system will reject the new call if the partition reserved for the class 1 new call is full. Thus, while resource partitioning can be affective in satisfaction of QoS constraints, it reduces resource sharing which results in low channel utilization and reward rate obtainable by the system.

To remedy the drawback associated with strict resource partitioning, we have followed the design principles of "resource sharing" and "more resources allocated to more important classes" (subject to the condition that QoS constraints are satisfied) and developed spillover CAC that partitions channels based on the QoS constraint and reward of each service class. Specifically the service call type with the most stringent QoS constraints and highest reward is allowed to occupy all partitions, while the service type with the least stringent QoS constraints and lowest reward is allowed to occupy only the common partition which accepts all service call types. By creating a common partition that accepts all service call types, we increase the degree of resource sharing. By allocating more partitions to service classes with higher QoS constraints, we allocate more resources to more important service classes for the purpose of reward optimization

and QoS satisfaction. The advantage of spillover CAC is evidenced by spillover CAC being able to outperform partitioning CAC based on strict resource partitioning and even CAC algorithms based on setting thresholds in terms of solution optimality. Another design principle leaned is that the mathematical model for analyzing partitioning-based algorithms is less complicated and the solution technique is more tractable often with closed-form solutions. Consequently, partitioning-based CAC lends itself to high solution efficiency.

The design principle "resource sharing", however, introduces race conditions among service call types. For example, if all service classes share channel resources in a partition, then a high arrival rate of a low priority service call type may cause starvation and consequently QoS violation of a high priority service call type. The threshold technique remedies this problem. It preserves the design principle of resource sharing but sets thresholds to throttle less important service call types, thus effectively providing more channel resources to more important service call types. The advantage of threshold-based CAC is evidenced by the fact that threshold-based CAC performs much better than partitioning CAC in solution optimality.

One drawback of threshold-based CAC is that the threshold set for a service call type represents a binary cut-off point after which calls will be rejected. This creates undesirable situations in cases there are still channels available to use by the service call but it has to be rejected because of the hard threshold. We have developed elastic CAC based on probabilistic call acceptance to reject only a fraction of calls rather than all calls after a threshold value is reached. By means of a pair of thresholds for each service call type, with the high threshold representing the binary cutoff point and the low threshold

representing the probabilistic cutoff point, elastic CAC increases the degree of resource sharing through a higher degree of channel multiplexing, thus resulting in less calls being rejected to better satisfy imposed QoS requirements. The design principle learned here is that for threshold-based CAC algorithms probabilistic call admission based on double thresholds per service call type is better than binary call admission based on a single threshold for solution optimality. Another design principle leaned is that the mathematical model for analyzing CAC algorithms based on setting thresholds is generally more complicated. Further, the solution technique used to solve the mathematical model does not lend itself to closed-form solutions and thus numerical methods often must be applied to solve the mathematical model (which in our research is a Stochastic Petri net model).

We also learn that for CAC algorithm design, there is often a tradeoff between solution optimality vs. solution efficiency. Elastic CAC performs the worst in solution efficiency while it performs the best in solution optimality. The use of double thresholds increases the search space and also improves solution optimality at the expense of solution efficiency. At the other extreme, partitioning CAC performs the best in solution efficiency because of strict resource partitioning, but performs significantly poorly compared to all other CAC algorithms in solution optimality because of failing to follow the design principle of resource sharing. Conversely, spillover CAC adheres to the design principle of resource sharing for achieving solution optimality as well as the design principle of resource partitioning for achieving solution efficiency. Thus, spillover CAC is the CAC algorithm of choice when both solution optimality and solution efficiency are considered equally important.

10.2. Publications

The dissertation research work thus far has resulted in the following publications:

[P1] I.R. Chen, O. Yilmaz and I.L. Yen, "Admission control algorithms for reward optimization with QoS guarantees in mobile wireless networks," *Wireless Personal Communications*, Vol. 38, No. 3, Aug. 2006, pp. 357-376.

[P2] O. Yilmaz, I.R. Chen, G. Kulczycki, and B. Frakes "Spillover call admission control for reward optimization with QoS guarantees for multiple service classes in mobile wireless networks," *IEEE 3rd International Workshop on Performance Analysis and Enhancement of Wireless Networks*, Okinawa, Japan, March 2008.

[P3] O. Yilmaz and I.R. Chen, "On QoS guarantees with reward optimization for servicing multiple priority classes in wireless networks," *IEEE 17th International Conference on Computer Communications and Networks*, St. Thomas, US Virgin Islands, August 2008, pp 1 – 6.

[P4] O. Yilmaz and I.R. Chen, "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*, Minneapolis, MN, July 2006, pp. 605 – 612.

[P5] O. Yilmaz and I.R. Chen, "Utilizing call admission control for pricing optimization of multiple service classes in wireless cellular networks," accepted to *Computer Communications*, Nov. 2008.

[P6] O. Yilmaz and I.R. Chen, "Comparative performance analysis of CAC reward optimization algorithms in wireless networks" accepted to *23th IEEE Advanced Intelligent Networking Applications*, Bradford, UK, May 2009.

The dissertation research has generated the following paper submissions:

[P7] O. Yilmaz, I.R. Chen, G. Kulczycki, and B. Frakes "Performance analysis of spillover-partitioning call admission control in mobile wireless networks," submitted to *Wireless Personal Communications*, Feb. 2008, revised, Oct. 2008.

[P8] O. Yilmaz and I.R. Chen, "Elastic Threshold-Based Admission Control for QoS Satisfaction with Reward Optimization for Servicing Multiple Priority Classes in Wireless Networks" submitted to *Information Processing Letters,* May 2008.

[P1] develops a simple and efficient learning mechanism that utilizes historical data for calculating arrival and departure rates of service calls from the perspective of a cell. The workload characterization algorithm presented in Chapter 4 is mainly based on this paper. This paper also develops and analyzes partitioning-based CAC and threshold-based CAC vs. partitioning-threshold hybrid CAC which combines the benefit of these two algorithms to improve the maximal reward obtainable while satisfying QoS of service classes. Chapter 5 is based on this paper. [P2] and [P7] (extended version) present spillover CAC and compare the performance of this algorithm with partitioning, threshold-based and hybrid CAC algorithms. In [P7] we also perform complexity analysis of pure vs. fast spillover CAC algorithms. Chapter 6 is largely based on these two papers. [P3] and [P8] (extended version) present elastic threshold-based CAC which greatly improves search optimality. Chapter 7 is based on these two papers. [P6] performs a comparative analysis of elastic threshold-based CAC vs. partitioning, threshold-based, hybrid, and spillover CAC to tradeoff solution optimality vs. solution efficiency. Chapter 8 is based on this paper and partially based on [P2] and [P7]. Finally, [P4] and [P5] (extended version) utilize CAC algorithms based on the application framework for pricing optimization of multiple service classes in PCS wireless networks. Chapter 9 is largely based on these two papers.

10.3. Limitations and Future Research Directions

Finally, we discuss limitations of our dissertation research and lay out future research work. We view solution complexity as a limitation for which efficient heuristic search may be used to relax the limitation. We also view solution techniques for solving SPN models to obtain "exact" solutions as a limitation for which closed-form analytical expressions to obtain "approximate" or "upper-bound" solutions may be used to relax this limitation. In the future we wish to derive performance upper bounds to understand the theoretical performance bound a CAC algorithm can ever obtain for guiding the design and analysis of CAC algorithms for reward optimization and QoS satisfaction. There are a few future research directions extending from the dissertation research work, including (a) considering other reward charge models (e.g., charge-by-connection) and investigating optimal resource allocation settings under which hybrid, spillover or elastic CAC algorithms can yield even higher rewards with QoS guarantees; (b) considering other reward collection models, e.g., reward is collected only on call termination and/or reward is lost when a call is terminated prematurely; (c) exploring further design principles for the development of CAC algorithms outside the class of partitioning-based and/or threshold-based CAC algorithms developed in the dissertation research; (d) investigating the effect of pricing-demand functions on optimal pricing, including those that consider cross-pricing elasticity of different service types; (e) extending the research on optimal pricing to dynamic and adaptive optimal pricing for solving intelligent routing and congestion control problems; (f) empirically evaluating the intelligent switch routing application and exploring other applications to further demonstrate the applicability of the CAC algorithm design principles developed in the dissertation research.

ACRONYMS AND NOTATIONS

ACRONYMS

BS	Base station
BSC	Base station controller
CAC	Call admission control
CDMA	Code division multiple access
GPS	Global positioning system
HH	High class 1 and high class 2 arrival/departure rates
KAIST	Korea Advanced Institute of Science and Technology
LAN	Local area network
LM	Low class 1 and medium class 2 arrival/departure rates
ML	Medium class 1 and low class 2 arrival/departure rates
PCS	Personal communication service
PSTN	Public switched telephone network
QoS	Quality of service
SPN	Stochastic Petri nets

NOTATIONS

a^i	Proportionality constant of service class *i*
B^i_h	Blocking probability of service class *i* handoff calls
B^i_n	Blocking probability of service class *i* new calls
B^i_ht	Threshold blocking probability of service class *i* handoff calls
B^i_nt	Threshold blocking probability of service class *i* new calls
$c\_min^i_h$	Minimum number of channels needed to satisfy the handoff call QoS constraints for class *i*

$c\_min^i_n$	Minimum number of channels needed to satisfy the new call QoS constraints for class *i*
C	Number of wireless channels in a cell
C_x	Number of channels in partition *x*
C^i_h	Channels for service class *i* handoff calls
C^i_n	Channels for service class *i* new calls
C^i_{hT}	Threshold for class *i* handoff calls
C^i_{nT}	Threshold for class *i* new calls
C_S	Shared partition of hybrid CAC
C_T	Threshold separating high and low priority calls
E^i_h	Enabling predicate of class *i* handoff calls
E^i_n	Enabling predicate of class *i* new calls
Es^i_h	Enabling predicate of service class *i* handoff calls in the shared partition
Es^i_n	Enabling predicate of service class *i* new calls in the shared partition
$ETR(C, \lambda^1_h, \lambda^1_n, \lambda^2_h, \lambda^2_n)$	Reward rate of elastic CAC
ETR^i_h	Reward rate generated per unit time due to class *i* handoff calls
ETR^i_n	Reward rate generated per unit time due to class *i* new calls
$f(\lambda^i_h)$	Function of service class *i* handoff call arrival rate
$f(\lambda^i_n)$	Function of service class *i* new call arrival rate
$HR(C,\lambda^1_h,\lambda^1_n,\lambda^2_h,\lambda^2_n)$	Reward rate of hybrid CAC
k^i	Number of channels required for a class *i* call
LTh^i_h	Low threshold of service class *i* handoff calls
LTh^i_n	Low threshold of service class *i* new calls
$M(UC^i_h)$	Number of tokens in place UC^i_h
$M(UC^i_n)$	Number of tokens in place UC^i_n
$M(UCs^i_h)$	Number of tokens in place UCs^i_h
$M(UCs^i_n)$	Number of tokens in place UCs^i_n
N^i_{hs}	Average number of class *i* handoff calls in the shared partition
N^i_{ns}	Average number of class *i* new calls in the shared partition

N^{i}_{h}	Average number of class *i* handoff calls
N^{i}_{n}	Average number of class *i* new calls
N_{ABCD}	Number of times that mobile user entering into cell *D* (the next cell) from cell *C* (the current cell), given that the previous two cells are *A* and *B* in the sequence
N_{BCD}	Number of times that the mobile user entering into cell *D* (the next cell) from cell *C* (the current cell), given that the previous cell is *B*
P	Probability
P_{i}	Partition *i*
P^{i}_{h}	Probability of accepting a service class *i* handoff call in Elastic CAC
P^{i}_{n}	Probability of accepting a service class *i* new call in Elastic CAC
P_{ABC}	Probability that if the mobile user is in cell *B* it will go to cell *C* as the next cell given that the previous cell was *A*
P_{BC}	Probability that if the mobile user is in cell *B* it will go to cell *C* as the next cell
PR^{i}_{h}	Reward rate obtained from service class *i* handoff calls in partitioning CAC
PR^{i}_{n}	Reward rate obtained from service class *i* new calls in partitioning CAC
R_{T}	Total reward generated by each cell
R^{i}_{h}	Reward rate obtained from service class *i* handoff calls
R^{i}_{n}	Reward rate obtained from service class *i* new calls
S^{i}_{h}	Service of handoff service calls of service class *i*
S^{i}_{n}	Service of new service calls of service class *i*
$SR(C,\lambda^{1}_{h},\lambda^{i}_{h},\lambda^{1}_{h},\lambda^{i}_{h})$	Reward rate of spillover CAC
t(n)	t-stat value of samples with n degrees of freedom
T_{ABCD}	Average dwell time of the mobile user in cell *C,* given that the next cell is *D* and the previous two cells are *A* and *B*

T_{BCD}	Average dwell time of the mobile user in cell *C*, given that the previous cell is *B* and the next cell is *D*
Ts^{i}_{h}	Threshold value of class *i* handoff calls in the shared partition
Ts^{i}_{n}	Threshold value of class *i* new calls in the shared partition
UC^{i}_{h}	Place for class *i* handoff calls
UC^{i}_{n}	Place for class *i* new calls
UCs^{i}_{h}	Place for class *i* handoff calls in the shared partition
UCs^{i}_{n}	Place for class *i* new calls in the shared partition
β^{i+1}	Number of price increments
ε^{i}	Elasticity constant of service class *i*
f^{i}	Ratio between the current handoff call arrival rate and new call arrival rate of service class *i*
$\Lambda_n(C)$	Arrival rate of new calls made by the mobile device in cell *C*
$\Lambda_h(C)$	Arrival rate of handoff calls from cell *C* into its neighbor cells
$\theta_n(C)$	Departure rate of new calls by the mobile device at cell *C*
$\theta_h(C)$	Departure rate of handoff calls by the mobile device at cell *C*
λ^{i}_{h}	Handoff call arrival rate of class *i* to a cell
μ^{i}_{h}	Handoff call departure rate of class *i* out of a cell
η	Number of possible future price combinations
δ^{i}	Price increment for service class *i*
v^{i}	Reward rate of service class *i*

BIBLIOGRAPHY

[1] M. Aldebert, M. Ivaldi, and C. Roucolle, "Telecommunications demand and pricing structure: An economic analysis," *7th Int. Conf. Telecommunications Systems: Modeling and Analysis*, Nashville, TN, March 1999, pp. 255-267.

[2] A. Aljadhai and T.F. Znati, "Predictive mobility support QoS provisioning in mobile wireless environments," *IEEE Journal on Selected Areas in Communications*, Vol. 19, No. 10, pp. 1915-1930.

[3] L. Badia, M. Lindström, J. Zander, and M. Zorzi "Demand and pricing effects on the radio resource allocation of multimedia communication systems," *IEEE Global Telecommunications Conference*, Vol. 7, December 2003, pp. 4115-4121.

[4] T. Basar and R. Srikant, "Reward-maximizing pricing and capacity expansion in a many-users regime," *23st Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 1, June 2002, pp. 294-301.

[5] H. Chen, S. Kumar, and C.-C. J. Kuo, "Differentiated QoS aware priority handoff in cell-based multimedia wireless network," *Electronic Imaging 2000, IS&T/SPIE's 12th International Symposium*, San Joes, CA, January 2000, pp. 940-948.

[6] I.R. Chen and C.M. Chen, "Threshold-based admission control policies for multimedia servers," *The Computer Journal*, Vol. 39, No. 9, 1996, pp. 757-766.

[7] I.R. Chen, O. Yilmaz and I.L. Yen, "Admission control algorithms for reward optimization with QoS guarantees in mobile wireless networks," *Wireless Personal Communications*, Vol. 38, No. 3, August 2006, pp. 357-376.

[8] S.-T. Cheng and J.-L. Lin, "IPv5-based dynamic coordinated call admission control mechanism over integrated wireless networks," *IEEE Journal on Selected Areas in Communications.* 23, 2005, pp. 2093-2103.

[9] J.-G. Choi and S. Bahk, "Multiclass call admission control in QoS-sensitive CDMA networks," *IEEE International Conference on Communications.* Helsinki, Finland, June 2001, pp. 331-335.

[10] C. H. Choi, M. I. Kim, T. J. Kim, and S. J. Kim, "Adaptive bandwidth reservation mechanism using mobility probability in mobile multimedia computing

environment," *IEEE Local Computer Networks Conference*, Florida, Nov. 2000, pp. 75-85.

[11] Conway, R. W. Some tactical problems in digital simulation. *Management Science* 10, 1963, pp. 47-61.

[12] F. Davoli, M. Marchese, and M. Mongelli, "Neural decision making for decentralized pricing-based call admission control," *2005 IEEE International Conference on Communications*, Vol. 3, May 2005, pp 1555-1560.

[13] Y. Elovici, Y. Ben-Shimol, and A. Shabtai, "Per-packet pricing scheme for IP networks," *10th International Conference on Telecommunications*, Vol. 2, February - March 2003, pp.1494-1500.

[14] B. Epstein and M. Schwartz, "Reservation strategies for multi-media traffic in a wireless environment," *45th IEEE Vehicular Technology Conference*, 1995, pp. 165–169.

[15] O. Ercetin and L. Tassiulas, "Pricing and peering strategies of differentiated services content networks," *8th IEEE International Symposium on Computers and Communication*, vol.1, 2003, pp. 157 – 162.

[16] Y. Fang, "Thinning algorithms for call admission control in wireless networks," *IEEE Transactions on Computers*, Vol. 52, No. 5, May 2003, pp. 685-687.

[17] Fishman, G. S. "Grouping observations in digital simulation," *Management Science*, 24, 1978, pp. 510-521.

[18] V. Frost and B. Melamed, "Traffic modeling for telecommunications networks," *IEEE Communication Magazine*, Vol. 33, pp. 70-80, Mar. 1994.

[19] G. Le Grand and E. Horlait. "A Predictive end-to-end QoS scheme in a mobile environment," *6th IEEE Symposium on Computers and Communications*, Hammamet, Tunisia, July 2001, pp. 534-539.

[20] R. Guerin, "Queuing-blocking systems with two arrival streams and guarded channels," *IEEE Transactions on Communication*, Vol. 36, February 1988, pp. 153-163.

[21] Y.-R. Haung and J.-M. Ho, "Distributed call admission control for a heterogeneous PCS network," *IEEE Transactions on Computers*, 51, 2002. pp. 1400-1409.

[22] D. Hong and S. S. Rappaport, "Priority oriented channel access for cellular systems serving vehicular and portable radio telephones," *Communications, Speech and Vision, IEE Proceedings I* , Vol. 131, No. 5, October 1989, pp. 339-346.

[23] D. Hong and S. S. Rappaport, "Traffic model and performance analysis for cellular mobile radio telephone systems with prioritized and non-prioritized handoff procedures," *IEEE Transactions on Vehicular Technology*, Vol. VFT35. No.3, pp. 77-92, August 1986.

[24] J. Hou, J. Yang and S. Papavassiliou, "Integration of pricing with call admission control to meet QoS requirements in cellular networks,'' *IEEE Trans. on Parallel and Distributed Systems*, Vol. 13, No. 9, September 2002, pp. 898-910.

[25] L. Huang, S. Kumar, and C.-C. J. Kuo, "Adaptive resource allocation for multimedia QoS management in wireless networks," *IEEE Transactions on Vehicular Technology*, Vol. 53, 2004, pp. 547-558.

[26] D. Jeong, D.-K. Choi, and Y. Cho. "The performance analysis of QoS provisioning method with buffering and CAC in the multimedia wireless internet," *54th IEEE Vehicular Technology Conference*, Atlantic City, New Jersey, October, pp. 807-811.

[27] N. Jin, G. Venkitachalam and S. Jordan, "Dynamic pricing of network resources," *IEEE Global Telecommunications Conference*, Vol. 6, December 2003, pp. 3216 – 3220.

[28] D.B. Johnson and D.A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing*, 1996, pp. 153-181.

[29] J. Joutsensalo and T. Hamalainen, "QoS aware adaptive pricing for network services," *IEEE Global Telecommunications Conference*, Vol. 4, November 2001, pp. 2455-2459.

[30] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, Vol. 8, 1997, pp. 1-11.

[31] F.P. Kelly. "Routing and capacity allocation in networks with trunk reservation" *Mathematics of Operations Research*, Vol. 15, 1990, pp. 771-792.

[32] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, Vol. 49, March 1998, pp. 237-252.

[33] N. J. Keon and G. Anandalingam, "Optimal pricing for multiple services in telecommunications networks offering quality-of-service guarantees," *IEEE/ACM Trans. on Networking*, Vol. 11, No. 1, February 2003, pp. 65-80.

[34] P. B. Key and D. R. McAuley, "Differential QoS and pricing in networks: where flow control meets game theory," *IEE Proc Software*, Vol. 146, No. 1, February 1999, pp. 39-43.

[35] M. Kyriakakos, N. Frangiadakis, L. Merakos, and S. Hadjiefthymiades, "Enhanced path prediction for network resource management in wireless LANs," *IEEE Wireless Communications*, Vol. 10, No. 6, December 2003, pp. 62-69.

[36] F.S. Lai, J. Misic, and S.T. Chanson. "Complete sharing versus partitioning: quality of service management for wireless multimedia networks," *7th International Conference on Computer Communications and Networks,* Lafayette, Louisiana, October 1998, pp. 584-593.

[37] S. Lanning, D. Mitra, Q. Wang, and M. Wright, "Optimal planning for optical transport networks," Philosph. Trans. Royal Soc. London A, Vol. 358, No. 1773, August 2000, pp. 2183-2196.

[38] Law, A. M., and J. S. Carson. "A sequential procedure for determining the length of a steady-state simulation," *Operations Research,* 27, 1979, pp. 1011-1025.

[39] B. Li, C. Lin, and S.T. Chanson, "Analysis of a hybrid cutoff priority algorithm for multiple classes of traffic in multimedia wireless networks," *Wireless Networks*, Vol. 4, 1998, pp. 279-290.

[40] Y. B. Lin and I. Chlamtac, *Wireless and Mobile Network Architecture*, John Wiley and Sons, 2001.

[41] L.-L. Lu and J.-L. C. Wu, "Handoff prediction by mobility characteristics in wireless broadband networks," *6th IEEE International Symposium on World of Wireless Mobile and Multimedia Networks.* Taormina, Italy, June 2005, pp. 469-471.

[42] J. K. MacKie-Mason, H. R. Varian, "Pricing the Internet," in *Public Access to the Internet*, MIT Press, Cambridge, MA, 1995.

[43] J. K. MacKie-Mason and H. R. Varian, "Pricing the congestible network resources," *IEEE Journal of Selected Areas in Communications*, Vol. 13, 1995, pp. 1141-1149.

[44] J. K. MacKie-Mason, L. Murphy, and J. Murphy, "Responsive pricing in the Internet," *Internet Economics*, Eds McKnight and Bailey, 1997.

[45] P. Marbach and R. Berry. "Downlink resource allocation and pricing for wireless networks," *Proceedings of INFOCOM'02*, New York, USA, June 2002, vol. 3, pp. 1470 - 1479.

[46] B.L. Miller. "A queuing reward system with several customer classes," *Management Science*, Vol. 16, 1969, 234-245.

[47] C. Milne, "Telecoms demand: Measures for improving affordability in developing countries. A toolkit for action. Main report," *WDR Dialogue Theme 3rd cycle Discussion Paper WDR0603*, June 2006, http://regulateonline.org.

[48] N. Nasser and H. Hassanein, "Prioritized multi-class adaptive framework for multimedia wireless networks," *IEEE International Conference on Communications.* Paris, France, June 2004, pp. 4295 - 4300.

[49] S. E. Ogbonmwan, W. Li and D. Kazakos. "Multi-threshold bandwidth reservation scheme of an integrated voice/data wireless network," *2005 International Conference on Wireless Networks, Communications and Mobile Computing*, Maui, Hawaii, June 2005, pp. 225-231.

[50] C. Oliveira, J.B. Kim, and T. Suda, "An adaptive bandwidth reservation scheme for high-speed multimedia wireless networks," *IEEE JSAC*, Vol. 16, No. 6, August 1998, pp. 858-874.

[51] I. C. Paschalidis and L. Yong "Pricing in multiservice loss networks: static pricing, asymptotic optimality, and demand substitution effects," *IEEE/ACM Transactions on Networking*, Vol. 10, Issue 3, June 2002, pp. 425 – 438.

[52] I. C. Paschalidis and J. N. Tsitsiklis, "Congestion-dependent pricing of network services," *IEEE/ACM Transactions on Networking*, Vol. 8, Issue 2, April 2000, pp. 171 – 184.

[53] S. D. Patek and E. Campos-Nanez, "Pricing of dialup services: An example of congestion-dependent pricing in the Internet," *39th IEEE Conference on Decision and Control*, Vol. 3, 2000, pp. 2295-2301.

[54] J. M. Peha, "Dynamic pricing as congestion control in ATM networks," *IEEE Global Telecommunications Conference*, Vol. 3, November 1997, pp. 1367 – 1372.

[55] P. Rappaport, J. Alleman, and L. D. Taylor, "Household demand for wireless telephony: An empirical analysis," *31st Annual Telecommunications Policy Research Conf.*, Arlington, VA, September 2003, http://web.si.umich.edu/tprc/papers/2003/215/HouseholdWirelessDemand2.pdf.

[56] I. Rhee, M. Shin, S. Hong, K. Lee, and S. Chong, "On the levy-walk nature of human mobility," Tech. Rep., NCSU, 2007, http://netsrv.csc.ncsu.edu/levy-mobility/.

[57] T. Sim, "Trunk reservation analysis of TELUS' Edmonton telecommunication network," *Information Systems and Operational Research Journal*, Vol. 41, No. 3, 2003, pp. 275-286.

[58] R. Simon, W.S. Chang, and B. Jukic, "Network path pricing: A QoS-based model," *8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, August 2000, pp 457-465.

[59] W.-S. Soh and H.S. Kim, "Dynamic bandwidth reservation in cellular networks using road topology based mobility predictions," *IEEE INFOCOM*, Hong Kong, March 2004, pp. 2765-2777.

[60] H. R. Varian. *Intermediate Microeconomics: A Modern Approach.* W. W. Norton and Company, 1999.

[61] J. Wang, Q. Zeng, and D.P. Agrawal, "Performance analysis of a preemptive and priority reservation handoff algorithm for integrated service-based wireless mobile networks," *IEEE Transactions on Mobile Computing*, Vol. 2, No. 1, January-March 2003, pp. 65-75.

[62] Q. Wang, T.X. Brown, "Pricing versus admission control in multi-class loss networks", *Proceedings of 40th annual Conference on Information on Sciences and Systems*, Princeton, NJ, March, 2006, pp 265-270.

[63] M. Wu, W.E. Wong, and J. J. Li, "Performance evaluation of predictive handoff scheme with channel borrowing," *2003 IEEE International Performance, Computing, and Communications Conference.* Phoenix, Arizona, April 2003, pp. 531-536.

[64] X. Xiaochun and W. Xiaoyan, "Reward-maximizing pricing and resource allocation in a multi-service network," *The International Conference on Communication Technology*, Vol. 1, April 2003, pp. 135-138.

[65] S.-T. Yang and A. Ephiremides, "On the optimality of complete sharing policies of resource allocation," *IEEE 35th Decision and Control,* Kobe, Japan, December 1996, pp. 299-300.

[66] J. Ye, J. Hou and S. Papavassilliou, "A comprehensive resource management for next generation wireless networks," *IEEE Transactions on Mobile Computing*, Vol. 1, No. 4, October-December 2002, pp. 249-263.

[67] S. Yaipairoj and F. C. Harmantzis, "A dynamic pricing model for data services in GPRS networks," *IEEE Global Telecommunications Conference Workshops*, Nov. 2004, pp. 453 – 458.

[68] O. Yilmaz and I.R. Chen, "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*, Minneapolis, MN, July 2006, pp. 605 – 612.

[69] O. Yilmaz and I.R. Chen, "Utilizing call admission control for pricing optimization of multiple service classes in wireless cellular networks," accepted to *Computer Communications*, Nov. 2008.

[70] O. Yilmaz and I.R. Chen, "Comparative performance analysis of CAC reward optimization algorithms in wireless networks" accepted to *23th IEEE Advanced Intelligent Networking Applications*, Bradford, UK, May 2009.

[71] O. Yilmaz, I.R. Chen, G. Kulczycki, and B. Frakes "Spillover call admission control for reward optimization with QoS guarantees for multiple service classes in mobile wireless networks," *IEEE 3rd International Workshop on Performance Analysis and Enhancement of Wireless Networks, AINA Workshops,* Okinawa, Japan, March 2008, pp. 1299 – 1304.

[72] O. Yilmaz, I.R. Chen, G. Kulczycki, and B. Frakes "Performance Analysis of Spillover-Partitioning Call Admission Control in Mobile Wireless Networks," submitted to *Wireless Personal Communications,* Feb, 2008, revised, Oct. 2008.

[73] O. Yilmaz, and I.R. Chen, "On QoS guarantees with reward optimization for servicing multiple priority classes in wireless networks," *IEEE 17th International Conference on Computer Communications and Networks*, St. Thomas, US Virgin Islands, August 2008, pp 1 – 6.

[74] O. Yilmaz, and I.R. Chen, "Elastic Threshold-Based Admission Control for QoS Satisfaction with Reward Optimization for Servicing Multiple Priority Classes in Wireless Networks" submitted to *Information Processing Letters,* May, 2008.

[75] M. Yuksel and S. Kalyanaraman, "Simulating the smart market pricing scheme on differentiated services architecture," *Communication Networks and Distributed Systems Modeling and Simulation Conference part of SCS Western Multi-Conference,* 2001, pp 49-54.

[76] M. Yuksel and S. Kalyanaraman, "A strategy for implementing smart market pricing scheme on DiffServ," *IEEE Global Telecommunications Conference*, Vol. 2, November 2002, pp. 1430-1434.

[77] M. Yuksel and S. Kalyanaraman, "Elasticity considerations for optimal pricing of networks," *8th IEEE International Symposium on Computers and Communication*, Vol. 1, 2003, pp. 163-168.

[78] Y. Zhang and D. Liu, "An adaptive algorithm for call admission control in wireless networks," *IEEE Global Telecommunications Conference*, San Antonio, Texas, November 2001, pp.3628-3632.